Human Feedback in Statistical Machine Translation

The University Of Sheffield.

Varvara Logacheva

Department of Computer Science
University of Sheffield

This dissertation is submitted for the degree of
Doctor of Philosophy

October 2017

Acknowledgements

First and foremost, I am enormously grateful to my supervisor Professor Lucia Specia for her gentle supervision and support, for never leaving me alone and never pushing.

I'd also like to thank my examiners Professor Kalina Bontcheva and Professor Lieve Macken for their careful attention to my thesis and for making my viva so smooth and enjoyable.

Thanks to my awesome collaborators: Chris Hokamp from Dublin City University, Fred Blain, Kashif Shah and Michał Łukasik from the University of Sheffield. Besides contributing to my work they were really good friends and made me feel a part of a team.

Sheffield's NLP group was a perfect place to work in. Thanks to everyone for our chats over tea and coffee, for lunches and dinners, nights at Red Deer and Devonshire Cat — that was almost as important as research itself.

During my PhD I was lucky to have internships in outstanding places: Dublin City University and Unbabel. Thanks to Chris for welcoming me in Dublin, to João and the whole Unbabel team for making me feel at home in Lisbon.

Thanks to people whose influence brought me to Sheffield: to Eduard Klyshinsky for being my first mentor in NLP, to Francis Tyers for introducing me to Machine Translation, to Andrey Fedorovsky for showing a real passion for research.

Finally, I would never get to this point without my family: my parents Maria and Konstantin, my sister Liza and my husband Stas. Their belief in my superpower has always helped to carry on.

Abstract

The thesis addresses the challenge of improving Statistical Machine Translation (SMT) systems via feedback given by humans on translation quality. The amount of human feedback available to systems is inherently low due to cost and time limitations. One of our goals is to simulate such information by automatically generating pseudo-human feedback. This is performed using Quality Estimation (QE) models. QE is a technique for predicting the quality of automatic translations without comparing them to oracle (human) translations, traditionally at the sentence or word levels. QE models are trained on a small collection of automatic translations manually labelled for quality, and then can predict the quality of any number of unseen translations.

We propose a number of improvements for QE models in order to increase the reliability of pseudo-human feedback. These include strategies to artificially generate instances for settings where QE training data is scarce. We also introduce a new level of granularity for QE: the level of phrases. This level aims to improve the quality of QE predictions by better modelling inter-dependencies among errors at word level, and in ways that are tailored to phrase-based SMT, where the basic unit of translation is a phrase. This can thus facilitate work on incorporating human feedback during the translation process.

Finally, we introduce approaches to incorporate pseudo-human feedback in the form of QE predictions in SMT systems. More specifically, we use quality predictions to select the best translation from a number of alternative suggestions produced by SMT systems, and integrate QE predictions into an SMT system decoder in order to guide the translation generation process.

Table of contents

Chapter 1

Introduction

Machine translation (MT) and, in particular, statistical machine translation (SMT) has reached a level of quality which allows it to be used in a number of applications. It is still virtually impossible to generate an error-free machine-translated text, however, automatic translation is often good enough for gisting purposes, and after manual revision it can be used for publication. At the same time, this improvement in quality has enabled users to produce meaningful feedback on MT, which can be used to further improve it. While the output of the first MT systems was so bad that it was hardly possible to mark any particular errors, now MT errors can often be localised and classified. This allows to develop new components of MT systems which can improve the translation quality by tackling the problems identified by users.

Feedback can be collected in many ways, including informal ones, e.g. comments on translation. However, there is one type of feedback that is both informative and potentially available in large quantities. That is manual post-editing of automatically translated texts. Nowadays the majority of professional translators translate texts using pre-generated draft translations which they edit to get the final output. These translations can come from a database of previously translated documents, however, many computer-assisted translation (CAT) tools allow using MT systems to generate a draft. In this case the human feedback on MT quality is a by-product of the translator's work. Thus, it can be available in large amounts and its collection does not need any special arrangements nor additional training for annotators.

There exist multiple types of MT model architectures. The earliest MT systems were rule-based: they performed translation applying a set of hand-written or automatically generated rules, and the examples of rule-based MT systems can still be useful for some particular settings (e.g. `Apertium`[1] which is used to translate between closely related under-

[1] https://www.apertium.org/

resourced languages, and TectoMT[2] shows that rule-based MT combined with some statistical information can be efficient for highly inflected languages). Later, the SMT proved more effective (Koehn et al., 2003). The advantage of SMT is that it does not need pre-defined translation rules — it extracts the correspondences between words and phrases of the source and the target languages from the parallel data. The recent emergence of Neural Machine Translation (NMT) models (Bahdanau et al., 2014) set the research in MT on a different track. It also uses parallel data for training, but the neural network architecture allows to address some flaws of SMT. However, despite the fact that NMT models have lately been shown to outperform SMT, our research is based on SMT and aims at incorporating human feedback into statistical MT models. The reasons for that are the facts that such models were predominant in the MT community when we started the research, and that NMT is still not well-studied and cannot beat SMT for many language pairs. Therefore, further in the thesis the abbreviation MT will denote SMT, unless specified otherwise.

Post-edited data can be incorporated into an SMT system in many ways. They can be coarsely divided into into two broad groups: the *interactive* and *offline* strategies. The difference between them is that in scenarios we call *interactive* the aim is to change a system shortly after a correction is made. This is usually referred to as the "human-in-the-loop" scenario: the tool evolves as a user works with it so that an improved version of the MT system is available during the same work session. On the other hand, the *offline* methods accumulate some number of user corrections and then learn from them in an offline mode.

The *offline* methods are represented by research on automatic post-editing and error correction tools: stand-alone tools on top of an MT system which are trained on human feedback and can improve automatic translations using statistics extracted from post-edits. The *interactive* methods are designed for the following scenario. A translator translates a text using a CAT tool. There is an SMT system underlying this tool. When the user starts translating the document s/he corrects the output of MT system. These corrections are used to update different components of the MT system: they are fed to an update algorithm one by one or in small batches, and after a short time the user gets the output of an updated system. Any further corrections are also incorporated into the system so that it constantly improves while the user is working. The primary goal of this process is to increase user productivity by avoiding the need to correct the same MT errors.

Two of the errors that users often correct are mistranslation, i.e. incorrect translations of single words, or words that are left untranslated because they are unknown to the system. These errors can be fixed by adding appropriate translations into translation tables. There are many methods to perform these updates: translations can be added to main translation

[2]https://ufal.mff.cuni.cz/tectomt

tables (in this case they should be assigned a proper score) or form a new domain-specific (or even document-specific) translation table (this solution requires an appropriate translation table combination technique). The language model (LM) can also be updated in a similar way, although this task is not as challenging as translation table improvement, because LM requires only monolingual data for training. Finally, the translator corrections can be used to change the weight given to particular components of an MT system. This process is different from the previously described ones, because it uses only sentence-level scores, without making use of the information on erroneous sentence segments and their corrected versions.

Both *interactive* and *offline* methodologies are more effective when there is a domain shift, i.e. when the domain of a new text to translate is different from that of the training data of the baseline MT system. That is particularly notable in systems that update MT systems' translation models. If the target domain (the domain of the newly translated data) has many technical terms, then the translation of text in this domain performed with an out-of-domain MT system will contain many out-of-vocabulary words, and some of the terms that exist in the original corpus can have alternative domain-specific translations that did not occur in the training data. These errors can be fixed if the appropriate translations are added into the MT system's translation table. On the contrary, errors of other types are not so easy to correct as they usually involve context, and the available post-edits are often not numerous enough to list all possible contexts or to generalise them. Therefore, if post-edits cover specifically wrong translations of terms, they can improve the MT system quality, otherwise their incorporation into an MT system brings little or no improvement.

So far the *interactive* methodology has been better explored and proved more effective, although it also has its limitations: it is more effective when adapting an MT system to a strict domain with many repetitions and little variation in syntax, and brings less improvement when translating texts of heterogeneous and less formal domains. However, despite its convenience, sometimes the "human-in-the-loop" scenario is inapplicable. It does not suit for cases where human feedback is not generated naturally (as in case of translators working with CAT tools) and its collection needs to be arranged specifically — it means that this process is expensive and cannot be repeated often. In such cases human feedback is scarce and should be reused as much as possible. This brings us back to *offline* methods that make use of pre-collected user data.

The main problem of *offline* methods is the insufficient amount of data available. One of the possible solutions is to try to mitigate this problem by finding an efficient way of acquiring new data. Given that human effort is expensive and should be minimised, we can acquire data automatically by simulating user feedback. In fact, that is the motivation

of automatic post-editing systems: they try to simulate human actions (post-editing) thus propagating human feedback onto unseen automatic translations. However, we can consider human feedback in a different way. Post-edition can be regarded as the information on whether a translation is good or bad. Having a small amount of this information, we can train a system which identifies the quality of an automatically translated text. Applying this system to newly translated texts we can get virtually unlimited amount of pseudo-human feedback to incorporate into an MT system.

This task is well established in MT and is usually referred to as quality estimation (QE). QE takes into account various features of the sentence and its translation and sometimes of the MT system that generated it. It is different from quality evaluation because it does not rely on an oracle translation of the source text and is based mainly on the fluency of target text and its correspondence to the source. Translation quality can be estimated at different levels: the first QE models concentrated on quality of words, whereas later it turned out that sentence-level quality is easier to estimate. Furthermore, there are ways of determining document-level quality, which depends not only on correspondence of source and target, but also on high-level coherence and consistency of the text. Document-level quality scores are not suitable for our purpose of improving MT systems. They give too little information: if we have only one score per document, we do not know anything about the particular errors the MT system makes. Moreover, document-level scores are difficult to incorporate into an MT system. The reason for that is that the majority of decoders are sentence-level — they translate sentences as independent units and therefore cannot make use of discourse features that document-level QE relies on.

Instead, we are interested in translation quality of individual words and ngrams. The motivation for this is the following. The automatically translated sentences are rarely completely wrong: in the majority of cases a part of the sentence is correct. Likewise, they are rarely completely flawless: it is common to have a mistranslated or ungrammatical phrase in an otherwise correct sentence. Therefore, locating this incorrect segment can be useful for an MT system because we can inform it of the error with higher precision. However, word- and phrase-level QE models still underperform compared to sentence-level QE models. Hence, we also investigate the use of sentence-level scores for the improvement of MT where possible, e.g. for an MT system tuning which requires sentence-level score.

1.1 Aims

The main aim of our research is to **improve MT by learning from human feedback**. This task can be decomposed into two principal components: acquisition of data and its incorporation into MT. Therefore, our research has the following subgoals:

- Generate high-quality pseudo-human feedback using Quality Estimation techniques. This is necessary because the existing human feedback is inherently scarce. Therefore, we need a way of generating human-like feedback automatically, and QE is a good way of scaling human judgements onto unseen translations and creating almost unlimited amount of feedback.
- Devise methods of incorporating quality scores computed by QE models into SMT system. While there exist techniques that use post-edits for the improvement of MT, we aim at raising the translation quality by supplying an MT system with numerical quality scores of translations.

1.2 Contributions

The main contributions of this thesis are:

- We developed a **Quality Estimation tool** `Marmot` which works at the levels of words and phrases and can be extended to other levels of granularity. It extracts a wide range of features and performs training of models of different types using Python libraries and standalone machine learning tools. The system is opensource. It is written in Python and can be easily extended (Logacheva et al., 2016b).
- We improved the **performance of word-level QE** models by suggesting new features and manipulating the data (Logacheva et al., 2015).
- We compared the metrics used for the evaluation of word-level QE models and **suggested a new metric** that is unbiased and is better at discriminating between QE models (Logacheva et al., 2016c).
- We developed a method of automatic generation of **artificial training data** for QE systems, because the available human data is sometimes insufficient even for a QE system training. The artificial data cannot be used to replace post-edits in an MT system retraining, but suits for improving the quality of sentence-level QE models (Logacheva and Specia, 2015b).

- We established a new level of granularity for QE task, namely, **QE at the level of phrases**. We notice that MT errors are context-dependent: a choice of word is influenced by this word's neighbours, especially in phrase-level SMT where phrases are handled as atomic units. This level estimates the quality of MT-like phrases to help the decoder decide if a chosen phrase is suitable or not. In order to achieve this we explore the notion of phrase itself by trying different ways of segmenting sentence into phrases and then train phrase-level QE models using different sets of phrase features (Logacheva and Specia, 2015a).

- Finally, we devise ways to **incorporate the sentence-level, word-level and phrase-level quality scores into SMT** in a number of ways: we perform N-best list reordering with QE predictions of different levels of granularity, then we tune an SMT system using quality score as an additional feature, and finally we incorporate word-level quality scores directly into the decoder.

1.3 Structure of the thesis

The work consists of three parts.

- Part 1 (Background) is devoted to our survey on previous work in the area of human feedback in Machine Translation:

  - Chapter 2 contains a survey of types of human feedback for MT and ways of using it for the improvement of MT quality. We review the works on direct use of human feedback as well as the methods of its indirect incorporation (when feedback is used to train an intermediate model that will transform the MT output) and identify the main weaknesses and limitations of the existing techniques.
  - Chapter 3 deals with different approaches to Quality Estimation for MT. Starting with the first work in confidence estimation for MT we trace the evolution of QE to the state-of-the-art models. We review two most popular directions of QE: estimation of quality of entire machine-translated sentences and of individual words in sentences. In addition, we present our new QE tool which operates at the level of words and phrases and implements the established models and features as well as our novel solutions.

- Part 2 describes our contributions to Quality Estimation:

  - In Chapter 4 we talk about strategies to improve QE systems performance that use data selection and manipulation techniques. A large part of this chapter contains the experiments on generation of artificial training data for QE models. The rest of the chapter is devoted to improvement of QE models performance by manipulating the existing data.
  - Chapter 5 assembles our work on subsentence-level QE. Here we present a new QE task: the estimation of translation quality at the level of phrases. Besides that we compare approaches to the evaluation of word-level and phrase-level QE models and suggest a new metric for evaluation of subsentence-level QE models. Finally, we report the implementation of these contributions in a new shared task on phrase-level QE.

- In Part 3 we bring together Quality Estimation and Machine Translation: we describe how we used QE for the improvement of translation quality:

  - In Chapter 6 we describe our experiments on the use of sentence-level QE scores for active selection of training data for SMT systems. We use three sentence-level QE models of different performance to analyse how the QE accuracy influences the downstream MT quality.
  - Chapter 7 describes our experiments in direct improvement of translation quality using Quality Estimation: we rescore N-best lists and perform discriminative MT system training (tuning) using sentence-level quality scores, and incorporate word-level QE scores in the SMT decoder.
  - Finally, in Chapter 8 we present our conclusions of the conducted research and outline the possible directions of the future work.

1.4 Publications

These are publications whose text is partially included in the thesis:

1. V. Logacheva, M. Lukasik and L. Specia. (2016) “Metrics for Evaluation of Word-level Machine Translation Quality Estimation”. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-2016), Berlin, Germany, August 2016, pp. 585–590.

2. O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A.J. Yepes, P. Koehn, V. Logacheva, C. Monz, M. Negri, A. Neveol, M. Neves, M. Popel, M. Post,

R. Rubino, C. Scarton, L. Specia, M. Turchi, K. Verspoor and M. Zampieri. (2016) "Findings of the 2016 Conference on Machine Translation". In Proceedings of the First Conference on Machine Translation (WMT-2016), Berlin, Germany, August 2016, pp. 131–198.

3. V. Logacheva, F. Blain, and L. Specia. (2016) "USFD's Phrase-level Quality Estimation Systems". In Proceedings of the First Conference on Machine Translation (WMT-2016), Berlin, Germany, August 2016, pp. 800–805.

4. V. Logacheva, C. Hokamp, and L. Specia. (2016) "MARMOT: A Toolkit for Translation Quality Estimation at the Word Level". In Proceedings of the 10th edition of the Language Resources and Evaluation Conference (LREC-2016), Portoroz, Slovenia, May 2016, pp. 3671–3674.

5. V. Logacheva and L. Specia. (2015) "Phrase-level Quality Estimation for Machine Translation". In Proceedings of the 12th International Workshop on Spoken Language Translation (IWSLT-2015), Da Nang, Vietnam, December 2015, pp.143–150.

6. V. Logacheva, C. Hokamp, and L. Specia, "Data enhancement and selection strategies for the word-level quality estimation". In Proceedings of the 10th Workshop on Machine Translation (WMT-2015), Lisbon, Portugal, September 2015, pp. 311–316.

7. V. Logacheva and L. Specia. (2015) "The role of artificially generated negative data for quality estimation of machine translation". In Proceedings of the 18th annual conference of the European Association for Machine Translation (EAMT-2015), Antalya, Turkey, May 2015, pp. 51–58.

These are publications related to the thesis:

1. F. Blain, V. Logacheva, and L. Specia. (2016) "Phrase Level Segmentation and Labelling of Machine Translation Errors". In Proceedings of the 10th edition of the Language Resources and Evaluation Conference (LREC-2016), Portoroz, Slovenia, May 2016, pp. 2240–2245.

2. O. Bojar, R. Chatterjee, C. Federmann, B. Haddow, C. Hokamp, M. Huck, V. Logacheva, P. Koehn, C. Monz, M. Negri, P. Pecina, M. Post, C. Scarton, L. Specia, and M. Turchi (2015) "Findings of the 2015 Workshop on Statistical Machine Translation," In Proceedings of the 10th Workshop on Machine Translation (WMT-2015), Lisbon, Portugal, September 2015, pp. 1–46.

3. K. Shah, V. Logacheva, G. Paetzold, F. Blain, D. Beck, F. Bougares, and L. Specia (2015) "Shef-nn: Translation quality estimation with neural networks". In Proceedings of the 10th Workshop on Machine Translation (WMT-2015), Lisbon, Portugal, September 2015, pp. 342–347.

4. V. Logacheva and L. Specia, (2014). Quality-based active sample selection strategy for statistical machine translation. In Proceedings of the 9th edition of the Language Resources and Evaluation Conference (LREC-2014), Reykjavik, Iceland, May 2014, pp.2690–2695.

5. V. Logacheva and L. Specia, (2014). Confidence-based Active Learning Methods for Machine Translation. In Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2014), Workshop on Humans and Computer-assisted Translation, Gothenburg, Sweden, 26 April 2014, pp. 78–83.

Part I

Background

Chapter 2

Human feedback in machine translation

There are multiple ways of getting information on the quality of machine-translated text from humans and multiple forms of this information itself. The feedback can be collected in a free form or follow some structure. Since text is a complex object, its quality has many aspects. When collecting feedback on the text quality researchers can concentrate on one or several aspects, or aim for an overall evaluation.

The type and format of feedback are also related to their application. While some types of feedback are suitable only for manual quality analysis (i.e. to inform an expert of the existing problems), other can be used by some tools that improve the translation quality, or by Machine Translation systems themselves. Our work concentrates on the latter application of feedback: we are interested in seeing how the data received from users can improve the MT quality in an automatic manner.

We start with reviewing the types of feedback in Machine Translation and the available data resources (section 2.1). Then we discuss ways to use it in Machine Translation: direct improvement of MT system components with the new data in the interactive or offline mode, training of automatic post-editing models to deal with MT errors, and indirect ways of the improvement of automatic translation quality.

2.1 Feedback

The first question to answer when dealing with human feedback about MT quality is the form of this feedback and the ways of acquiring it. In this section we review the ways of collecting and storing user evaluation of automatic translations: we look into types of feedback, some issues related to those types, and the available feedback data.

2.1.1 Types of feedback

There are multiple types of user feedback to machine translation that give information on translation quality or the usability of MT and Computer-Assisted Translation (CAT) systems. Since the focus of the research is MT quality, we consider only the types of feedback that relate to the translation quality itself, although usability can influence the quality in some cases.

Translation quality assessment can be **numerical**, i.e. express some aspect of translation quality in the form of number, or of some other type. The advantage of numerical feedback is that it has a certain format and a range of allowable values, so it is potentially easier to use in automatic ways. It can be used as input for a computer program with no pre-processing, unlike some other more informative but less formalised types of feedback.

The numerical quality score can be acquired via human evaluation of a sentence (or a piece of text of a different size) or comparison of several sentences. When enquiring users about a single sentence, we can ask them to evaluate its overall quality or some aspect of it: the **fluency** of a sentence, often associated with grammatical correctness and right word order, its **adequacy**, which focuses on the delivery of the original meaning, or some specific aspect, for example, the perceived **post-editing effort** (i.e. how easy/difficult a translation would be to manually correct), the **suitability for gisting** (a translation is considered suitable for gisting if it preserves the main idea of the original text and is readable). All these aspects of quality can be measured in different scales: binary, ternary, one to five, one to ten, etc. Alternatively, a user can be asked to **rank** two or more translations by their quality without giving them any scores. The answers can then be converted to numbers. This way of evaluation is convenient when comparing several MT systems, but does not indicate absolute translation quality, which is key for methods aiming to improve the underlying MT systems.

The numerical types of feedback (both ranking and independent evaluation) can be collected using manual evaluation tools such as `Appraise` (Federmann, 2012). There are other more task-specific feedback types which are tightly related to their acquisition and further use processes, so the feedback acquired via such procedures is usually not distributed as standalone data. One of such examples is the `Caitra` tool (Koehn, 2009): it presents a user with N-best phrase segmentations of a sentence and best translation variants for these phrases (in other words, it shows the highest-scoring part of the translation lattice), and the user can pick the segments that form the best translation. The user's choices of better translations and order of phrases potentially allow direct modification of translation and reordering models. However, no such experiments have been reported in the literature.

Other ways of user evaluation are less formal and more complex. A user can be asked to perform **post-editing** (Krings, 2001), i.e. to edit an automatic translation so that it becomes

a correct sentence in the target language. Another task is **error markup**: the user is asked to mark errors in the translation, possibly specifying the types of errors — this task is similar to post-editing, but a user only points to errors without correcting them. Notably, these types of human feedback, especially post-editing, are more resourceful: they can serve as sources of information of different types, including numerical ones. For example, the post-editing can give the numeric estimate of translation quality computed as the percentage of words that were changed. This quantity is widely used in MT and related research and is referred to as Human Translation Error Rate (**HTER**) (Snover et al., 2009). Besides, the translation quality can be deduced from post-edits in different ways: one can measure **time spent on editing** of a segment, **number of keystrokes** or mouse clicks performed during editing, **number and duration of pauses**, distribution of the whole editing time over different actions: dictionary lookup, corrections, etc.

2.1.2 Post-editing

Comparison with other types of feedback

Although there potentially exist many different feedback types, only a few of them became widely used in MT and related research topics. Many types of numerical feedback did not become widespread because of difficulties related to their collection. Since MT outputs text in natural language, there exist very many aspects of quality that could be measured: adequacy, fluency, post-editing effort as perceived by user, suitability for some specific task, etc. The scales used for labelling are also different: ternary, four-point, five-point, one to ten, the interpretations of these scores highly divergent and may vary even within one research group. Likewise, the opinions of different assessors about one sentence can be different, which makes the collected data less reliable.

In contrast to that, post-editing is self-explanatory: the task of correcting the translation is clear, although it also requires additional instructions. Moreover, many translators are used to post-editing as a part of their everyday work: some computer-assisted translation tools suggest an output of an MT system as a draft, so translation reduces to post-editing. Besides, many translation companies regularly perform post-editing of human and automatic translations for quality control, so their employees do not need any detailed training to participate in a post-editing experiment.

Another advantage of post-edits is that they are omni-purpose: they can be used in different tasks and serve as a source of multiple types of feedback. The reason for that is the richness of information provided by post-editing: it gives an idea of overall quality of the whole text and quality of its segments of any size (up to sub-word units), provides valid

replacements for incorrectly translated parts. With some additional resources at hand one could use post-edits ot extract the information on the types of errors or quality of translation of some specific phenomena (this can be achieved via automatic translation analysis tools like `Hjerson` (Popović, 2011)). This means that a collected dataset of post-edits can be then extended or reused by many other researchers, whereas a corpus labelled with a highly specific type of feedback is less usable outside a particular application.

On the other hand, the error markup is more advanced than post-editing in terms of informativeness: it allows one to define the types of errors precisely (as opposed to deducing them from post-edits) which potentially allows to better understand the flaws of an individual MT system. Moreover, unlike edits, the errors in a sentence can overlap so that one word is contained in two or more segments of different error types. Error markup is however a much more time-consuming and challenging task than post-editing. While post-editing requires a user to have good command of the source and target languages, error markup also demands good understanding of linguistics and a chosen error typology. These features made error markup much more difficult to collect than post-edits, so it did not become widespread for the purpose of MT improvement. Due to its informativeness and popularity in industry post-editing became the most wide-spread type of human feedback. The majority of research on the use of human feedback in MT deals with post-editing: previous work does not examine the different types of feedback but consider post-edits as the only option.

Post-editing also has its drawbacks: it is a time-consuming task that requires a performer to be proficient in the target language and fluent in the source language (as it was shown in (Kunchukuttan et al., 2012), the knowledge of the domain is sometimes also needed). Besides that, although post-editing task is self-explanatory, one of the main problems the developers of datasets face is that post-editing is very user-dependent and without additional guidelines the post-edits produced are not systematic, i.e. different from editor to editor. Another problem which developers face when they try to engage non-professional translators into the feedback generation is the low quality of output.

What to edit?

Despite the intuitively clear nature of post-editing task, it requires more careful formulation. Guzmán (2007) gives the classification of post-edits with respect to the degree of the changes made. The author divides post-edits into three groups: rapid post-editing, which addresses only major mistakes that hamper to understand the text, full post-editing, which aims at making a text completely fluent and correct, and minimal post-editing, which covers all the range between these two categories and therefore usually cannot be clearly defined.

The majority of researchers ask post-editors to produce as few changes as possible to convert the MT output into correct target-language text (Potet et al., 2012b). However, this guideline is quite obscure as users' opinion on what is the correct text may vary substantially. As a result, if more than one post-editors take part in post-editing, the performed corrections can be inconsistent, i.e. they can correct the same mistake in many different ways. We argue that the post-editing for further use in MT should be closer to rapid version than to full post-editing. Since current MT technology has no notion of style of text, they will not be able to make use of the corrections concerning the stylistics, on the contrary, they can create additional noise.

Low quality of feedback

The major problem of the process of collecting the human feedback is its high cost. Users need to be paid for assessing MT output. Moreover, hiring professional translators, who are presumably needed to produce post-edits, is even more expensive. Some researchers (Déchelotte, 2010; Kunchukuttan et al., 2012; Liao et al., 2011) tried to tackle this issue using crowdsourcing techniques, which significantly reduce the cost of post-editing. **Crowdsourcing** consists in collecting the feedback from non-expert and usually anonymous users via the Internet. It helps to reduce the costs, however, produces the feedback of much lower quality. For example, according to the estimates provided by Potet et al. (2012b), only one in three sentences submitted by users is usable. Déchelotte (2010) gives even worse statistics: only 9 entries out of 100 tested entries were recognised as useful. In order to improve the result users can be asked to meet some requirements concerning language proficiency (Ambati et al., 2010), but this may reduce their number.

Another way to ensure the good quality of the post-edits done by non-expert users is to **filter** them. Déchelotte (2010) suggests several criteria of automatic feedback assessment and filtering: the length of the suggested translation should be similar to the length of automatic translation, some errors or malevolence can be detected by simple heuristics: senseless successions of characters, common profanities. Liao et al. (2011) suggest a number of cross-validation techniques that are based on the assumption that if post-edits from multiple users agree, they are correct.

Pighin et al. (2012b) collect crowd feedback on an online MT system and then ask human annotators to evaluate the quality of this feedback. The annotations show the similar statistics: only 17% of post-edits were rated as good and only 13% as useful (in the case where the automatic translation was incomprehensible and the post-editing improved it). This research also improves the filtering strategy proposed earlier: the authors suggest a list of features

and learn a classifier on a set of post-edits labelled as useful or useless by an expert. The classifier outperforms the baseline, showing the maximum accuracy of 75.5%.

Kunchukuttan et al. (2012) consider different crowdsourcing settings: sentence translation, phrase verification, various payment types. Manual translation of judicial texts from English into Hindi appeared to be too complicated for non-professional users. The best result is 32% correct translations, 44% incorrect and 26% spurious translations (translations generated by other MT systems). The main problems of crowdsourced translations are:

- misspellings;
- bad translations made by users with bad knowledge of language or domain;
- translations generated by online MT tools (at times changed to disguise it).

Other settings of translations or post-edits acquisition experiments include monolingual users translating sentences from an unknown language using phrase translations provided by an SMT system (Koehn, 2010) or iterative interaction of monolingual users with source-language speakers (Hu et al., 2011).

2.1.3 Available feedback data

Post-edits

As was shown, the most informative and hence popular type of human feedback is the post-editing of automatic translation. There has recently appeared many corpora of post-edits. Such datasets usually contain a set of source sentences with their translations into the target language performed automatically by an MT system and a manual post-editing of this translation. Each sentence sometimes has a free reference (human) translation and some additional information. Since post-editing is a time-consuming task, such resources are usually not large: they contain from several hundreds to few hundred thousand examples, which is few compared to the size of parallel corpora used for training of MT systems.

One of the first datasets of this kind is the corpus of post-edits of automatic translations from French into English (Potet et al., 2012b). It contains 10,000 examples, but is was edited by non-native speakers of English, which makes it less reliable, and contains examples of irrelevant edits: some corrected words are not errors but valid alternative terms (e.g. the dataset contains multiple corrections of the word “football” into “soccer” which are equivalent words used in British and American English, respectively). This corpus also contains 1,500 reference translations that were post-edited to become closer to the automatic translations. These examples give the possibility to directly compare the automatic translations, reference translations and post-edits.

Similar corpora of comparable or smaller size were collected under the scope of FAUST (Pighin et al., 2012a,b) and TaraXU (Fishel et al., 2012) projects. They contain post-edits and ranking of automatic translations (for datasets which provide multiple translations for one source segment) for a number of European languages: English, French, Spanish, Romanian, Czech, Serbian, German, however, almost all language pairs contain English as a source or target language. The Trace corpus (Wisniewski, 2013) has data for English–French and French–English language pairs, 6,000 and 7,000 sentences, respectively. All post-edits for a language pair were done by one translator, besides that, 1,000 sentences from both language pairs contains the an alternative post-edit done by a different translator. Half of the sentences come from datasets provided by WMT, IWSLT and Word Sense Disambiguation campaigns and belong to the news domains. Another half were collected through the web portal which performs translation between French and English and belong to different domains and genres. The translations were produced with two MT systems: a rule-based and a statistical one. Autodesk Post-Editing Corpus[1] is more diverse in terms of available languages: it contains post-edited translations from English into simplified and traditional Chinese, Czech, French, German, Hungarian, Italian, Japanese, Korean, Polish, Brazilian Portuguese, Russian, Spanish with between 30,000 and 410,000 segments per language. The data contains sentences from software user manuals of Autodesk products.

The majority of the latest corpora of post-edits were created for shared tasks in Quality Estimation (QE) and Automatic Post-Editing (APE). QE aims at estimating the quality of an automatically translated text, and APE is a task of correcting MT output. Therefore, training data for both of these tasks can be extracted from post-edits. The corpora used for QE task have been growing: while the QE dataset contained only 800 examples in 2013 (Bojar et al., 2013), in 2014 there were already 2,000 for one of directions (Bojar et al., 2014), in 2015 the provided corpus contained 14,000 sentences with their post-edits (Bojar et al., 2015), in 2016 — 15,000 (Bojar et al., 2016). The set of available languages is a subset of the ones of other datasets: the WMT corpora contain translations between English and French, German or Spanish.

Other feedback

Despite being one of the most popular feedback types, post-editing is not the only type used in MT-related research. There also exist corpora of automatic translations labelled for quality by explicitly asking user to rate a sentence. The scores reflect different aspects. EAMT-09 corpus (Specia et al., 2009) contains general quality score in a 1–4 scale given by four assessors — which allows to compute inter-annotator agreement. The automatic

[1] https://autodesk.app.box.com/Autodesk-PostEditing

translations in this corpus were produced by 4 SMT systems, so their performances can also be compared. The EAMT-11 corpus (Specia, 2011) contains the same information along with post-edits of the same sentences and post-editing time, so one can compare the translation quality perceived by user and real quality reflected in post-edits. The datasets contain data for the French–English and English–Spanish language pairs, all data belongs to the news domain. Some corpora have coarser-grained scales: Turchi and Negri (2014) present a corpus of automatic translations annotated for quality with binary labels (little/much post-editing), in the corpus for WMT-14 sentence-level QE task (Bojar et al., 2014) the scale is ternary: good/almost good/bad sentences.

Another type of feedback which received some attention is the error markup. It is more complex than post-editing, because it has to be based on some error typology which may not be extensive and suitable for a particular MT architecture. One of the most popular examples of such a typology was presented in the work by Vilar et al. (2006), where incorrect segments can be assigned one of five error classes: missing words, incorrect words, unknown words, word order or punctuation errors, and each class has subclasses for fine-grained classification. The Terra corpus (Fishel et al., 2012) is an example of applying this error classification to MT. It has annotations for four language pairs (English–Czech, French–German, German–English, English–Serbian) and a set of MT models of different types. Unfortunately, the number of annotated translations ranges from only 50 to 250 for different language pairs, so the corpus can be considered only as an example of annotation and not as a data resource for experiments.

Vilar et al.'s error classification was criticised for being ad hoc and specific to a particular type of MT systems (statistical MT). Lommel et al. (2014) developed a Multidimensional Quality Metric (MQM) which is more universal and can address various aspects of quality of automatic as well as human translations. According to MQM, all errors fall within one of two major classes: accuracy and fluency. The former is further divided into terminology errors (incorrect translation of a term), mistranslation (incorrect translation of a regular word), omission of word, etc., and in the latter the authors distinguish grammatical errors (with subtypes), style, spelling and typographical errors. There also exist the third top-level error type: verity. It deals with the errors in the source data and correspondences between source and target which are not directly related to translation quality (e.g. date format accepted in different countries), so it is irrelevant in cases when the source sentence is assumed to be correct. A large dataset labelled for errors using MQM scheme has been released for the WMT-14 word-level QE shared task (Bojar et al., 2014). It contains data for four language pairs with the largest subset comprising 2,000 sentences (English–Spanish language pair).

However, MQM's complexity and large number of categories makes it hard to generalise the feedback for the purpose of improving MT.

Overall, provided the existence of large post-edits datasets, the development of corpora labelled with other types of feedback becomes less necessary, because post-edits can be converted to numerical quality scores of any level of granularity and can even give information on a limited number of types of errors. The latter will be less reliable than manual error markup, but it can be acquired in much larger amounts.

2.2 Incorporation of feedback into machine translation

Despite the various possible types of feedback, the most frequently used type is automatic translation post-edited by humans. As was discussed before, post-edits is a compromise between the amount of information given and the effort needed to produce such feedback.

There are many possible ways of incorporating post-edits into an MT system. The most evident way would be to add them to the training corpus. However, as was already pointed out, the post-edits exist in quite small amounts compared to the size of training datasets for MT models, so they are unlikely to bring any significant improvements when used as additional training data. To the best of our knowledge, there is no thorough analysis of the performance of MT systems with post-edits used as its training data. The research by Potet et al. (2011) gives some information on concatenating the post-edited sentences with the initial training data. However, their data is extremely limited: the experiments reported use only 175 post-edited sentences. The direct addition of this data to the training corpus of 1.6 million sentences does not improve the system's quality even if the post-edited sentences are given higher priority by replicating them multiple times.

2.2.1 Usual SMT pipeline

Here we provide a brief outline of the main components of a MT system that can be improved with the human feedback data. An MT system is a collection of models which are trained on monolingual or bilingual data and then combined in order to generate the translation. There exist multiple publicly available open-source MT tools: `Moses` (Koehn et al., 2007) is written in C++ and Perl, whereas `cdec` (Dyer et al., 2010) uses Python instead of Perl, and `Phrasal` (Green et al., 2014a) is implemented in Java. While there are differences between these tools, they all implement the following main components.

Alignment This is a preliminary stage which serves for identification of correspondences of words in the source and the target corpora. It is trained on a set of sentence-aligned parallel

texts (texts in the source language and their translations into the target language where every source sentence is associated with its translation). The training is usually unsupervised, since for the majority of language pairs there are no datasets labelled with correspondences between words.

One of the most popular methods for training of alignment models is the EM algoritm (Brown et al., 1993). The parameters (here — probabilities of target words being translations of source words) are initialised with uniform values (so that every source word is translated with any target word with the same probability), then the words in the training data are aligned according to this model — with the uniform probabilities this will mean that a source word is aligned to every word in the target sentence. Then the model is re-estimated from the alignment of the training data — this time it is not going to be uniform any more. The process is repeated multiple times. For the purposes of an MT system training the alignment is run twice: from the source to the target and from the target to the source, then the two obtained alignment models are combined in order to fill gaps.

Alignment is included into the majority of SMT systems, but there also exist stand-alone tools. The most well-known one is `GIZA++`[2] and its multi-threaded version `mgiza`[3]. `Fastalign` (Dyer et al., 2013) works faster than other alignment tools and, unlike others, allows alignment of new data with a pre-trained model.

Translation model Translation model is a core of MT system. It defines the possible translations of words and phrases and their probabilities. Lexical translations (translations of individual words) are extracted from alignments of a corpus and their probabilities are computed using maximum likelihood estimation principle:

$$p(f_i|e_j) = \frac{count(f_i, e_j)}{\sum_f count(f, e_j)}$$

where f_i and e_j are a target and a source words, respectively. Lexical probabilities are estimated for both source–target and target–source directions.

The generation of translation based on word probabilities is troublesome and can often result in bad translation. That happens because there usually is no one-to-one correspondence between words in the source and target sentences. Therefore, one needs to take care of cases when words are not translated or translated with more than one word, or when the original word order is not kept in the translation.

However, phrase-based translation models solve many of these issues. Phrase-level models provide translation probabilities for sequences of words instead of individual words.

[2]https://github.com/moses-smt/giza-pp
[3]https://github.com/moses-smt/mgiza

These sequences are not phrases in the linguistic sense, they are just chains of adjacent words from source and target sentences. Phrases are extracted from the alignments: a sequence of words from a source sentence can be translated with a sequence of words from a corresponding target sentence, if none of these source words is aligned to a word which does not belong to the target sequence. In other words, all the alignments between the source and the target should lie within the chosen sequences. Note that some words in a phrase can be unaligned — that solves the problem of words that are not translated or translated with a sequence of words. Probabilities of phrase translations are defined analogously to those of word translations.

Language model Language model (LM) ensures the fluency of translation, in other words, it rewards translations that are similar to the real texts in the target language. The goal of an LM is to compute the probability of a sentence **w**=$w_1, w_2, w_3, ..., w_n$:

$$p(\mathbf{w}) = p(w_1, w_2, w_3, ..., w_n)$$

The most popular type of LMs used are **ngram** LMs. They consider the probability of a sentence **w** as a Markov chain. First, they decompose the probability $p(\mathbf{w})$ into a multiplication of probabilities using the chain rule:

$$p(w_1, w_2, w_3, ..., w_n) = p(w_1)p(w_2|w_1)p(w3|w_1, w_2)...p(w_n|w_1, w_2, ..., w_{n-1})$$

and then assume that the probability of w_i given its history can be approximated as the probability of w_i given its history of *limited size*:

$$p(w_i|w_1, w_2, ..., w_{i-1}) \simeq p(w_i|w_{i-m}, w_{i-m+1}, ..., w_{i-1})$$

The quantity $m+1$ is called *order* of an LM: the length of ngrams which approximate the context of words. Then, a sentence is considered as a collection of its ngrams, and the task of an LM is to estimate the probabilities of these ngrams. In a conventional ngram LM the probabilities are estimated from a monolingual corpus using maximum likelihood estimation principle:

$$p(w_3|w_1, w_2) = \frac{count(w_1, w_2, w_3)}{\sum_w count(w_1, w_2, w)}$$

There also exist techniques for estimating probabilities of out-of-vocabulary words and ngrams that did not occur in the training data.

Such LMs have been criticised for the fact that the probabilities are estimated in the discrete space, which makes the estimation of missing ngrams very error-prone. There have recently appeared neural network language models that operate on continuous space and are able to mitigate this problem (Schwenk, 2013).

Discriminative training (Tuning) An SMT model contains a number of components, each dealing with a particular aspect of translation quality: a translation model suggests the most plausible translation of individual words or phrases in a sentence, a reordering model keeps track of the word order, a language model ensures that the resulting sentence is fluent. Ultimately, each of these components produces a score for each candidate translation and the best translation is defined as that which maximises model score which is the combination of all components:

$$f(x) = exp \sum_{i=1}^{N} \lambda_i h_i$$

where each $h_i(x)$ is a model component, or feature of the translation, and λ_i is its weight (importance).

The importance of the model components influences the translation quality to a great extent. The best combination of feature weights is defined on a held-out set of sentences (the development set). The development set is repeatedly translated with different combinations of feature weights and the translation compared to its reference translation in order to find the best weight. This stage of an MT system training is referred to as *tuning*. There is a number of algorithms used for tuning. Minimum Error Rate Training (MERT) based on the maximum entropy principle has been used as a baseline tuning technique (Och, 2003). Margin Infused Relaxed Algorithm (MIRA) algorithm maximises a margin between different training instances (Crammer and Singer, 2003). While it is initially an online algorithm, its batch version has been implemented in `Moses` (Hasler et al., 2011). Pairwise Ranking Optimisation (PRO) method ranks translation candidates based on their pairwise ranking (Hopkins and May, 2011).

A handful (10–15) of "baseline" features is typically used in SMT models. The usual `Moses` setup uses the following 14:

- 4 translation model features: direct and reverse translation probabilities for words and phrases,
- 6 reordering model features,
- distortion feature (probability of a phrase not changing its position in the sentence),
- language model score,

- word and phrase penalties.

However, the feature set is potentially unlimited: any new information about the source and/or target sentence can be considered during decoding in order to improve the translation. On the other hand, too many features could lead to overfitting of a system, especially when using MERT algorithm.

Decoding The previous stages described refer to MT system training. Besides training, there is a process of generation of translation itself, referred to as *decoding*. The aim of decoding is to find the best (most probable) translation. It is formulated as

$$\widehat{e} = argmax_e P(e|f),$$

where e is a source sentence and f is its translation.

The problem of decoding is the fact that there exist exponential number of options: a sentence can be segmented into phrases in a large number of ways, and for every of such segmentations there exist huge number of translations. Therefore, the best translation cannot be found via exhaustive search. There are some ways of reducing the search space.

The partial translations are referred to as *hypotheses*. A new hypothesis is generated from an existing hypothesis by extending it with a translation for a source phrase which has not been covered by this hypothesis. The decoding starts with an empty hypothesis. A commonly used search algorithm used in SMT decoding is called *beam search*. It consists in the following procedure. Every new hypothesis is evaluated with a function that uses a set of features (lexical probabilities, LM scores, etc.) and their weights defined at the *tuning* stage. This evaluation allows to define the most probable hypotheses and discard the rest. All hypotheses are stored in bins according to the number of phrases they contain. The size of search space is regulated by the maximum number of hypotheses in a bin.

Besides the evaluation of a hypothesis score, the algorithm computes the *future cost* each of them: the approximate difficulty of translating the part of the sentence which was not covered by the hypothesis. The combination of a hypothesis score and future cost create the ranking of hypotheses. The decoder outputs a hypothesis which covers the maximum number (preferably all of them) of source words and has the highest score.

Evaluation The evaluation of MT systems is performed as follows. A small held-out set of source sentences (test set) is decoded with the MT system, and the automatic translations compared with the reference translations for the test set. The comparison can address various aspects of text similarity, but the most wide-spread metrics measure lexical similarity of automatic and reference translations. The most wide-spread metrics are BLEU (Papineni

et al., 2002) which is based on lexical precision, METEOR (Banerjee and Lavie, 2005) based on the combination of lexical precision and recall, and an edit-distance metric TER (Snover et al., 2006).

2.2.2 Frameworks for feedback acquisition and incorporation

Although the collection of user feedback and its incorporation into an MT system do not need to be performed jointly, their integration can be beneficial. If user feedback is immediately available to the MT system, it can make adjustments in the real-time mode so that the user gets an improved version of the engine immediately after pointing out a particular error. This eliminates the problem of repeated MT errors which users often complain about.

There are several types of frameworks whose final goal is to improve the performance of translator, i.e., increase the speed of manual translation by supplying the translator with additional information. These can be built-in dictionaries, glossaries, spellcheckers. Moreover, such framework can provide a draft translation of a text so that instead of translating from scratch translator performs editing. The draft translations can come from a collection of previously translated sentences from which a system picks the one similar to the new sentence (such collections are called Translation Memories), alternatively, it can be generated by an MT system. In the latter case, the translator generates post-edits of the automatically translated sentences as a by-product of his/her work, and they can be used to improve the performance of the MT system in the online mode, so that the next translation is produced with a new version of model that takes into account the latest edits.

The most popular type of such frameworks is **Computer-Assisted Translation** (CAT). There currently exists a large number of CAT tools including commercial products like Trados[4] and open-source tools like `PET` (Aziz et al., 2012). One of the benefits of CAT tools is the fact that they are used by the majority of translators (Zaretskaya et al., 2015).

Another type of user-machine interaction is implemented in **interactive machine translation** (IMT) frameworks. Unlike CAT, it implements tighter integration of user actions into MT engine: here the re-training can occur multiple times within the sentence, because the automatic translation is re-generated every time a user edits it. The concept of interactive machine translation (IMT) was first suggested by Foster et al. (1997) and then actively developed under the scope of TransType (Langlais et al., 2000) and TransType2 (Esteban et al., 2004) projects. Later, IMT was enhanced with online re-training in CASMACAT

[4]http://www.translationzone.com/

project (Ortiz-Martínez and Casacuberta, 2014). There are also commercial solutions that provide interactive MT service, for example, Lilt[5] or Unbabel[6].

Despite the fact that IMT was designed as an improved, more handy version of CAT, case studies show that users often prefer conventional post-editing to interactive MT (Koehn and Haddow, 2009). On the quality side, productivity gains of post-editing are higher than those of IMT systems, but retuning to interactive feedback leads to larger improvements in quality (Green et al., 2014b).

2.2.3 Use of feedback with no retraining

Interactive machine translation

The IMT technology is based on the assumption that users generate the translation successively from the left to the right. Consequently, they are expected to check the MT system output in the same manner. Therefore, the user interaction with IMT system is expected to look as follows:

1. User is shown an automatic translation of a sentence.
2. If the first n words are translated correctly, user approves them. This prefix of n words is now fixed and will not be changed by the system.
3. User corrects the first erroneous word.
4. System re-generates a new automatic translation which keeps the user-approved prefix and takes into account changes s/he made.

In other words, an IMT system translates a sentence into the target language given its source *and* beginning of translation. While in SMT the search problem is defined as:

$$\widehat{e} = argmax_e P(e|f),$$

the incorporation of user-defined prefix of the sentence allows one to reformulate it as follows:

$$\widehat{e_s} = argmax_{e_s} P(e_s|e_p, f),$$

where e_p and e_s are the prefix (part of the sentence translated by user) and the suffix (the rest of the sentence), respectively. Note that $e = e_p + e_s$.

[5] https://lilt.com/
[6] https://unbabel.com/

Therefore, a conventional IMT system performs constraint decoding without changing the MT system itself. The early work on IMT concentrated on the improvement of the decoding procedure: Gandrabur and Foster (2003) improve the hypotheses selection by using confidence estimation of the hypotheses as an additional feature. Ueffing and Ney (2005a) improve the method by suggesting a number of confidence score-based heuristics. Vakil and Khadivi (2012) propose a new nearest-neighbour-based search algorithm.

Another source of improvement in IMT systems design is better understanding of user actions: Sanchis-Trilles et al. (2008) use information about mouse actions, Alabau et al. (2010) present an experiment on IMT system usability — here the IMT system is combined with a hand-writing text recognition tool, which is believed to make the work with productive and comfortable.

One of the main users' complaints about the conventional MT post-editing scenario is the need to correct the same MT errors multiple times (Macklovitch, 2006). This problem is particularly remarkable when translating texts with many repetitions, e.g. software manuals, patents, legal text. The reason for this problem is that the MT system underlying the IMT tool is not re-trained on user corrections: it only uses them to limit the search space. Therefore, the next step in the development of IMT systems was to actually incorporate the user corrections into the MT model.

Another problem that is more related to usability of IMT systems than to their performance is the fact that the user is forced to correct a sentence strictly from left to right. Such a strategy can be inefficient in cases when an automatic translation contains few errors and just needs minor editing: here, as soon as user changes something, the whole suffix of a sentence is re-decoded, and a new version can be different from the previous one and is not guaranteed to be correct, whereas the earlier prediction could be almost correct. The simulated experiments showed that this strategy is effective in terms of the automatic evaluation metrics (in other words, it led to faster increase in TER (Snover et al., 2006) or BLEU (Papineni et al., 2002) scores). However, these estimates did not take into account the congnitive effort the user needs to make every time when the suggestion changes. When presented to real users, an IMT failed to improve their productivity (Alabau et al., 2012). As a solution to this problem Marie and Max (2015) suggest an alternative approach: a user can mark the *correct* segments and the MT system re-decodes the rest. However, the absence of case studies does not allow one to make any conclusions on the efficiency of this approach, compared to traditional IMT.

Adaptive machine translation

There is another way of improving the MT output without changing the underlying models. Recently used n-grams and words (or phrases) translations can be stored in cache and the

final score for future models defined as a linear combination of n-grams from the underlying model and n-grams from the cache. This technique is referred to as adaptive modelling. It was successfully applied for improving language modelling in speech recognition, and to improve language modelling in SMT, particularly in the interactive environment (Nepveu et al., 2004). It turned out to be more effective when the test domain differs from the training domain. Furthermore, caching the translation model had almost no effect on the overall quality, while caching language model reduced the perplexity on unknown domains. Further research on using adaptive MT corroborates this result: the paradigm is useful for domain adaptation and shows much more improvement on language models than on translation models (Tiedemann, 2010).

Recent research in IMT also explores the usefulness of caching for interactive MT settings (Bertoldi et al., 2013). The CAT system combines caching and online adaptation techniques by filling the cache with all the phrase pairs from the source sentences and its post-edited translation. Therefore, the cache supports the existing correct translations and provides some new translation variants.

2.2.4 Retraining of models

Despite the success of bounded search procedures introduced under the scope of TransType and TransType2 projects and in adaptive MT systems, in many cases they are not sufficient to make use of user corrections, for example, in the cases of out-of-vocabulary (**OOV**) words or new translations for existing words. Therefore, many IMT systems use post-edits to dynamically change the underlying MT models.

Alignment

The first problem which arises when using the user-generated translation for the improvement of SMT systems is the need to align the new sentences. The full retraining takes too much time, so in order to include new data into the translation model, one has to employ approximated alignment techniques.

Hardt and Elming (2010) describe an approximated alignment of post-edited sentences: first each word from the post-edit is aligned with the source word on the same position, then the greedy search algorithm finds optimal local alignments. Blain et al. (2012) acquire the alignments between source and post-edit using the MT hypothesis as a pivot. The decoder returns alignments between the source and the hypothesis, then the alignment between the hypothesis and its post-edit are defined with TER edit distance metric (Snover et al., 2006). The source-to-post-edit alignment is deduced as a combination of these two alignments. As

an alternative to TER, Bojar (2011) uses the Longest Common Subsequence algorithm as implemented in Unix `diff` utility. The `Addicter` tool (Zeman et al., 2011) uses HMM-based word alignment with one restriction: the aligned words must have the same POS-tag. Cettolo et al. (2010) present another algorithm to extract parallel phrases from sentence pairs. It was initially designed for extracting parallel phrases from comparable texts, but can be used for a pair of sentences, as demonstrated in (Wäschle et al., 2013).

Word alignment can also be performed using pre-trained alignment models. Incremental alignments are available in `GIZA++` and `fastalign`. However, incremental alignment might not be the best strategy when the source text contains many OOV words: Farajian et al. (2014) point out that the alignment errors in incremental alignment mostly affect the alignments of unknown words, which are a common error type in MT. They propose an extension of the alignment process based on the strategy presented by Esplà-Gomis et al. (2012): there, the unknown words are aligned without a pre-trained alignment model: the word correspondences are defined based on their frequency in external bilingual sources such as phrase tables of third-party MT systems.

Translation models retraining

IMT systems aim at using human feedback directly after its acquisition so that users can see the improvements once they corrected the MT errors. However, this scenario is infeasible in a traditional SMT setting, because there model retraining takes too much time and cannot be performed after every correction.

Therefore, research on IMT systems retraining focuses on online algorithms. Ortiz-Martínez et al. (2010) suggest an online learning algorithm for all components of the MT system. They use incremental EM algorithm (Neal and Hinton, 1999) to generate the alignments of the new sentences and modify all the models so that their parameters can be updated incrementally after every correction. The test set is automatically translated with a baseline model and then manually corrected, the corrections are used to change the model parameters. This system performs better than the static baseline system.

Another way to incorporate post-edits is to create "local" translation and language models and combine them with the models of the baseline system (Hardt and Elming, 2010). The local models are combined with main models via log-linear combination of weights defined by minimum error rate training (Koehn and Schroeder, 2007). Foster and Kuhn (2007) suggest measuring the lexical distances between the baseline dataset and the new data via TF-IDF, LSA or other metric and then assigning combination weights based on these weights. Bisazza et al. (2011) propose the filling-up technique, which is the reinforcement of the main phrase table with the phrases from the local phrase table, possibly with different weights.

The experiments show that if post-edits come from the same domain, but slightly different topic, the use of such post-edits has almost no result (Cettolo et al., 2014). Conversely, translation quality improves when the system is re-trained on post-edited sentences from the same document. Thus, this technique is particularly effective in project adaptation scenario — as a user edits sentences of a document, the MT system adapts to this particular document. The research under the scope of **Matecat** project also demonstrates the strong effect of user corrections at the document level. Wäschle et al. (2013) perform interactive update of the MT system by mixing its global translation and language models with local ones whose phrases and n-grams are extracted from the post-edits of sentences from the same document.

The similar approach is used in the PEPr model (Simard and Foster, 2013). Although it is used for automatic post-editing, it has much in common with other research on incremental retraining. Simard and Foster (2013) construct a phrase table of manually post-edited sentences and rescore it after adding every new entry. The main difference is that the phrase table is filled with the phrases from the original automatic translations along with the user corrections, so that the new variants compete with the existing ones.

The `Moses` SMT toolkit (Koehn et al., 2007) has an incremental retraining module based on stream-based translation models described by Levenberg et al. (2010). The new phrases are stored in a dynamic suffix array analogously to the original description of suffix array for SMT given by Lopez (2008), and the phrase scores are computed on the fly during decoding. This technique solves the problem of time at the cost of quality: the incremental system performs worse than the batch system (see for example a work by Mathur et al. (2013), where batch baseline is better than incremental baseline trained with incremental `GIZA++`). A similar approach is implemented in the `cdec` translation toolkit: Denkowski et al. (2014b) describe an extension of `cdec` for dynamic model adaptation. There, the baseline data is stored in a suffix array, and while the user post-edits automatic translations, their corrected versions are added to a lookup table. At decoding time a sentence-specific translation model is estimated based on the joint static (baseline) and dynamic (post-edited) data, and the translation rules approved by user are rewarded. Another approach to dynamic adaptation in `Moses` implements the dynamic cache-based phrase tables described by Bertoldi et al. (2013). They are similar to cache-based models described in section 2.2.3, but the cache is filled with phrases extracted from newly acquired post-edits.

Language models retraining

Unlike translation models, language models can be successfully adapted without human intervention in real-time. A crucial factor of a language model's quality is the size of training data, which can be increased without corrections from users. Another factor is the quality

of the training set: a large amount of out-of-domain data may have no positive effect on the quality of the language model. However, the selection of suitable data does not require human post-editing of MT output either. Moore and Lewis (2010) describe a selection technique based on a small seed language model of good quality. This idea was implemented in XenC tool (Rousseau, 2013).

Nevertheless, the post-edits are also used in some of the above-mentioned approaches to update language models. Ortiz-Martínez et al. (2010) update the LM by increasing ngram counts for ngrams that occur in post-edits. Wäschle et al. (2013) use the post-edited data to create a new LM which considers only ngrams with content words.

Retuning the machine translation system

Changes in the components of an MT system require retraining of their weights, i.e. retuning. The tuning in SMT is traditionally performed via minimum error rate training (Och, 2003), which is a batch learning procedure. The incremental update of weights demands online learning methods.

The research by Martínez-Gómez et al. (2012) explores the applicability of four online machine learning methods (passive-aggressive algorithm, discriminative ridge regression, perceptron, and Bayesian predictive adaptation) to the task of tuning weights of MT system models. According to this research, ridge regression leads to good results on the task of weights update. Following this work, López-Salcedo et al. (2012) conduct experiments on updating the feature weights using discriminative ridge regression as an optimization technique. However, the method fails to improve the translation quality.

Wäschle et al. (2013) extract sparse features from post-edits. Sparse features are lexicalised features, i.e. pairs of source and target words/phrases directly used as features. Such pairs are extracted from the stream of post-edits, and their weights are retuned incrementally using structured perceptron algorithm (Collins, 2002).

Mathur et al. (2013) suggest two methods for tuning MT systems using small amounts of post-edited data. The first method is based on sparse features for every phrase in the sentences of N-best lists. The phrases from the hypotheses which are closer to the post-edit are rewarded, other phrases are demoted. The update is performed with the perceptron algorithm. The second method re-tunes the traditionally used features of an SMT system. The feature values are adjusted by the MIRA, which is also an online algorithm. Here, even the individual re-tuning algorithms give significant improvement when applied to systems trained on highly repetitive corpus, and the combination of both techniques gives an improvement of over 6 BLEU points. Conversely, the experiments with corpora with lower repetition rate give some improvements only over an incremental baseline (i.e. the word alignments

were learned via incremental GIZA++ to allow the use of test sentences as new data), and no improvement compared to a batch baseline.

Re-tuning of an SMT system can be based on other types of feedback. Saluja et al. (2012) present a method for re-tuning the weights of an SMT system using binary feedback about the quality of sentences. This human-in-the-loop scenario is potentially more effective than post-editing or IMT because indicating whether a sentence is good or bad is much faster than actually correcting it. The system described follows the method of structured output learning with binary-labelled data originally proposed by Chang et al. (2010). It uses a modification of SVM method — the latent structural SVM that handles hidden variables (Yu and Joachims, 2009). The target function is expanded with a summand which takes into account the sentence's rating (1 or 0). The method is implemented and embedded into the cdec toolkit. The experiments presented in the paper show good results for the re-tuning with binary labels. However, the strategy was tested on only one dataset, namely the Chinese-English BTEC corpus (Paul, 2009). The dataset is small in size (182,000 sentences) and contains mostly quite short sentences with 4 references for each source. This corpus is different to the data that is usually post-edited, so there is no information on the performance of the method in a real-world task.

Summary

Many researchers propose various ways of using post-edits and occasionally other types of human feedback in interactive settings. However, many of them target at only individual aspects of the post-edits incorporation or present work in progress, e.g. describe the approach to post-edits collection and extraction of alignments, but do not suggest any method for their incorporation into the MT system. The end-to-end approaches that tackle all elements of an MT model are not numerous: Denkowski et al. (2014a) presents an extension of cdec that fetches post-edits and updates all components of an MT system. The Thot toolkit by Ortiz-Martínez and Casacuberta (2014) combines the IMT technique (section 2.2.3) with dynamic adaptation of all MT components (Ortiz-Martínez et al., 2010). Another end product is the MateCat system[7] which assembles the results of research by multiple groups.

Overall, the methods described bring substantial improvements in translation quality that reach 1 to 6 BLEU points for different datasets. However, despite these advances one cannot conclude that the methods presented are unconditionally effective. Some of the methods were tested on very specific data: e.g the experiments on weights tuning on binary user feedback suggested by Saluja et al. (2012) are reported only for BTEC (Basic Travel Expressions Corpus). There are some other issues related to the data chosen for the experiments. It has

[7] https://www.matecat.com/

been noticed that the dynamic adaptation of MT components is more effective when there is a domain shift between the baseline MT data and the new data which is being translated and post-edited. For example, in (Hardt and Elming, 2010) and (Wäschle et al., 2013), methods proposed performed well when translating patents but much less so when applied to other more heterogeneous and less strict domains. This implies the high specificity of updates: they can only correct highly repetitive errors which occur either in highly repetitive datasets or in a local contexts (e.g. incorrect translations of a specific term often repeat within a document but much less often in different documents, even if they belong to the same domain). Although Bertoldi et al. (2013) use this fact to perform fine-grained tuning of an IMT system to a particular document and user, in many cases this feature is considered as a limitation.

Finally, it is difficult to tell if any of these methods prove effective in real-world tasks: in the majority of cases researchers do not have the possibility to perform real user evaluation and report results of simulated experiments. As was already noticed for IMT, simulated and real effectiveness are very different, and the user perception can be an issue: if users do not like a scenario suggested by a tool, it will not bring any improvement.

2.2.5 Active learning

One of the most efficient ways to improve the quality of a machine translation system is to enhance it with new training data. In the usual scenario the amount of monolingual data in the source and target languages is potentially unlimited. However, as parallel data has to be created by having humans translating monolingual content — an expensive process — it is often reasonable to select the sentences to translate using some strategy which chooses the most useful sentences from a large set of monolingual data in the source language. This will result in a system of higher quality using less training data. This technique is usually referred to as active learning (AL) (Settles, 2012) and has been extensively used in various Natural Language Processing (NLP) tasks such as POS-tagging (Ringger et al., 2007), parsing (Reichart and Rappoport, 2007), sentiment analysis (Xiao and Guo, 2013).

There are various applications of active learning to MT. The methods of measuring sentence usefulness include TF-IDF based metrics (Eck et al., 2005), metrics of informativeness relying on unseen n-grams (Ambati et al., 2010; Bloodgood and Callison-Burch, 2010) or vocabulary coverage (Lewis and Lansing, 2013), metrics that use difference in perplexities of sentences under in-domain and out-of-domain LMs (when active learning is performed to adapt an MT system to a new domain) (Axelrod et al., 2011; Banerjee et al., 2013). However, these approaches are beyond the scope of our research because they do not rely on user feedback to select the data.

Haffari et al. (2009) propose a method that differs from the previously mentioned ones. They train a classifier which uses a number of features such as the amount of unseen phrases/n-grams, the similarity to the existing training data, and the model score for translations. This method requires post-edited data for training, so it can also be considered as a way of indirect incorporation of user feedback into MT. This approach is further developed by Ananthakrishnan et al. (2010). They describe an error-driven method to define the most useful sentences. The idea is that the most useful sentences are those that lead to the biggest number of translation errors. They train a classifier that induces n-grams which are translated incorrectly. The classifier is then used to pick the source sentences that are likely to cause errors, i.e. contain the largest number of error-carrying n-grams. Du et al. (2015) use confidence score as one of sentence usefulness criteria, but the authors use the confidence of an MT system itself (model score), and not a score defined based on external criteria.

The IMT paradigm has also used active learning settings. González-Rubio et al. (2010) developed a system which combines IMT with active learning: the system translates the input text, but unlike the ordinary IMT systems, it does not ask the user to correct all sentences, but only those which do not score highly enough. The score of each sentence in this setting is defined by the translation model alone: it is a combination of lexical probabilities for all the words in a sentence (either mean score or the number of words whose lexical probability exceeds some pre-defined threshold). The authors conduct experiments with varying thresholds to define the optimal ratio between user effort and translation quality. In the work of González-Rubio et al. (2012) this strategy is expanded by comparing several methods of scoring translated sentences: confidence estimation based on lexical probabilities, random sampling, and n-gram coverage estimation. The experiments show that:

- the confidence estimation and random sampling have similar results with confidence estimation slightly outperforming,
- n-gram coverage estimation performs even worse than the random sampling,
- confidence estimation with subsequent incremental retraining of SMT models (as reported by Ortiz-Martínez et al. (2010)) performs closely to confidence estimation without retraining (i.e. dynamically changing the underlying SMT system has little effect on translation quality and user effort).

In chapter 6 we present our efforts towards AL for MT improvement. We use predictions done by a QE model in order to identify the most useful data instances for an MT system.

2.2.6 Automatic post-editing

One of the possible uses of post-edits is to learn some patterns from the corrected errors and use them to edit new automatically translated sentences. Earlier work reports experiments on error corrections with no direct human supervision. Instead, some heuristics are used: Knight and Chander (1994) correct the determiners in English texts using a set of rules, Povlsen and Bech (2001) perform automatic rule-based reordering to reduce the cost of further manual post-editing. In later work, post-processing rules are extracted from post-edits. Groves and Schmidtke (2009) extract editing patterns from post-edited translations. The most frequent corrections for English-German and English-French translation are addition or deletion of an article or a preposition. However, the research does not suggest any ways in which to use these patterns.

Statistical post-editing (**SPE**) is an alternative approach to automatic post-processing of machine-tranlated text (Simard et al., 2007). An SPE system is a system that simulates the work of human post-editors: it is an SMT system trained on a parallel corpus where the source side is machine-translated text and the target-side is its manual post-edit or a reference translation. This technology has been proved effective for improving the output of rule-based MT systems (Dugast et al., 2007), but showed little effect on phrase-based MT output (Potet et al., 2012a). However, the addition of source information to an SPE model, as it was done by Bechara et al. (2011), can bring some improvement.

SPE may also be used in domain adaptation, as was demonstrated by Rubino et al. (2012). However, the research by Potet et al. (2012a) shows that other domain adaptation methods outperform the SPE significantly. There, SPE as a domain-adaptation technique is compared with corpus-based and model-based methods. The corpus-based method consists in the enhancement of the training corpus with a domain-specific corpus and training of a joint model. The domain-specific corpus can be duplicated several times to gain more weight (the best result is achieved with domain-specific corpus repeated 1,000 times). Model-based method resorts to training of a separate phrase table and a language model for the domain-specific corpus and either joint decoding or merging of models for the two training datasets. The latter technique is more effective, however, both outperform SPE in terms of BLEU and TER scores by large margin.

The latest advances of MT, namely the neural MT models, influenced the APE field as well. Analogously to earlier works, the latest APE models are also trained as MT models where automatic translations serve as the source and their manual post-edits as the target. However, when based on a neural translation model with attention (Bahdanau et al., 2014) an APE model is more successful: the systems by Junczys-Dowmunt and Grundkiewicz (2016) and Libovický et al. (2016), which are based on a neural MT model both outperform the

baseline APE system (Simard et al., 2007) at the WMT-16 APE task (and the former system scores first in the task).

On the other hand, it has been shown lately, that phrase-based SMT can also show improved results upon the performance of early SPE models: Chatterjee et al. (2016) build on SPE models (Bechara et al., 2011; Simard et al., 2007) with addition of linguistic information, neural LMs and predictions made by a Quality Estimation system. Likewise, Pal et al. (2016) use phrase-based SMT, but extend it with Operation Sequence Model (Durrani et al., 2013), and represent the training data (automatic translation paired with its post-edit) as a sequence of text editing operations. These models were also able to outperform the baseline.

However, although the latest APE models are more successful than their predecessors, these achievements cannot be attributed solely to the use of more suitable methods. The latest APE models presented at the WMT-16 APE shared task were trained on the data distributed for the task. A notable difference of this dataset from the previous ones is its domain: it contains IT texts, whereas the previous models were mainly trained on news texts. The IT domain is more homogeneous than the news domain, so MT errors and corrections are more repetitive, which contributed to the success of APE models (see the findings of WMT-16 (Bojar et al., 2016) for an analysis of the APE task).

2.2.7 The use of quality estimation for the improvement of MT

Quality estimation (QE) is another MT-related task that uses post-edits and other forms of human feedback. QE is a technique that allows to evaluate the quality of a machine-translated segment without comparing it to a reference translation. Instead of that, QE systems use sets of various features of the source sentence, its automatic translation, and the MT system. On the one hand, QE is performed *after* the sentence is translated, so its applications go beyond the scope of MT: QE can help an end user of an MT system to understand how accurate a translated sentence is and decide if it is good enough for the user's purposes. On the other hand, QE can be considered as a way of propagating the user feedback onto the unseen data. If a QE system is trained on a collection of machine-translated sentences which are manually tagged with quality score or perceived post-editing effort (for example, from 5 — "perfect translation"/"no post-editing needed" to 1 — "unintelligible translation"/"needs to be fully rewritten"), it can define the same scores for unseen automatic translations. If this QE system is good enough, its predictions can be used in all tasks in analogous ways to human feedback.

This application was actually an original goal of QE: the first QE systems casted quality of translation as confidence of an MT system about this translation, and their purpose was to choose the best translation from an N-best list, i.e., the translation that the system is more confident about (Gandrabur and Foster, 2003). After the early work on QE, this reranking

task was re-established only recently: Luong et al. (2014d) perform N-best list re-ranking using word-level quality labels. Here, the authors define the sentence-level quality score as the ratio of correct words in a sentence, the ratio of correct translations of different parts of speech, the ngram precision according to word labels. Then the features are weighted on held-out N-best lists using the MERT and MIRA tuning algorithms. Despite the classification errors, word labels are effective in improving MT quality.

A more effective way to improve MT quality using word-level labels is lattice rescoring: instead of approving or rejecting the whole sentences, one chooses the best target segments from the translation lattice and combines them. The experiments by Luong et al. (2014a) show that re-weighing the translation lattice with QE scores improves the MT output, and this improvement is larger than the one achieved by reranking the N-best list. Moreover, the authors show that there is space for further improvement in translation quality: when oracle quality scores are used instead of word-level QE system predictions, the BLEU score goes up by a large margin. This shows that better QE models are needed.

Our work along these directions is presented in chapter 7. We incorporate QE predictions at the stages of N-best list reranking, tuning and decoding.

2.3 Conclusions

Post-edits are the only type of user feedback on MT system output which combines informativeness and relative simplicity of acquisition. Therefore, the possibilities of its use have lately been widely explored. The direct use of post-edits as training data is possible and usually has better effect than adding traditional training data (i.e. human translations created independently from the automatic translations), yet it requires large amounts of post-edits which are in most cases unavailable. Crowdsourcing can help collect the desired amount of data within reasonable time and budget, however, this data is likely to be of low quality.

The small amounts of post-edits available can improve the translation quality via incremental retraining methods in cases when the domain of initial training data differs from the domain of test data, i.e. in domain adaptation scenarios. Constructing additional phrase tables or augmenting the existing phrase tables with new data both have effect only in the cases of domain shift, moreover, this effect is more distinct if the target domain is homogeneous, has high repetition rate and does not have many sentences with irregular structure. In cases of heterogeneous domains (e.g. the proceedings of European Parliament, subtitles) the incremental adaptation methods have no effect on the quality. What is more, even in the case of homogeneous domain the effect of incremental adaptation can be seen mainly in the local scope: that is, an automatically translated sentence is more likely to benefit from the

post-editing of sentences from the same text than from any corrections of other texts in the same domain. It was shown that while updates locally improve quality and/or post-editor's experience, they cannot bring any noticeable difference at larger scale.

These results stem from the lack of data. Many papers in the area of incorporation of human feedback into MT report results of simulated experiments where human post-edits are replaced with reference translations (a description of such a simulated experimental setting can be found in the work by Denkowski et al. (2014a)). While these works often concluded that this method can work effectively with larger MT models if human feedback datasets of much larger size become available, but this is not the case: the post-editing or other way of labelling automatic translations for quality is still a time-consuming task. It requires the work of professional translators, which also makes it expensive. Ways of reducing the time and cost, such as crowdsourcing, proved to be ineffective. Therefore, quality-labelled datasets are unlikely to become larger. Currently the largest existing corpus of post-edits (Autodesk) contains 400,000 sentences for one of language pairs (which is not a lot considering the low length of sentences in this dataset), and it is an order of magnitude smaller than the size of parallel corpora used for training of MT systems.

Overall, existing human feedback datasets are too small to bring any significant improvement outside the scenario of domain adaptation in the local context, and collecting larger amounts of feedback is too expensive. An alternative is to generate pseudo-human feedback in an automatic manner, namely Quality Estimation of Machine Translation. QE provides quality scores for translation segments — i.e. it replaces human evaluation of translation quality. It can be trained on a small amount of human-labelled data, so existing datasets are large enough for it. The quality estimate produced by such a model is of course less reliable than human feedback, but unlike expert evaluation, automatic quality estimation is fast and cheap. Although it does not repair incorrect translations, it can indicate their flaws — and this information is already helpful and can be used in multiple ways. In the next chapter we review the main approaches to QE: its background, the datasets used for QE models training, the techniques which were attempted, the main flaws and limitations of the state-of-the-art QE models.

Chapter 3

Quality Estimation

Quality estimation (QE) of MT is a task of determining the quality of an automatically translated segment without consulting any oracle translation. This constraint makes it different from the automatic evaluation of MT, which determines the quality of an automatically translated text by comparing it with a free reference translation. QE has been briefly discussed in chapter 2 which was dedicated to various ways of improving the quality of MT using human feedback. It was noted that QE can be considered as a form of indirect human feedback: it is trained on actual human judgements about translations and its aim is to simulate the human labelling of the automatic translations.

Our survey in chapter 2 has led us to the conclusion that the existing human feedback is inherently scarce: the tasks of post-editing, error labelling or even sentence ranking (ranking of alternative translations of one sentence from best to worst or pairwise comparison of alternative translations) are laborious and time-consuming, so collecting any substantial amount of such data is infeasible. On the other hand, by using a QE system we can label any number of sentences with pseudo-human feedback.

In our work we decided to concentrate on the use of QE as a form of human feedback for the improvement of MT quality. Therefore, we need to review the main approaches to QE in order to understand which the best approaches are and what the capabilities of the state-of-the-art models are. The quality of MT can be estimated at different levels of granularity: at the level of words, phrases, sentences or entire documents. However, we would like to incorporate QE into MT models, so we should also take into account the constraints of MT. Since the vast majority of the existing MT systems operate at the sentence level, the estimation of quality at the document level will be useless for the improvement of MT quality, since MT systems will not be able to take into account document-level information. Therefore, we will concentrate on other levels of granularity: sentences, phrases and words.

3.1 Background: Confidence Estimation

The task of quality estimation is not unique to MT — the estimation of a system's confidence in order to identify potentially erroneous output has been applied to other NLP tasks like speech recognition (Seigel and Woodland, 2011) and information extraction (Culotta and McCallum, 2004). In general, quality estimation makes sense for any application whose output is used in a pipeline without human supervision, i.e. where this output is the input for the next system or is returned to an end user directly (e.g. MT for gisting purposes). In such cases every stage of the automatic processing can contain errors, so quality control is important. An example of such a pipeline involving machine translation is cross-lingual information retrieval.

The first works in QE for MT were inspired by the body of research in confidence estimation (CE) for speech recognition. The QE task was initially formulated as the determination of how confident an MT system is about the produced translation. The developers of first CE models tried to transfer the previous experience in the CE for automatic speech recognition systems to MT. The CE models for MT also aimed at determining how confident an MT system is about the translation. Such models usually exploited internal properties of an MT system, like translation probabilities, language model scores, statistics computed from lists of N-best translations.

However, there is a notable difference between CE and QE. The latter has a broader formulation: it aims at determining the quality of an arbitrary translation as opposed to the confidence of a particular MT system in its own translations, and the notion of quality is based on external criteria, e.g. intelligibility or post-editing effort. Moreover, it assumes that translation quality that can be degraded not only by poor estimates of translation probabilities, but also by a malformed source sentence. Thus, QE and CE systems differ in the types of target values that they predict. As follows from the name, CE estimates the confidence of an MT system's output (automatically translated sentence), i.e. the probability of the output. On the other hand, QE is more versatile, and its target label depends on the notion of quality which can be defined as intelligibility of translation (Quirk, 2004), perceived post-editing effort (Specia et al., 2009), post-editing effort computed from the amount of editing performed or time spent on a segment correction (Bojar et al., 2013), etc.

This contrast is also reflected in the feature sets used by the two classes of approaches: while CE mostly uses internal MT system information, QE aims at determining the quality of the output in a system-independent way. Therefore, QE systems can produce estimates even when internal information is unavailable. They can also directly compare the outputs

coming from systems with different architecture (e.g. statistical and rule-based MT systems). They can in principle even evaluate the quality of translations done by humans.

One of the first descriptions of a CE system is provided by Gandrabur and Foster (2003), where it is used to improve the quality of translations performed by an IMT system. The task of CE system in this application is to provide more accurate probability estimates for translation suggestions than the probabilities returned by the original translation system. Therefore, CE is used here for the reordering of the list of N-best suggestions and is evaluated via the performance of the IMT system. The probabilities are estimated for each generated suggestion, the CE is performed for ngrams of length 1 to 4, although some of the features used are sentence-level. The authors use three groups of features:

- features capturing the intrinsic difficulty of the source sentence: ngram probability and perplexity, word length and number of words;
- features defining how hard the source sentence is to translate: source-target alignment statistics for the sentence, words translation probability, number of unknown words;
- features defining how hard the source sentence is to translate for a particular MT system. This group of features contains translation model statistics: number of hypotheses and rank of the current hypothesis, N-best list and lattice size, model probability of a current hypothesis.

The training is performed with single-layer or multi-layer perceptron. The results differ depending on the translation model used: better-performing MT models gain less from the CE system.

The work by Ueffing et al. (2003) describes the use of word posterior probabilities (WPP) as confidence score at the word level. They had been used for CE in speech recognition, but this work was the first to introduce them in CE for MT. Word posterior probability is the probability of a target word appearing in the translation. It is computed as a sum of posterior probabilities of sentences containing the target word.

The idea is further developed in work of Zens and Ney (2006), who introduce ngram posterior probabilities (NPP). They are computed analogously to WPPs. The NPPs are used as confidence scores to reorder the N-best lists. The experiments show that the use of single words posterior probabilities does not improve BLEU, but higher order ngrams achieve improvements.

The experience of early work on CE for MT is summarised in a report of a workshop held in Johns Hopkins University (Blatz et al., 2004). The workshop participants conducted experiments on sentence-level as well as subsentence-level CE.

The data preparation revealed some problems with sentence-level QE: since the performance of MT systems was quite bad, a very small proportion of the automatically translated

sentences could be tagged as "good", so a CE model had no or very little positive data. Therefore, the authors relaxed the quality requirement and instead of classifying sentences as good or bad define if an MT system that produced them was better or worse than a given threshold, which was set so that 5% or 30% sentences are correct.

The training and test datasets for both sentence- and word-level CE were generated automatically by comparing machine translations with their references using automatic evaluation metrics. For sentences the authors used NIST and WER (Word Error Rate — percentage of correct words in a sentence) scores, for the word level the word was tagged as correct if it occurred in the reference.

The sentence-level feature set consists of 91 features that belong to the following groups:

- features used by the MT system to produce the output (i.e. decoder features): log-probability of translation model, lexical log-probability, word and alignment template penalty (analogous to phrase penalty in phrase-based SMT), LM log-probability, number of non-monotonous alignments.
- features extracted from MT system information: N-best list features, hypotheses, pruning and alignments statistics,
- features that do not depend on the MT system: ngram frequencies, sentence lengths, probabilities of sentences under external LMs. This group can be divided into three subsets depending on what information they need:
  - target features,
  - source features,
  - source-target features.

Word-level CE was formulated as a binary classification task, sentence-level tacked as binary classification as well as direct estimation of probability of a translation (regression). The classification was performed with a Naive Bayes classifier and multi-layer perceptron (MLP), the regression model was trained with multi-layer perceptron. The experiments showed that Naive Bayes is inferior to MLP at both sentence and word levels because many of the used features violate the independence assumption which is essential for the Naive Bayes model.

The work also reported feature selection. The best sentence-level feature subset is the one containing decoder features. The most informative word-level feature is word posterior probability. However, for both sentence- and word-level tasks the full feature sets achieve higher quality than any of their subsets.

One of the main drawbacks of the experiments presented by Blatz et al. (2004) is the lack of human annotation of the data: the quality of translation is given by automatic evaluation

metric scores. Despite being a widely-used approximation of translation quality at the corpus level, automatic metrics do not correlate well with human judgements at the sentence level. Quirk (2004) shows that a CE system trained on a small human-labelled dataset outperforms the one trained on a much larger (350 vs 15,000 sentences) set of automatically labelled examples. Later QE systems have been trained on human data generated via ranking of sentences or scoring post-edits.

3.2 Sentence-level Quality Estimation

As was mentioned before, the first approaches to QE were targeted at improving the MT output by identifying segments with low confidence. Since the majority of decoders take individual sentences as input, it seemed more logical to identify subsentential units (words or ngrams) which have low confidence and try to fix them. In such decoders the confidence (or quality) of the whole sentence is less useful than word-level confidence, because it does not allow one to guide the decoder.

The majority of applications of sentence-level QE lie beyond machine translation itself — they are related to applications or end-users who need the automatically translated texts:

- Online MT engines — QE system is needed to inform users of the overall quality of a sentence (e.g. how reliable the translation is)
- CAT systems which use automatic translations as a draft for post-editing: a QE model on top of the MT system can filter out sentences which are too bad to be post-edited, or sentences which are already good enough and do not need any editing.

However, research on sentence-level QE is much more prominent than on word-level QE. That might be related to the sentence-level nature of many MT models: they process a sentence as a whole and consider the sentence-level context. Likewise, an online MT engine can ask users if a translation of a particular sentence was good, thus gathering sentence-level user feedback, while determining word-level quality the same way is more difficult. Another difference between sentence-level and word-level tasks is the fact that the majority of sentence-level features are more generalisable: they tend to be numerical rather than symbolic, therefore, a well-performing system can be trained with comparatively small amount of data (2,000 to 3,000 sentences) which might be insufficient for a word-level QE system.

In this subsection we review the existing approaches to the estimation of quality of MT at the sentence level.

3.2.1 Labels and evaluation

One of the first questions that arises in relation to sentence-level QE is the choice of appropriate labels for sentences. What should be used as a sentence-level score when evaluating the quality of automatic translations? An obvious solution is to use **binary labels** discriminating between correct and incorrect sentences. However, sentences can be quite long and heterogeneous, so in the majority of cases it is difficult to rate it as "bad" or "good". Instead, more fine-grained scales (from 1 to 4 or 1 to 5) have been used, evaluating the overall sentence quality or one of its aspects (e.g. fluency, accuracy, suitability for post-editing or gisting).

A question that arises is how to acquire the quality scores. The JHU's CE Workshop showed that automatic evaluation metrics for MT do not suit for labelling of a dataset for a QE system training. One of the reasons is that many of popular evaluation metrics (e.g. BLEU) perform well enough at the corpus level only. The sentence-level approximation of BLEU — sBLEU — adds 1 to every sentence-level ngram count in order to avoid zero counts which turn the whole value to zero. This approach still results in unreliable scores. Another possible reason of unsuitability of automatic metrics for CE data labelling is that these metrics were designed to compare the performance of different MT systems on the same test set, whereas here they were used to compare the quality of translation of different sentences from one MT system. As it was shown by Quirk (2004), training on manually labelled data allows to achieve much higher accuracy of quality predictions than training on much larger sets of sentences labelled with automatic evaluation metrics.

However, finding reliable quality scores is a difficult task. For example, the QE data prepared for WMT-12 (Callison-Burch et al., 2012) contains perceived **post-editing effort scores in 1 to 5 scale** (from 1 – unintelligible translation to 5 – perfect or almost perfect translation) from 3 annotators. Despite the fact that post-edited outputs were available for consultation, 168 sentences (8%) got conflicting scores from the 3 post-editors — the difference of scores for these sentences was 2 points or more. This shows that probably the scores should be based not only on human perception, but on a more objective criterion.

A more objective metric obtained as a result of post-editing is the error count normalised by the sentence length. A metric that computes this score is **HTER**, which stands for Human Translation Error (Edit) Rate and computes the number of edits which a post-editor makes when correcting the automatically translated sentence (Snover et al., 2006). Another similar example of an objective label is post-editing time (Bojar et al., 2013). A weakness of post-editing time is its high variation across editors and large number of outliers that do not reflect the real quality of sentences. Therefore, HTER proved a more convenient label. On the other hand, it is important to note that post-editing effort does not account for "quality" in general, e.g. the research by Denkowski and Lavie (2012) shows that human quality labels have low

correlation with HTER scores. However, HTER is convenient and easy to obtain, so it has been used as a sentence-level label at QE shared task at WMT since 2014.

Another aspect which is important for any prediction model is the evaluation of its results. The first work on QE used the **ROC curves** to evaluate the performance of classifiers. A ROC curve is a plot of the number of true positives against true negatives for a range of different thresholds separating positive instances from negative ones. Its shape will be close to top and right borders of the graph for a well-performing classifier and a straight line from point (0, 1) to (1, 0) for a classifier that assigns scores randomly. In order to be able to compare the ROC curves analytically they use the IROC metric which is the area under the ROC curve (Blatz et al., 2004). ROC curves cannot be used to evaluate certain QE models: it requires the model be probabilistic, i.e. to output a real number and convert it to the target value ("good"/"bad") using a threshold. This evaluation technique is not suitable for non-stochastic classifiers nor for regression models.

Regression models are evaluated using Mean Absolute Error (**MAE**) and Root Mean Square Error (**RMSE**) scores, which are defined as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

The multi-class classification can also be evaluated with these metrics, because we can define the relationship between classes (i.e. we can explicitly compute the error when a sentence of class "3" was assigned a class "2"). However, this approach has recently received criticism in the work by Graham (2015). There, some drawbacks of MAE score are discussed: it essentially measures the difference between the real and the predicted score and in many cases can be minimised by always predicting the average score for the training set. As an alternative the author suggests using **Pearson correlation** as an evaluation metric for sentence-level QE systems. This has been a main metric in the two last editions of the WMT QE shared task.

3.2.2 Features

Blatz et al. (2004) introduced a large number of features that can be used in a sentence-level QE system. However, further work investigated the usefulness of some features that were absent in the CE system described by Blatz et al. (2004). Gamon et al. (2005a) use linguistic features such as grammar productions and semantic relationships combined with language

model perplexities. Albrecht and Hwa (2007) use pseudo-reference features, which are features extracted from alternative automatic translations of the source sentence produced by third-party MT systems.

Another research direction is to narrow down the feature set. The work by Specia et al. (2009) demonstrates the benefits of small feature sets. They extract a large set of features consisting of those used by Blatz et al. (2004) and some new features that provide information about alignments between the source and target sentences and detect mismatches between them (e.g. in the amount of numbers, punctuation, etc.). They train QE models on a number of datasets and select subsets of best-performing features. The experiments show that small subsets of features often perform better than the full set, and that features that do not depend on the internal MT system information can be sufficiently expressive and encode most information about the quality of translations.

A common approach to feature selection is to rate features somehow according to their relevance on a development set and then find the optimal subset of the most relevant features. Specia et al. (2009) get such a ranking as a by-product of the training process. Similarly, Shah et al. (2013) sort the features by their importance using Gaussian Processes. Optimal feature sets vary for different corpora, but some features were shown to perform well across different datasets.

These experiments formed a basis for a baseline sentence-level QE system that was established in the first edition of QE shared task (Callison-Burch et al., 2012). The system was trained with `QuEst` toolkit (Specia et al., 2013). It used 17 black-box features which perform well for different datasets, do not depend on internal MT system information and are easy to extract. These features are:

- lengths of source and target sentences,
- average source token length,
- LM probabilities of source and target sentences,
- number of occurrences of words in the hypothesis averaged for all words,
- average number of translations per source word with probability greater than 0.2 or 0.01,
- percentage of higher/lower frequency unigrams, bigrams and trigrams in the source sentence,
- number of punctuation marks in the source and the target sentences.

This baseline feature set is relatively strong: the majority of the participating systems in the WMT-12 QE task could not beat it.

Since then, the majority of best-performing QE systems use 17 baseline features in combination with some other features. **Decoder features** perform well, but require the

internal information from the MT system. Some systems successfully employ other **black-box features** like number of OOV words, source/target LM scores, similarity of source/target sentences and the SMT training and development datasets (Soricut et al., 2012). **Pseudo-reference features** are often reported to perform well. Formiga et al. (2013) compute a range of evaluation metrics (GTM-l, ROUGE, WER, PER, TER) between the target sentence and its pseudo-references and use these scores as features. The approach described by Langlois (2015) is similar, but a sentence is compared with its pseudo-references in terms of ngrams overlap. Moreau and Vogel (2012) also use similarity scores as features, but they are computed by comparing a sentence with some monolingual target language corpus. The similarity metrics used are TF-IDF score and the percentage of sentence ngrams which occur in the reference corpus. However, this approach is not as effective as the use of pseudo-references — the classifier trained on these features could outperform the baseline system.

The use of **linguistic information** was also investigated. Hardmeier (2011) uses tree kernel features — features that measure the similarity between syntactic tree structures of the source/target sentence and sentences in a source/target corpus. Combined with the baseline features, they outperform the baseline system (Hardmeier et al., 2012). The system by Rubino et al. (2013b) includes a wide range of linguistic features, such as source and target part-of-speech and syntactic information, and topic modelling features. The topic modelling features were also shown to be useful in sentence-level QE (Rubino et al., 2013a).

QE models by Camargo de Souza et al. (2013, 2014) use a number of features from **alignments** between the source and the target sentences which help to better identify the adequacy of translations. Another new group of features used in these works are **back-translation features**: similarity scores between the source sentence and the automatic translation of a target sentence back into the source language. They also demonstrate that a sentence-level QE system can benefit from word-level predictions: they include a number of features that summarise the information about labels assigned to words by a word-level system: number of words predicted on "OK"/"BAD" with different parts of speech, the most common patterns of label ngrams (e.g. "OK OK OK", "OK BAD BAD", etc.), length of "OK"/"BAD" predicted spans.

At WMT-15 the winning systems use features based on back-translations (Langlois, 2015) or translation and LM probabilities (Tezcan et al., 2015). The latest work on sentence-level QE is influenced by the spread of neural networks. Some of successful systems submitted to WMT-16 use neural network architectures which take source and target sentences as input directly and do not need feature engineering (Kim and Lee, 2016). However, systems that

use extensive linguistic information, pseudo-reference and back-translation features can still compete with them Kozlova et al. (2016).

3.2.3 Algorithms

The problem of evaluating the quality of an automatically translated sentence can be interpreted as a classification task (when the quality is one class out of the fixed set of two or more classes), a regression task (when the quality score can be a number in a fixed range, e.g. [0, 1]), or ranking task (when the task is to rank sentences from best to worst).

Therefore, the machine learning algorithms used for sentence-level classification task include a variety of techniques. While the first CE systems described by Blatz et al. (2004) used multi-layer perceptron and Naive Bayes, farther QE systems, including the system used as a baseline at WMT shared task, usually achieved best results using Support Vector Machine (SVM). This technique represents all data instances as points in a multidimensional space, and during training uses the closest instances of different classes to find a boundary between the classes. Soricut et al. (2012) show higher performance with M5P. This is a technique which builds a decision tree where each leaf is a linear regression function. It allows to model non-linear regression.

The regression models employed in QE systems include SVM regressor (it was used in many works, e.g. DCU-SYMC system (Rubino et al., 2013b), CNGL system (Biçici, 2013) and the baseline sentence-level system (Bojar et al., 2013)), Partial Least Square regressor (Specia et al., 2009). Random forest classifier was successfully used for sentence-level QE (Formiga et al., 2013). Recently, there was a lot of work on training QE systems with neural networks Kim and Lee (2016); Shah et al. (2016).

Another aspect of training is data selection, which has been shown to be useful for sentence-level QE systems. Beck et al. (2013) perform active selection of training sentences using Gaussian processes. They show that training on a carefully selected small datasets results in higher performance than using all of the training data.

3.3 Word-level Quality Estimation

As was discussed before, early work on QE for MT aimed at estimating the quality of individual words. The main reason for that was the fact that such word-level estimates could be used as feedback to an MT system itself to improve its quality. In addition to that, the sentence-level QE task was infeasible at that time because MT quality was poor: most of sentences produced by them were incorrect overall, but contained correct subsegments.

However, word-level QE did not become popular for a number of reasons, mainly the unavailability of reliable data. Similarly to sentence-level QE, the efforts to generate the QE datasets by comparing automatic translations with references resulted in very noisy labelling.

The task was re-established later, when post-editing of automatic translations became a common practice and datasets manually labelled at the word level appeared. Since WMT-13, word-level QE has been a part of QE shared task, which motivated many groups to do research in this field.

3.3.1 Labels

While most of sentence-level quality labels are numerical, it is often impossible to find a number associated with the quality of an individual word. A user of an MT system can only identify that a word either "fits" or "doesn't fit" the context, therefore, an obvious solution would be to use **binary labels** ("good"/"bad") as target values for the word-level QE. However, such direct human evaluation of MT is laborious and sometimes infeasible. Therefore, an easier way of labelling automatic translations at the word level is manual post-editing of an MT system output: a translator edits an automatically translated sentence, and the words which were edited are labelled as "bad".

The set of labels itself does not have to be binary: it is possible not only to find an erroneous word, but also to identify which type of error an MT system has introduced. For labels defined automatically this corresponds to the **type of edit operation** performed on the word: substitution of a word with another word, deletion, insertion or shift. Note that the "deletion" cannot be labelled as such: deletion error is a word which is missing from an automatic translation, and labels can be given only to words which are present in the sentence. A dataset labelled with edit operations was used for WMT-13 word-level QE shared task (Bojar et al., 2013).

This set of errors is quite limited and the error types do not reflect the causes of errors. Vilar et al. (2006) introduced a classification of errors aimed at automating error annotation. Vilar et al. distinguish five main error types: missing words, wrong word order, incorrect words, unknown words and punctuation error. While some of categories are similar to edit operations presented above, each of top-level classes has linguistically motivated subclasses, e.g. the "unknown words" class is divided into a subclass of unknown words and a subclass of unknown forms of words that exits in a translation model.

More fine-grained error typologies such as **MQM** (Lommel et al., 2014) divide all errors into adequacy, fluency and verity errors, which account for wrong translation, wrong choice of word forms and word order and errors in the source which are propagated on translation, respectively. Each class has subclasses. Datasets manually labelled with MQM errors were

used in WMT-14 QE shared task (Bojar et al., 2014). While there exist tools for automatic identification and annotation of certain MT error types (Popović, 2011), they have not been used for the generation of word-level QE training datasets because of low reliability. Moreover, not all error types can be identified automatically: for example, subclasses of the "Fluency" class of the MQM typology (use of a wrong part of speech, wrong word order, incorrect use of function words) can only be assigned manually, especially if several types of errors overlap.

The later word-level QE shared tasks used only binary labels. There are several reasons for that. First, the binary labels are easier to acquire — they can be obtained from post-edits, whereas manually assigning fine-grained error classes is more time-consuming task than post-editing and arguably more prone to disagreement. Converting post-edits to fine-grained error tags is a noisy process and may result in incorrect tags, which would influence the performance of QE models. Secondly, since the latest MT systems tend to perform well, the post-editor does not need to do much editing, which results in a low number of corrected words (10–20%). It means that if such data is used to train a QE system the number of instances of the "good" class can be 5 to 10 times larger than the number of "bad" words. This imbalance becomes an even more critical sparsity issue when there are multiple error classes.

3.3.2 Features

One of the first examples of features used for word-level QE is word posterior probability (WPP), which is computed as a sum of probabilities of translations which include the word. However, more recently the *black-box* word-level features have been explored as well. Luong et al. (2014b) describe a set of features that use statistics that are either extracted from the automatic translation and source sentence, or come from MT system-independent tools and datasets:

- source and target contexts of the word under consideration,
- frequencies of ngrams containing the word (frequencies are computed based on an external corpus),
- POS tags of the word, the corresponding source word and their contexts,
- word's constituency label and depth in a syntactic constituency tree,
- binary features indicating if the word is a stopword, a proper name, a number,
- the number of senses of the word in the WordNet database,
- occurrence of the target word in a pseudo-reference translation.

These features have been used for the baseline word-level QE system since 2015 (Bojar et al., 2016, 2015).

It was shown by Luong et al. (2014b) that the pseudo-reference feature (in this case, a binary feature indicating if a word exists in a translation produced by a third-party MT system) improves the results by a large margin. This is corroborated by the work of Esplà-Gomis et al. (2015), which describes the best-performing system at WMT-15 that used only features extracted from a number of pseudo-reference translations. On the other hand, Kreutzer et al. (2015) show that even a baseline set of features extended with feature interactions, or concatenations of features (features of the form "$f_i|f_j$" where f_i and f_j are categorical features) can perform closely to models which use pseudo-references.

Later models use the word vector representations (word embeddings) as features. Word vector representations are fixed length vectors that encode the information on the word context. Many techniques have been used for training these representations, the most widely used are CBOW and skip-gram (Mikolov et al., 2013), both of which are included in `word2vec` toolkit[1]. The word representations can be pre-trained and then used as features. Alternatively, word vectors and target labels can be trained jointly as it was done in a word-level QE system described by Kreutzer et al. (2015). As stated by Schnabel et al. (2015), the latter way should be preferred, because the performance of word embeddings in downstream tasks does not correlate with the quality of embeddings defined with intrinsic evaluation techniques. In other words, it is difficult to choose the word embedding technique suitable for a particular task and the fact that it is good for some other task does not make it "good" in general. On the other hand, the word vectors trained externally can use unlabelled datasets which are much larger than labelled data for a QE task.

3.3.3 Algorithms

Since the quality of words in automatic translations is usually represented as a label (presence of error and/or its type), the task of word-level QE is usually formulated as a classification task. Since words and errors in a sentence are related to each other, it is common to perform the sequence labelling on the automatic translation rather than classifying every word independently.

Conditional Random Fields (CRF (Lafferty et al., 2001)) — one of the best-performing sequence labelling models — is widely used for word-level QE (Luong et al., 2014b; Shang et al., 2015). However, other classifiers have been successfully used for QE as well. The UAlacant system (Esplà-Gomis et al., 2015) uses a multi-layer perceptron to predict word-

[1] https://code.google.com/p/word2vec/

level labels. Kreutzer et al. (2015) describe a linear classifier trained on baseline features. Despite its simplicity, this model outperforms many more sophisticated QE systems submitted to the WMT-15 shared task. Tezcan et al. (2016) achieve high performance by training a Random Forest classifier.

In the last couple of years, deep learning algorithms were applied to word-level QE task. The system by Kreutzer et al. (2015) uses feed-forward neural network. It trains word vector representations from context windows around the target word and the source word aligned to it, then these representations are combined with baseline features. Camargo de Souza et al. (2014) train a recurrent neural network on features extracted from N-best lists. The best-performing models at WMT-16 also use neural networks: Kim and Lee (2016) use recurrent neural network, Martins et al. (2016) combine feed-forward, recurrent and convolutional neural networks.

3.4 Tools

QE is a relatively recent direction of research, so there were not many publicly available QE tools. Currently there exist two open-source tools: `QuEst++` (along with its predecessor `QuEst`) and `WCE-LIG`. The former provides functionality for extracting features and training models at all levels of granularity except phrases, whereas the latter deals with word-level features and models.

Alongside the `QuEst++` and `WCE-LIG` we present `Marmot` — our new tool for subsentence-level QE. It has some advantages over the existing tools in terms of usability and is the only tool that implements phrase-level feature extraction (the phrase-level QE is described in section 5.1).

3.4.1 Existing tools

`QuEst` (Specia et al., 2013) was the first freely available tool for QE. It supports only sentence-level QE, but offers a big range of features: features that consider translation probabilities of words, LM scores for source and target parts of sentences, linguistic features such as POS-tags and syntactic relations. They use only source and target segments or are extracted from N-best lists, word lattices and other MT-system specific information.

`QuEst++` (Specia et al., 2015) is an extended version of `QuEst`. Its main difference is the support to new levels of granularity: besides sentences, it extract QE features of words and documents. The range of available document-level features is wide, however, the word-level features are limited to the ones listed in the work by Luong et al. (2014c).

Both tools' feature extraction pipelines are implemented in Java, with scripts for training of sentence-level models implemented in Python and exploits `scikit-learn` library[2]. The features are persisted in tsv format, so model training can also be performed with third-party tools.

Another recently issued QE tool written in Python — `WCE-LIG` (Servan et al., 2015) — is limited to word-level QE models. Similarly to `QuEst++`, it contains the features described by (Luong et al., 2014c) with the addition of some new features. Besides that, it can perform feature selection. One of drawbacks of `WCE-LIG` is that it is quite resource-intensive. Its feature extractors need POS-taggers, pseudo-reference translations performed by online MT systems, N-best lists produced by `Moses` and the `Moses` training scripts, Berkley parser, BabelNet, `TERp`, `GIZA++`, etc. This characteristic does not refer to the tool *per se*, but to the features it extracts. However, all these tools and resources are compulsory for running feature extraction, and the absence of any of them stops the whole process.

3.4.2 Marmot[3]

In the beginning of this PhD work there were no publicly available word-level QE tools, therefore, we created such a tool for our research. `Marmot` is a tool for performing QE at the level of words. It is written in Python, is highly flexible and modular, and benefits from numerous Python libraries.

Despite the emergence of other word-level QE tools, Marmot still has features that are not present in the word-level QE tools. Compared to `QuEst++` which is written in Java, `Marmot` is much easier to modify and embed into other applications: `Marmot` is written in Python, which is an easier language to read and modify. Moreover, Python is more popular in the research community, therefore it offers more useful libraries.

`WCE-LIG` is also implemented in Python and the set of available features overlaps that of `Marmot` for many features. However, we claim that `Marmot` is more flexible: it allows easier extension with new features, it is language-independent and more configurable. As it was noted, `WCE-LIG` does not allow end user to choose which features to extract — instead of that, all features are extracted, so the absence of any third-party tool stops the feature extraction. In `Marmot` the user can specify the preprocessing steps that need to be invoked and the features that s/he wants to generate. If some steps (e.g. POS-tagging) do not need to be run, they can be skipped safely, and the whole pipeline can be directly managed via the configuration file without changing the source code.

[2]http://scikit-learn.org/

[3]Joint work with Chris Hokamp (Logacheva et al., 2016b). C.Hokamp suggested and partially implemented the system architecture.

Overview

A QE system should be able to perform three main tasks: extract features from the data, train a model and perform tagging of new data using this model. `Marmot` is designed to perform all of these tasks in a flexible, efficient and transparent manner. In addition, models can be evaluated within the toolkit. Therefore, its pipelines allow users to go from raw training data to trained and evaluated models with minimal configuration. Experimental pipelines can be completely specified via configuration files, speeding up the development process. `Marmot` takes advantage of stable and well-maintained Python libraries that implement common machine learning algorithms and NLP tasks at pre-processing, feature extraction and model building stages.

The system has the following properties:

- it is written in Python, which is easy to learn, read and modify,
- users can easily extend its functionalities by adding support for new data formats and features,
- any algorithm from `scikit-learn` can be directly used within the training pipeline or features can be used with external tools (`Weka`, `CRF++`, etc.),
- the feature extraction is parallelised, which makes it much more efficient,
- experiment configuration files allow the use of the toolkit and the creation of pipelines without writing new code.

Design

Because of the modular architecture, users can easily add or implement new parsers, data representations, and features that fit their particular use cases.

Figure 3.1 shows the architecture of the complete pipeline in `Marmot`. The grey boxes denote the stages of pre-processing or training which require specification of pre-defined parameters and which are likely to be customised by a user. The white blocks inside the grey boxes denote user-determined procedures. Each stage in the pipeline for an experiment is specified in the configuration file as the path to a Python class or function. Therefore, the framework is agnostic about the actual functions that it calls, allowing the user to create a custom pipeline without writing any code, simply by specifying which parsers, feature extractors, and learning method(s) they wish to use. User-defined functions do not need to be included in the source code of the system, and no changes to the source code need to be made if a user provides a valid path to the function and makes sure it returns the output accepted as input by the next stages.

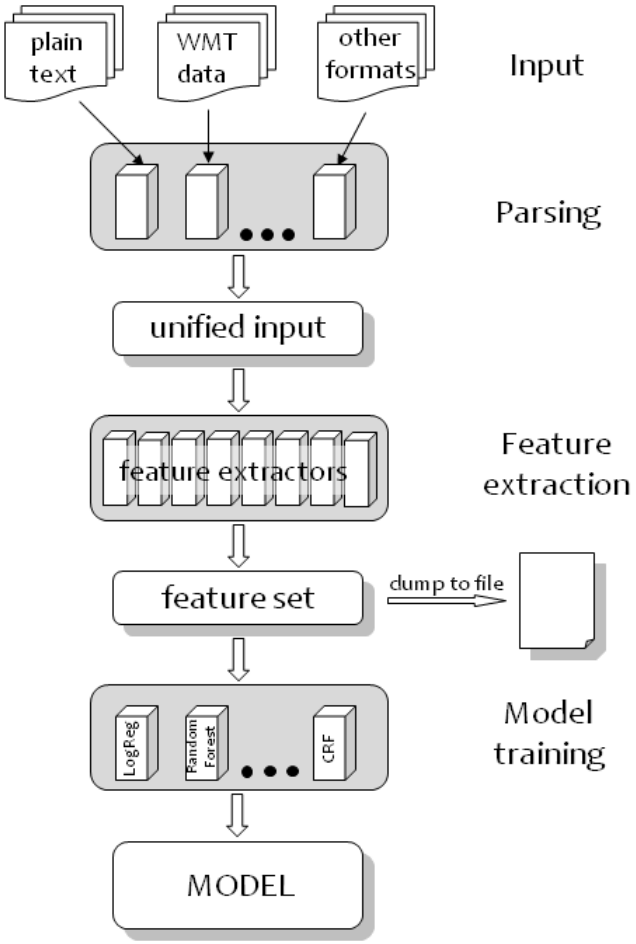

Fig. 3.1 System architecture.

The input data to be parsed is the first of the configurable stages. Data can be provided in many different formats, such as parallel corpora in one or more files, JSON, or XML, and new parsers are straightforward to implement. The parsed data is stored in *context objects*. Each context object represents a training or test example. It contains a target item and all information needed to extract features for this item: its label, the sentence it is taken from, its index in the sentence, and any user-specified additional representations, such as part-of-speech tags.

The context object is implemented as a Python dictionary: it consists of indexed fields which store information about a data instance. The keys of the dictionary are of string type, but the type of value is not restricted, so a context object can store any type of data. Therefore, `Marmot` can be extended to perform QE at any level: while it was originally designed for the word-level QE, i.e. the data instances were individual words, it has been extended to the sentence level without changes in the architecture of the context object.

Feature extractors take context objects as inputs, converting the context object into one or more features (real-valued or categorical). The extracted features can be passed directly to the model training module or persisted to a file for use with external machine learning frameworks. `Marmot` supports formats used by `CRF++` and `CRFSuite` and TSV (tab-separated values) format accepted by many tools.

The `Marmot` pipelines are also very efficient because the most time-consuming stages (parsing, feature extraction, and training) are parallelised.

Data preparation and feature extraction

`Marmot` provides a framework that can extract features from a variety of input formats. Any additional information needed for feature extraction (POS-tags, alignments) can also be acquired at the pre-processing stage, which is often more efficient than computing it during feature extraction. This is especially true when a model for the additional information must be learned over the whole training dataset as, for example, when word-alignment features are used. Computing them during pre-processing allows the `Marmot` experimental pipeline to run much more quickly because they can be re-used across features.

All feature extractors inherit from one abstract base class. Because feature extractors implement the same interface, adding new features is easy, and feature sets can be specified just by listing the desired features in the configuration file.

Although extracting the additional representations at pre-processing stage is more efficient, these can be acquired during feature extraction as well. Hence a feature extractor that requires POS tags will look for them in the context object received as input. If it does not find them, it will try to generate POS tags using a POS-tagger if provided. Finally, if none of these options exists, this feature extractor will raise an informative error, guiding the user in correcting the configuration file and providing the necessary resources. Currently the only POS-tagger that is supported in `Marmot` is `TreeTagger`. However, POS-tagging by a different tool can be provided in the input. The feature extractors are not bound to any format of POS-tags.

The standard interface of the feature extractor makes it easy for users to implement new feature extractors. The current version of `Marmot` already contains a set of word-level features based on the one used by Luong et al. (2014c). The word-level features available in `Marmot` are:

- length of the source sentence,
- length of the target sentence,
- ratio of source and target lenghts,

- target token,
- left context of the target token (context length may be changed, default is 1),
- right context of the target token,
- source word aligned with the target token,
- left context of the source token,
- right context of the source token,
- binary feature indicating if the target token is a stop word,
- target token is a punctuation mark,
- target token is a proper noun,
- target token is a number,
- highest order of ngram which contains target word and its left context,
- highest order of ngram which contains target word and its right context,
- backoff behaviour of trigrams.

Backoff behaviour features contain the LM information: they check if a trigram or bigrams and unigrams that it comprises exist in the LM. This feature is implemented as described in Raybaud et al. (2011). In order to use this feature a user only needs to provide a corpus to train an LM on — the LM itself is not needed, it is generated at runtime using the `SRILM` tool (Stolcke, 2002).

`Marmot` can also make use of word embeddings trained with `word2vec`. We provide feature extractors that make use of word vector representations both at the word and phrase levels. The feature extractors can extract word vectors for source and target tokens and for their contexts of arbitrary length if they are supplied with a pre-trained vector model or a monolingual corpus to train a model. The training process can also run in multiple threads.

Model learning

There is a wide range of possibilities for model training. Classification can be performed with one of the classifiers defined in `scikit-learn`. Sequence labelling is done using the `pystruct`[4] module. Analogously to parsing and feature extraction, the machine learning method used is specified in the configuration file, so any classifier from `scikit-learn` can be used directly. The training module can also easily integrate any machine learning libraries that can take numpy arrays as input, or features can be saved as `.csv` files and used as input to any external algorithm.

The standard approach is to train a classifier for all the training examples. `Marmot` can also train a separate classifier for every type that occurs in the training data. This approach

[4] https://pystruct.github.io/

allows a suite of classifiers to model differences in distributions of errors between individual words or word classes.

Benchmarking

System ID	F_1-**Bad**
UAlacant/OnLine-SBI-Baseline	43.12
HDCL/QUETCHPLUS	43.05
UAlacant/OnLine-SBI	41.51
Baseline features + VW classifier	40.84
SAU/KERC-CRF	39.11
SAU/KERC-SLG-CRF	38.91
• SHEF2/W2V-BI-2000	38.43
• SHEF2/W2V-BI-2000-SIM	38.40
SHEF1/QuEst++-AROW	38.36
UGENT/SCATE-HYBRID	36.72
• **DCU-SHEFF/BASE-NGRAM-2000**	36.60
HDCL/QUETCH	35.27
• **DCU-SHEFF/BASE-NGRAM-5000**	34.53
SHEF1/QuEst++-PA	34.30
UGENT/SCATE-MBL	30.56
RTM-DCU/s5-RTM-GLMd	23.91
RTM-DCU/s4-RTM-GLMd	22.69
• **Baseline features +** `CRFSuite`	16.78

Table 3.1 Official results for the WMT-15 word-level QE task. Systems whose results are significantly different with $p = 0.05$ are grouped by a horizontal line. Systems **in bold** used the baseline feature set. Systems that used `Marmot` toolkit for feature extraction and/or model training are indicated by a •.

`Marmot` was used to extract features for the baseline system in the WMT-15 and WMT-16 word-level QE shared task. These features were also made available to all participants. Table **??** shows the official results of the WMT-15 task. Despite the fact that the baseline system (bottom line of the table) performs poorly, most other systems used either the baseline feature set or `Marmot` pipelines for feature extraction and training show better results. Table **??** also contains an additional result (shown in grey) which is worth mentioning: it was produced by a system trained with Vowpal Wabbit (VW) toolkit[5] on baseline features. The system was a trial experiment by the HDCL team (Kreutzer et al., 2015). It demonstrates that

[5]https://github.com/JohnLangford/vowpal_wabbit/

features extracted with `Marmot` can be very effective if used with the right machine learning algorithm.

System ID	F_1-**Bad**
UAlacant/OnLine-SBI	41.51
UAlacant/OnLine-SBI + Baseline	**43.12**
HDCL/QUETCH	35.27
HDCL/QUETCH + Baseline	**43.05**

Table 3.2 Performance of WMT15 systems with and without the baseline features.

System ID	F_1-**mult** ↑	F_1-BAD	F_1-OK
English-German			
UNBABEL/ensemble	0.495	0.560	0.885
UNBABEL/linear	0.463	0.529	0.875
UGENT-LT3/SCATE-RF	0.411	0.492	0.836
UGENT-LT3/SCATE-ENS	0.381	0.464	0.821
POSTECH/WORD-RNN-QV3	0.380	0.447	0.850
POSTECH/WORD-RNN-QV2	0.376	0.454	0.828
UAlacant/SBI-Online-baseline	0.367	0.456	0.805
CDACM/RNN	0.353	0.419	0.842
SHEF/SHEF-MIME-1	0.338	0.403	0.839
SHEF/SHEF-MIME-0.3	0.330	0.391	0.845
BASELINE	0.324	0.368	0.880
RTM/s5-RTM-GLMd	0.308	0.349	0.882
UAlacant/SBI-Online	0.290	0.406	0.715
RTM/s4-RTM-GLMd	0.273	0.307	0.888

Table 3.3 Official results for the WMT-16 word-level QE shared task. Systems **in bold** used the baseline feature set extracted with `Marmot` (probably extended with other features). Submissions in the grey area are those which are not significantly different from the baseline.

Both winning systems — from HDCL and UAlacant (Esplà-Gomis et al., 2015) teams — were trained using the set of baseline features in addition to system-specific features. Table 3.2 gives the comparison of the performance of these systems with and without baseline feature set. In both cases, the baseline features improve performance significantly. Table 3.3 shows the results of the WMT-16 word-level QE shared task. Some of its participants also used features extracted with `Marmot`.

The `Marmot` tool is opensource. It can be downloaded from https://github.com/qe-team/marmot and modified. The documentation is available at https://qe-team.github.io/marmot/.

The description of Marmot has been published at the LREC-2014 conference (Logacheva et al., 2016b).

3.5 Conclusions

Quality estimation of automatically translated text has started as a problem of estimating the confidence of a generated translation under a particular MT system. By now it has evolved into an independent task with established baselines, and its applications lie beyond the scope of MT itself. There has been much more work done on sentence-level QE than on other levels. This is attributed to the fact that data for sentence-level QE is easier to acquire and sentences are much more generalisable than words, so they require less training data to achieve state-of-the-art performance. However, word-level QE has been gaining popularity because sentence-level quality scores do not contain information on which words are correct and erroneous, which is important for some tasks like automatic post-editing.

A variety of features has been explored for sentence-level QE systems. They include MT system-dependent features, features that can be extracted from the sentences themselves or using additional information from external sources (linguistic information, probabilities of words, word sequences or translations of words). A common trend in sentence-level QE is to use the established baseline feature set as a basis and boost the system performance by adding more information about sentences. The choice of additional features usually depends on the availability of tools and resources for a particular language pair and domain, however, the majority of well-performing systems use the information from pseudo-reference translations.

The range of features used for word-level QE is not so wide. It includes various linguistic features (part-of-speech, syntactic and semantic information for a word and its neighbours), the probability of the word in the source and target context (translation and LM probabilities for the word). Recent word-level QE systems have successfully integrated word vector representations.

Both these levels of granularity have now established baselines and toolkits for extracting features and performing model learning. The QuEst++ toolkit can operate at different levels: it trains QE models for words, sentences and entire documents. It contains a wide range of features of different types. The WCE-LIG tool serves for training of word-level QE models, implementing a set of baseline word-level features and some additional features. Along with the existing tools we present our QE toolkit Marmot. It works at word level and is implemented in Python, which is easier to use and modify. It provides access to Python NLP and Machine Learning libraries, and can extract some specific subsentence-level features

which are not available in other QE tools, for example phrase-level features. It is also more flexible and modular than other toolkits.

Despite the growing popuarity of work on QE the majority of state-of-the-art QE models are still far from performance levels that would allow them to be used in real world tasks. One of the main reasons for that is data scarcity. Although QE model training needs less human-labelled data than the methods discussed in chapter 2, existing datasets are often too small, and the majority of language pairs have no QE datasets at all. Moreover, many of the available datasets are not suitable for the tasks: since automatically translated sentences often contain very few errors, QE datasets do not have enough examples of MT errors, which hampers error detection. In the subsequent chapter we describe some ways of solving these problems by creating artificial datasets and manipulating the existing data to improve the performance of QE models.

Part II

Quality Estimation

Chapter 4

Data manipulation strategies for QE system training

The quality of MT has been going up in recent years, so an automatically translated text often does not need much editing. As far as QE is concerned, the improved accuracy of MT system leads to an increased proportion of positive examples in the quality-labelled datasets: most words in automatically translated sentences — and often entire sentences — can be labelled as good (needing no editing). This fact, while being positive in general, can make QE harder: the QE training data becomes imbalanced. QE models trained on the data with an insufficient number of negative examples tend to be bad at detecting incorrect translations.

In this chapter we report our work on this problem with a number of different data preparation techniques. First, we manipulate the existing human-labelled datasets: we select the most appropriate subset of training data (section 4.1.1) or create new data instances from the existing training sentences (section 4.1.2). For the cases where the training data is too scarce and does not allow filtering we suggest a different approach: we generate the new artificial sentences for training (section 4.2). We show that manipulations with training data can improve the performance of a model substantially even without changing the training algorithms.

	avg.HTER	HTER = 0	0 <HTER <0.2	0.2 <HTER <0.5	HTER >0.5
WMT-15	0.23	11.2%	41.1%	38.4%	9.2%
WMT-16	0.27	15.4%	28.5%	41.0%	15.1%

Table 4.1 Distribution of sentence-level HTER scores in WMT-15 and WMT-16 quality-labelled datasets.

4.1 Manipulation of existing data[1]

4.1.1 Filtering of the data

The majority of datasets that contain automatic translations labelled for quality have a low number of errors: their average HTER scores are often below 0.2, which means that the number of erroneous words is five times lower than that of correct words. Table 4.1 outlines the statistics of error frequency in two QE datasets: the English–Spanish dataset used in QE shared task at WMT-15 (Bojar et al., 2015) and the English–German dataset released for the shared task at WMT-16 (Bojar et al., 2016). We see that overall the data has low proportion of errors: the average values do not exceed 0.3 (30% of incorrect words) and most sentences have HTER of 0.5 or lower (with around half of sentences scoring below 0.2).

An additional illustration of this imbalance is provided in figure 4.1 which shows the HTER scores distribution for the WMT-15 dataset: 50% of the sentences have HTER of 0.15 or lower (points below the bottom orange line in the figure), 75% of the sentences have HTER of 0.28 or lower (points below the middle green line).

This situation can be particularly harmful for word-level QE models: a large number of sentences with few or no edits biases these models to tag more words as "OK", i.e. the tagging becomes too optimistic, which results in higher F_1-score for the "OK" class and lower F_1-score for the "BAD" class. The QE shared tasks use F_1-BAD score as a primary metric, because the main goal of QE models is to find errors, so we would like to improve this score in the first place. However, we should aim at a reasonable balance between the two scores, because the high F_1-BAD score alone is often not an indication of good quality: a naive labelling of all words as "BAD" can yield quite high F_1-BAD score, although it cannot be considered a good labelling.

The easiest way to change the distribution of classes is to add more negative examples or remove positive examples. In table 4.1 we see that both datasets contain around 15% sentences which have no errors. While sentence-level QE models might still need these

[1]This is a part of a joint work with Chris Hokamp (Logacheva et al., 2015). The part of work described here was done by V. Logacheva.

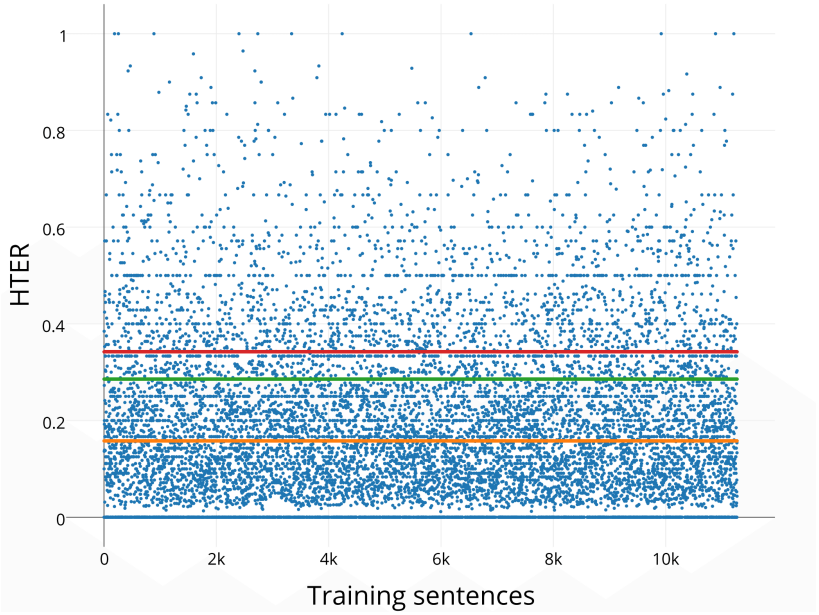

Fig. 4.1 Distribution of HTER scores for the training data of the WMT-16 dataset: each blue dot represents a training sentence. Dots below the orange line make 50% of the data, dots below the green line, 75% of data, dots above red line, the worst 2000 sentences (18% of the data).

sentences to be able to correctly classify instances with no errors, the word-level models already have enough examples of good words, so error-free sentences just increase the bias towards the "OK" label and bring little useful information. Therefore, we assume that we can remove these sentences with minimal harm to the model. Besides, there are many sentences whose HTER is higher than zero but still small. These sentences also increase the bias while providing few negative examples, so some of them can also be eliminated.

In order to filter out sentences that have too few errors, we performed a simple training data selection strategy on the WMT-15 (English–Spanish) dataset: we used only sentences with the highest amount of errors. To define the optimal number of sentences to select, we sorted sentences in the original training corpus by their HTER in decreasing order (the worst sentences first) and built models on top N sentences from this sorted set, where N ranges from 1,000 to 11,000 (the entire dataset). Figures 4.2 and 4.3 show the results of application of this strategy to the WMT-15 baseline system (denoted as "CRF" in figures). Its F_1-BAD score on the full WMT-15 dataset is 16.78, as shown in table 3.1. We compare it with another system that uses the same set of baseline features and is trained with a `scikit-learn` implementation of Random Forest classifier with default parameters (denoted as "Random Forest"). In figure 4.2 we plot how the F_1-BAD score changes as we increase the number of sentences in the training data to include less erroneous ones.

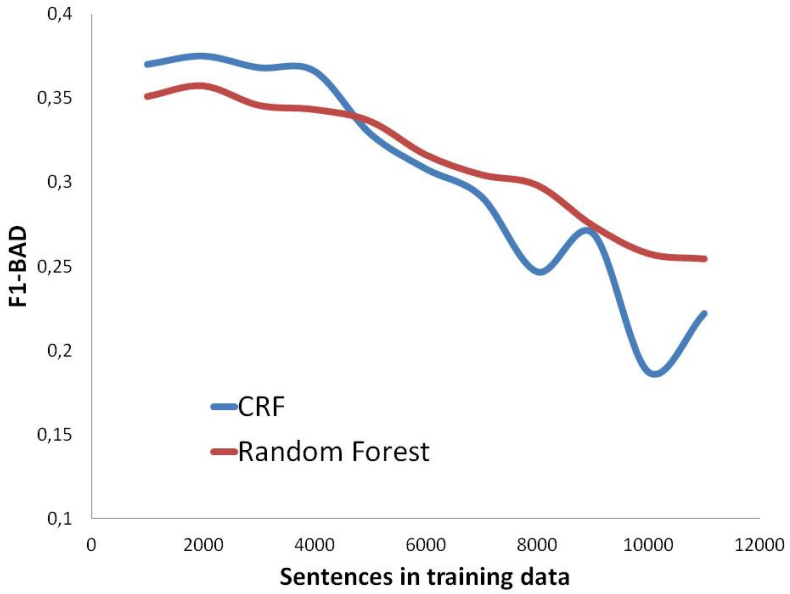

Fig. 4.2 Performance of WMT-15 baseline system (**CRF**) and a Random Forest classifier with baseline features (**Random Forest**) trained on subsets of WMT-15 training data (F_1 for the "BAD" class).

Both tested models benefit from data filtering: we see that training on a subset of the original dataset containing 2,000 worst sentences results in the best-performing (in terms of F_1-BAD) model for both Machine Learning algorithms. The best result is 0.37 for CRF and 0.35 for Random Forest, which is a big improvement compared to F_1-BAD scores of 0.22 and 0.25, respectively, for models that use the full training data. CRF achieves higher F_1-BAD score, but Random Forest is less sensitive to the increasing number of good examples.

However, this improvement can be meaningless if it is accompanied by a strong deterioration of F_1-OK score, so we also compare how the F_1-score for the "OK" class is affected by the reduction of data size and changed distribution of labels (see figure 4.3). As expected, the F_1-OK score dropped when most of better-translated sentences were filtered out. Nevertheless, we can see that it stabilises soon: when the number of training instances reaches 5,000 the difference in the original F_1-OK score and the F_1-OK of models trained on filtered data gets quite small.

The exact size of the training set needed for the best trade-off between F_1-BAD and F_1-OK depends on the dataset and should be adjusted to a particular task, but our experiment clearly show that data filtering technique can be effective in the case of unbalanced data and can increase the score for the minority class with a small decrease of performance for the majority class.

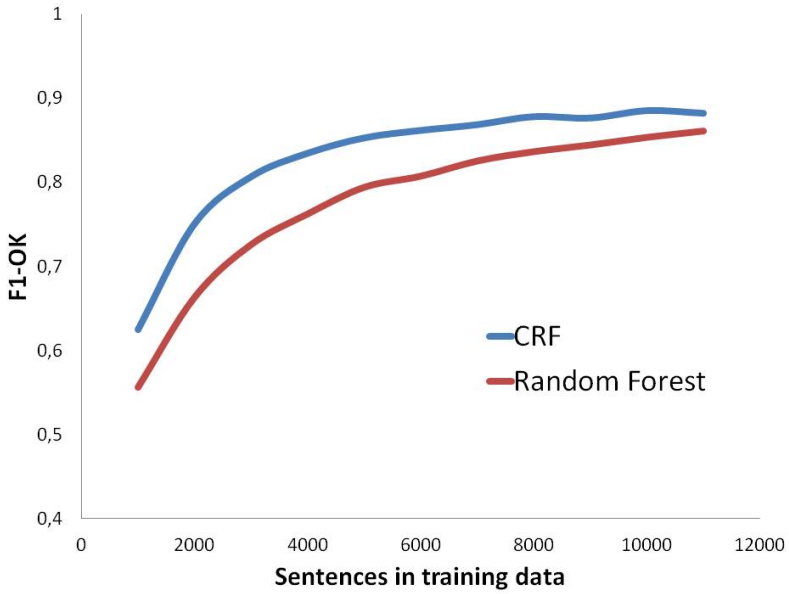

Fig. 4.3 Performance of WMT-15 baseline system (**CRF**) and a Random Forest classifier with baseline features (**Random Forest**) trained on subsets of WMT-15 training data (F_1 for the "OK" class).

4.1.2 Bootstrapping of the data

Although the size of some recently released quality-labelled datasets is considerably larger than that of datasets used before, the available data may still be too sparse to perform QE at the word level. This happens because word-level models use more lexical features which can be too sparse if the size of training data is small. There are techniques for training of individual models with a small subset of most suitable training instances for every example of dev/test (Bicici, 2016). However, this scenario is not optimal for a number of reasons: first, retraining a model for every new instance slows down the prediction process, which is a drawback for real-world and interactive task, and secondly, in the settings where the training data is insufficient, further filtering might result in poor quality. Therefore, although we showed that filtering of the data is often beneficial, we should also look for solutions which do not reduce the training data size but rather increase it.

We should also bear in mind our final goal to improve MT quality with QE predictions. This task might require predicting the quality of a sentence while it is being translated, in other words, to make predictions on unfinished sentences. While it is conceptually possible in the QE word-level task (nothing stops us from estimating the quality of words which have been generated so far, even if the sentence has not been finished), existing QE datasets might be unsuitable for that. They contain only complete sentences, and their performance

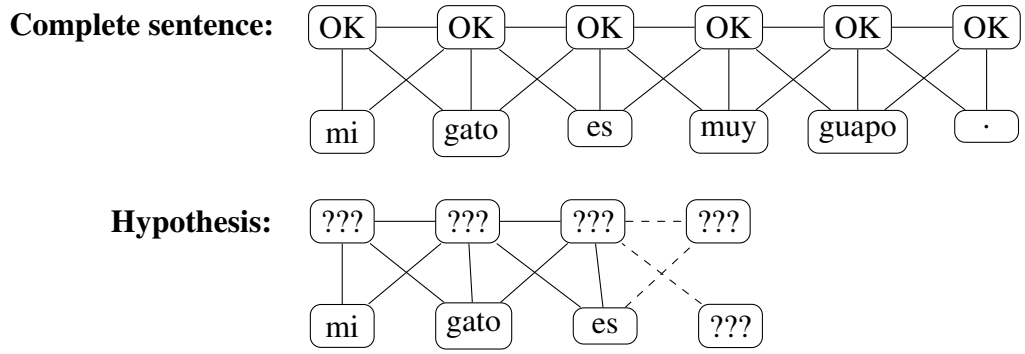

Fig. 4.4 CRF model for complete and incomplete sentences.

on partial sequences can be low. This particularly concerns CRFs, which are widely used for word-level QE: CRF has undirected connections between elements, so all elements in a sequence can potentially influence each other, including distant pairs. So it is possible that a QE model trained on regular data will consider an unexpected end of sequence as a sign of incorrect translation and propagate this information back towards its beginning. See figure 4.4 for an example of such uncertainty in a CRF model: here, every label is influenced by the word it is assigned to as well as words to the left and right and their labels. While in a complete sentence all words are known and their labels can be defined according to a pre-trained model, in an incomplete MT hypothesis the absence of a word propagates uncertainty onto the label of the previous word and all the way back.

On the other hand, if the training data contains examples of unfinished yet correct sentences, we could get a more objective labelling for unfinished sequences. There are no unfinished sequences in the original QE datasets, but we can generate them from complete sentences in the training data. We suggest two methods of additional data generation:

- **1-to-N** sequences: generate sequences that consist of the first n words of a sentence, where $n \in [1, N]$ (N = number of words in the sentence). For example, for each sentence of 10 words we should add nine new training examples: a sequence that consists of the first word only, a sequence that consists of the first two words, the first three words, etc.

- **ngrams**: extract all trigrams of a sentence: for a sentence $w_1 w_2 ... w_i ... w_n$ those will be $w_1 w_2 w_3, w_2 w_3 w_4, ..., w_{i-1} w_i w_{i+1}, ..., w_{n-2} w_{n-1} w_n$.

The bootstrapping of the test set is produced as follows. In order to label a sequence $\mathbf{s} = s_1 s_2 \ldots s_n$ we convert it into a list of n sub-sequences $L_\mathbf{s} = [s_1;\ s_1 s_2;\ s_1 s_2 s_3;\ \ldots;\ s_1 s_2 ... s_n]$. Each sub-sequence from $L_\mathbf{s}$ is labelled by the system. The final labelling for every word $s_i \in \mathbf{s}$ is taken from a sub-sequence where s_i is the last symbol, so that we compose the final labelling

for the sequence **s** from the tags for words s_i listed in bold: $[\mathbf{s_1}; s_1\mathbf{s_2}; s_1s_2\mathbf{s_3}; \ldots; s_1s_2\ldots\mathbf{s_n}]$. This test set bootstrapping process uses the **1-to-N** bootstrapping technique; the **ngram** bootstrapping is performed analogously.

Our hypothesis is that the largest gain is achieved when the training and test datasets are bootstrapped using the same technique. However, we test all the combinations of training and test sets, including the *duplication* of the training data: here each training sentence is copied as many times as there are words in the sentence: thus, words from longer sentences get the largest weight. The motivation of this duplication is the following. Both bootstrapping techniques increase the size of the training data, therefore, any improvements achieved with bootstrapped training sets can be attributed to the enlarged dataset. So we train a model which contains more data but no incomplete sentences. We do not duplicate all sentences the same number of times because such transformation is unlikely to bring any improvements as the relative weight of every training example stays the same. By creating variable number of duplicates we mimic the **1-to-N** technique: there, for every sentence of N words we get N new sequences, each of them being a substring of the original sentence. Here, for a sentence of N words we also create N sequences, but this time we simply copy the sentence itself.

The results of our experiments are outlined in table 4.2. We trained word-level models on WMT-15 word-level QE dataset (Bojar et al., 2015) using the `CRFSuite`[2] toolkit. We tested bootstrapping techniques for training and test sets (with *baseline* setting meaning no bootstrapping). We trained two QE models: one uses the LBFGS optimisation algorithm, the other uses passive-aggressive algorithm. The table shows that the results contradict our hypotheses: the bootstrapped LBFGS models perform better than the baseline, but the for passive-aggressive models that does not hold. In addition, the improvements of the LBFGS model trained on duplicated data are larger than those of bootstrapped LBFGS models. Finally, none of the bootstrapped models shows good results on a test set bootstrapped with the same technique: e.g. the ngram-bootstrapped LBFGS model is better at labelling original and 1-to-N-bootstrapped test sets, the 1-to-N-bootstrapped passive-aggressive model achieves higher score for the original test set (0.275) than for the 1-to-N-bootstrapped test set (0.258). The better performance of bootstrapped LBFGS models compared to the LBFGS baseline should be attributed to the unsuitability of LBFGS algorithm for the task, because the passive-aggressive baseline could not be beaten by any of bootstrapped models.

Therefore, the bootstrapping of the data cannot produce better scores neither for complete nor for incomplete sequences. In fact, the issue of incomplete sequences in the test data is questionable: as shown in table 4.2, the models trained on baseline data are capable of labelling the incomplete sentences. The results for bootstrapped test sets are even higher than

[2]http://www.chokkan.org/software/crfsuite/

Training \ Test	LBFGS			Passive-aggressive		
	baseline	1-to-N	ngrams	baseline	1-to-N	ngrams
baseline	0.129	0.205	0.213	0.283	0.304	**0.327**
1-to-N	0.237	0.238	0.227	0.275	0.258	0.227
ngrams	0.216	0.201	0.181	0.257	0.254	0.232
duplicated	0.246	**0.274**	0.273	0.273	0.299	0.286

Table 4.2 Experiments with bootstrapped data (F_1-score for "BAD" class). The *baseline* setting means no bootstrapping (original data).

for their original versions. On the other hand, since the models trained with two different algorithms give the opposite results, we can assume that the bootstrapping of the data can be useful for certain algorithms.

4.2 Generation of artificial data[3]

As was previously discussed, datasets that contain automatic translations labelled for quality often lack negative examples, and filtering of the data (see section 4.1.1) sometimes cannot be applied, for example, when the dataset is too small. In such cases we should resort to the opposite strategy: instead of removing positive instances we can add examples of bad translations. Manual labelling of automatic translations is a laborious task and often is not affordable, so we would like to find a cheap way of generating and labelling bad translations.

Earlier work on QE followed the hypothesis that machine translations can be assumed to have low quality (Gamon et al., 2005b), which is not the case nowadays, so we cannot label all machine-translated sentences as bad. Instead, we can use automatic quality evaluation metrics based on reference translations: we can automatically translate a large amount of data and use BLEU, TER, METEOR or some other metric to compare these translations with their references (the data which we translate should be taken from a parallel corpus), and use the acquired scores as labels. However, these scores can be very unreliable, especially for word-level QE, where every word that does not occur in a reference will be labelled as bad, although it can be a correct paraphrase.

Previous efforts have been made for negative data generation, including random generation of sentences and the use of translations in low-ranked positions in N-best lists produced by statistical MT systems as the examples of low quality sentences (section 4.2.1). These methods are however unsuitable for QE at the word level, as they provide no information about the quality of individual words in a sentence. Here we adopt a different strategy: we

[3]A part of this work was published in (Logacheva and Specia, 2015b).

insert errors in otherwise correct sentences. This provides control over the proportion of errors in the negative data, as well as knowledge about the quality of individual words in the generated sentences. The goals of the research presented here are to understand the influence of artificially generated data (by various methods and in various quantities) on the performance of QE models at both sentence and word levels, and ultimately improve upon baseline models by extending the training data with suitable artificially created examples. In section 4.2.1 we further review existing strategies for artificial data generation. We explain our generation strategies in section 4.2.2. In section 4.2.3 we describe our experiment and their results.

4.2.1 Previous work

Many text generation tasks evaluate a generated utterance with a probability distribution computed on a set of well-formed texts: the more similar to the training data, the better. However, some tasks need to explicitly define which outputs are good and which are bad and these tasks usually lack the examples of erroneous sentences or texts.

Discriminative language modelling

One example of task that requires low quality examples is discriminative language modelling (DLM), i.e., the classification of sentences as "good" or "bad". It was first introduced in a monolingual context within automatic speech recognition (Collins et al., 2005), and later applied to MT. While in speech recognition negative examples can be created from system outputs that differ from the reference (Bhanuprasad and Svenson, 2008), in MT there are multiple correct outputs, so negative examples need to be defined more carefully.

In the work by Okanohara (2007) bad sentences used as negative training instances are drawn from the distribution $P(w_i|w_{i-N+1},...,w_{i-1})$: first the start symbol $< s >$ is generated, then the next words are taken based on the word probability given the already generated words.

Other approaches to discriminative LMs use the n-best list of the MT system as training data (Li and Khudanpur, 2008). The translation variant which is closest to the oracle (e.g. has the highest BLEU score) is used as a positive example, while the variant with high system score and low BLEU score is used as a negative example. Such dataset allows the classifier to reduce the differences between the model score and the actual quality score of a sentence.

Li et al. (2010) simulate the generation of an n-best list using translation tables from SMT systems. By taking entries from the translation table with the same source side they

create a set of alternative translations for a given target phrase. For each sentence, these are combined, generating a confusion set for this sentence.

Quality estimation for MT

To the best of our knowledge, the only previous work on adding errors to well-formed sentences is that by Raybaud et al. (2011). In the work of Raybaud et al. (2011), the training data for the negative data generation process consists of a set of MT hypotheses manually post-edited by a translator. Hypotheses are aligned with the corresponding post-edits using the `TERp` tool (Snover et al., 2008). The alignment identifies the edit operations performed on the hypothesis in order to convert it to the post-edited version: leave word as is (no error), delete word, insert new word, substitute word with another word. Two models of generation of error strings from a well-formed sentence are proposed. Both are based on the observed frequency of errors in the post-edited corpus and do not account for any relationships between the errors and the actual words. The *bigram error model* draws errors from the bigram probabilities $P(C_i|C_{i-1})$ where C_i is an error class. The *cluster error model* generates clusters of errors based on the distribution of lengths of erroneous word sequences in the training data. Substituting words are chosen from a probability distribution defined as the product of these words' probabilities in the IBM-1 model and a 5-gram LM. A model trained only on artificial data performs slightly better than one trained on a small manually annotated corpus.

Human error correction

Another task that can benefit from artificially generated examples is language learner error correction. The input for this task is text that potentially contains errors. The goal is to find these errors, similarly to QE at the word level, and additionally correct them. While the text is written by humans, it is assumed that these are non-native speakers, who possibly translate the text from their native language. The difference is that in this task the source text is a hidden variable, whereas in MT it is observed.

The strategy of adding errors to correct sentences has also been used for this task. Human errors are more intuitive to simulate as language learners explicitly attempt to use natural language grammars. Therefore, rule-based systems can be used to model some grammar errors, particularly those affecting closed class words, e.g. determiner errors (Izumi et al., 2003) or countability errors (Brockett et al., 2006).

More recent statistical methods use the distributions of errors in corpora and small seed sets of errors. They often also concentrate on a single error type, usually with closed class words such as articles and prepositions (Rozovskaya and Roth, 2010). Felice and Yuan

(2014) go beyond closed class words to evaluate how errors of different types are influenced by various linguistic parameters: text domain, learner's first language, POS tags and semantic classes of erroneous words. The approach led to the generation of high-quality artificial data for human error correction. However, it could not be used for MT error identification, as MT errors are different from human errors and usually cannot be assigned to a single type.

4.2.2 Generation of artificial data

The most straightforward strategy for artificial data generation is to create a sentence by taking all or some of its words from a probability distribution of words in some monolingual corpus. The probability can be defined for unigrams only or conditioned on the previous words (as it is done for discriminative LMs). This however is a target language-only method that does not suit the QE task as the "quality" of a target word or sentence is dependent on the source sentence, and disregarding it will certainly lead to generation of spurious data.

Random target sentences based on a given source sentence could be generated with bilingual LMs. However another limitation of this approach is the assumption that all words in such sentences are wrong, which makes the data useless for word-level QE.

Alternatively, artificial sentences can be generated using MT systems for back-translation. The target sentences are first fed to a target–source MT system, and then its output is passed to a source–target system. Back-translations are more similar to the original sentence than to an arbitrary human reference. However, according to our experiments, if both systems are statistical the back-translation is too similar to the original sentence, and the majority of their differences are interchangeable paraphrases. Rule-based systems could be more effective, but the performance of available rule-based systems is bad, and good enough rule-based systems are not available for free.

A two-stage error generation method

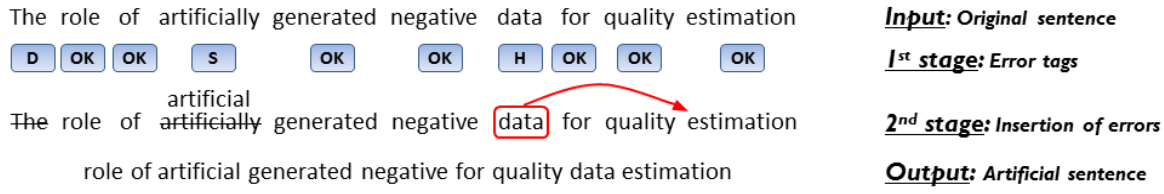

Fig. 4.5 Example of the two-stage artificial data generation process

As previously discussed, existing methods that artificially generate entire sentences have drawbacks that make them difficult or impossible to use for QE. Therefore, following

Raybaud et al. (2011) and previous work on human error correction, our approach is to inject errors into otherwise correct texts. This process consists of two stages:

- labelling of a sentence with error tags,
- insertion of the errors into that sentence.

The first stage assigns an error tag to every word in a sentence. The output of this stage is the initial sentence where every word is assigned a tag denoting a type of error that needs to be incurred on this word. We use five tags corresponding to edit operations in the TERp tool: no error (**OK**), substitution (**S**), deletion (**D**), insertion (**I**) and shift (**H**). During the second stage the words in the sentence are changed according to their tag: substituted, deleted, shifted, or left in place if word has the tag **OK**. Figure 4.5 gives an example of the complete generation process.

Error tagging of sentences We generate errors based on a corpus of post-edited machine translations. We align translations and post-edits using the TERp tool (exact matching) and extract counts on the number of shifts, substitutions, insertions and deletions. TERp does not always capture the true errors, in particular, it fails to identify phrase substitutions (e.g. *was* $\rightarrow$ *has been*). However, since editors are usually asked to minimise the number of edits, translations and post-edits are often close enough and the TERp alignment provide a good proxy to the true error distribution.

The TERp alignments can be used to collect statistics on errors alone or to combine the frequency of errors with the words they are incurred on. We suggest three methods of generation of an error string for a sentence:

- **bigramEG**: the *bigram* error generation that uses a bigram error model regardless of the actual words (Raybaud et al., 2011).
- **wordprobEG**: the conditional probability of an error given a word.
- **crfEG**: the combination of the bigram error model and error probability conditioned on a word. This generation method can be modelled with Hidden Markov Model (HMM) or CRF.

The first model has the advantage of keeping the distribution of errors as in the training data, because the probability distributions used depend only on the frequency of errors themselves. The second model is more informed about which words commonly cause errors. Our implementation of the third method uses CRFs to train an error model. We use all unigrams, bigrams and trigrams that include the target word as features for training. This method is expected to produce more plausible error tags, but it can have the issue that the

vocabulary we want to tag is not fully covered by the training data, so some words in the sentences to tag will be unknown to the trained model. If an unknown word needs to be tagged, it will more often be tagged with the most frequent tag, which is "OK" in our case. In order to avoid this problem we replace rare words in the training set with a default string or with the word class, i.e. the word's POS tag. We also consider the scenario where the POS tags are used as additional features.

The training data needed for this stage of data generation is a corpus of well-formed target-language sentences where each word is tagged with a tag corresponding to an error which is likely to be made in this word during machine translation (we use the same five error tags: 'OK' for correct word, 'I' for insertion of word, 'D' for deletion of word, 'S' for substituting the word with another (incorrect) word, and 'H' for shifting word to another position). This corpus is achieved from a collection of automatic translations with post-edits. We align automatic translations with their post-edits with TERp tool, and see all the differences between them, and the type of these differences (inserted word, changed word, etc.). We take the post-edits with corresponding error tags as the training data for error generators.

Insertion of errors We consider errors of four types: **insertion**, **deletion**, **substitution** and **shift**. Shift errors require the distribution of shift distances which are computed based on the TERp-aligned training corpus. Substitutions and insertions need the new words to be drawn from some list of words with a probability distribution assigned to it, so that every word occurs with some probabilities and probabilities of all words from the list sum to 1.

We suggest three methods for the generation of these lists and distributions:

- **unigramWI**: common list for all the words. The word list is a vocabulary of some large monolingual corpus, the probabilities are frequencies of words in this corpus.
- **paraphraseWI**: separate list for every word (including a special list for out-of-vocabulary words). Every word is assigned a list of possible paraphrases. The paraphrases for a word are defined as follows. First all possible sources of a target word are extracted from an SMT system's lexical translation table. Then the reverse lexical translation table is used to extract all target words that can be translations of these sources (see table 4.3). The probabilities of the paraphrases are computed as $P(w') = P(s|w) \times P(w'|s)$, where w is the considered word, and w' are words from its paraphrase list. The distribution $P(w')$ should be normalised so that it sums to 1. That gives us a confusion set for each target word.
- **lexprobWI**: separate list for every **source** word, which contains the possible translations of the word. This method is similar to the previous one (see table 4.4): the lexical

translation table is searched for the word pairs where the source side matches the word under consideration, all target sides of the found pairs are added to the translation list.

target	source	⇒	source	**target**	→	Paraphrases list
target	source	⇒	source	**target**	→	
target	source	⇒	source	**target**	→	
target	source	⇒	source	target		
target	source	⇒	source	**target**	→	
target	source	⇒	source	target		
target	source	⇒	source	**target**	→	
target	source	⇒	source	target		
target	source	⇒	source	**target**	→	

Table 4.3 Generation of a paraphrase list for **paraphraseWI**. Same colours denote same words.

source	target		Translations list
source	**target**	→	
source	**target**	→	
source	target		
source	**target**	→	
source	target		
source	**target**	→	
source	target		
source	**target**	→	

Table 4.4 Generation of a translations list for **lexprobWI**. Same colours denote same words.

The **lexprobWI** is different from other word inserters: it does not fit into the data generation scenario described above. While **unigramWI** and **paraphraseWI** perform changes to the sentence in the *target* language, the **lexprobWI** needs a *source* sentence. However, the word inserters take their input from the error generators which assign error tags to words of a valid sentence. Therefore, in order to be compatible with **lexprobWI**, error generators need to assign errors to source sentences.

We trained another set of error generators which tags source sentences with errors. That required a different training set: instead of target sentences with error markup we needed analogous source-language data. Similarly to the training data preparation procedure described above, we align automatic translations with post-edits using TERp to achieve the error markup. After that, we align the post-edits with the corresponding source sentences using one of the alignment tools from MT systems, e.g. `GIZA++` (Och and Ney, 2003), and

map the error markup to the source side. Thus, we get a source corpus where each word is tagged with an error which can be made by an MT system while translating this word into the target language.

Notice also that despite taking the source sentence as input, this word inserter outputs a target sentence. Therefore, the whole data generation pipeline for **lexprobWI** is the following:

- the source-side training data for error generator is prepared,
- an error generator is trained on the source data,
- an error generator receives a source sentence and tags it with error tags,
- the **lexprob** word inserter receives a pair of source and target sentences and error tags for the source sentence. It aligns the sentences and maps the error tags to the target sentences,
- the **lexprob** word inserter introduces errors on the source sentence and target sentences simultaneously: the words tagged with 'D' are simply deleted in both languages, the words tagged with 'H' are shifted to the new position. The substitution for the words tagged with 'S' is chosen from the translations list defined for the source word,
- the unaligned (and untagged) words from the target sentence are left without changes,
- the **lexprob** word inserter returns the target sentence with injected errors.

4.2.3 Experiments

We conducted a set of experiments to evaluate the performance of artificially generated data on different tasks of QE at the sentence and word levels.

Tools and datasets

The tools and resources required for our experiments are: a QE toolkit to build QE models, the training data for them, the data to extract statistics for the generation of additional examples.

For the feature extraction we used the `QuEst++` toolkit. We trained the sentence-level QE models using `sklearn` versions of Support Vector Machine (SVM) classifier (for ternary classification task) and SVM regression (for HTER prediction). Word-level classifiers were trained with `CRFSuite`. The CRF error models were trained with `CRF++`[4]. POS tagging was performed with `TreeTagger` (Schmid, 1994). For the sentence-level models we used 17 baseline features (see section A.1.1) for all tasks. For the word-level models we used the set

[4]https://code.google.com/p/crfpp/

of 30 baseline features described by Luong et al. (2014d). The QE models were built and tested based on the data provided for the WMT14 English–Spanish QE shared task.

The statistics on error distributions were computed using the English–Spanish part of training data for WMT-13 shared task on QE[5]. The statistics on the distributions of words, alignments and lexical probabilities were extracted from the English–Spanish portion of Europarl corpus (Koehn, 2005). We trained the alignment model with `fastalign` and extracted the lexical probability tables for words using scripts for phrase table building in `Moses`. For all the methods, errors were injected into the News Commentary corpus[6].

Generated data

Combining three methods of errors generation and two methods of errors insertion into sentences resulted in a total of six artificial datasets. Here we perform some analysis on the generated data.

The datasets differ in the percentage of errors injected into the sentences (see table 4.5). The main differences are between the type of error generator used and the language of dataset where the errors were injected.

The datasets where errors were injected into source sentences have significantly lower percentage of errors. This is explained by the fact that the number of error tags is reduced twice by the cross-lingual alignment procedure. First original error tags from the corpus of post-edits are mapped to the source side. Since the alignment procedure does not necessarily produce alignments for every word, some error tags are lost. Then, after assigning tags to source sentences, the **lexprobWI** maps them back to target sentences losing some tags again. This led to a low number of errors in the artificial datasets, especially those that used CRF-based error generators.

BigramEG datasets have 23% of edits for the target sentences and 12% for the source sentences, which matches the distribution of errors on the real data. **WordprobEG** datasets contain fewer errors for the target sentences and more errors for the source sentences. The **crfEG** models contain the lowest number of errors when applying them to both source and target sentences. As it was expected, data sparsity makes the CRF model tag the majority of the words with the most frequent tag ("OK"). Replacing rare words with a default word token or with a POS tag did not improve these statistics.

We computed the perplexity of all datasets with respect to an LM trained on the Spanish part of the Europarl corpus (see table 4.6). The figures match the error percentages in the data — the lower the number of errors, the more is kept from the original sentence, and thus the

[5]http://www.quest.dcs.shef.ac.uk/wmt13_qe.html
[6]http://statmt.org/wmt14/training-parallel-nc-v9.tgz

Word inserters	Target language (UnigramWI & ParaphraseWI)	Source language (LexprobWI)
Error generators		
BigramEG	23%	12%
WordprobEG	17%	15%
crfEG	5%	0.7%

Table 4.5 Percentage of errors in the artificial datasets.

Word inserters	UnigramWI	LexprobWI	ParaphraseWI
Error generators			
BigramEG	699.9	285.45	888.64
WordprobEG	538.84	357.6	673.61
crfEG + default word	165.36	137.57	172.97
crfEG + POS tag	161.59	139.72	167.23

Table 4.6 Perplexity of the artificial datasets.

more natural it looks (lower perplexity). Note that sentences where errors were inserted from a general distribution (**unigramWI**) have lower perplexity than those generated using using paraphrases. This can be because the **unigramWI** model tends to choose high-frequency words with lower perplexity, while the constructed paraphrases contain more noise and rare words.

Experimental setup

We evaluated the performance of the artificially generated data in three tasks: the ternary classification of sentences as "GOOD", "ALMOST GOOD" or "BAD", the prediction of HTER (Snover et al., 2009) score for a sentence, and the classification of words in a sentence as "GOOD" or "BAD" (tasks 1.1, 1.2 and 2 of the WMT-14 QE shared task[7], respectively).

The goal of the experiments was to check whether it is possible to improve upon the baseline results by adding artificially generated examples to the training sets. The baseline models for all tasks were trained on the data provided for the corresponding shared tasks for the English–Spanish language pair. All models were tested on the official test sets provided for the corresponding shared tasks.

Since we know how many errors were injected into the sentences, we know the TER scores for our artificial data. The discrete labels for the ternary classification task are defined as follows: "BAD" sentences have four or more non-adjacent errors (two or more adjacent erroneous words are considered one error), "ALMOST GOOD" sentences contain

[7] ttp://statmt.org/wmt14/quality-estimation-task.html

one erroneous phrase (possibly of several words), and "GOOD" sentences are error-free. For the sentence ternary classification task we added only "BAD" artificially generated sentences to the training sets, for the rest of the tasks we used all the generated sentences.

The new training examples were added to the baseline datasets. We ran a number of experiments gradually increasing the number of artificially generated sentences used. At every run, the new data was chosen randomly in order to reduce the influence of outliers. In order to make the results more stable, we ran each experiment 10 times and averaged the evaluation scores.

Sentence-level ternary QE task

The original dataset for this task contains 949 "GOOD", 2010 "ALMOST GOOD", and 857 "BAD" sentences, whereas the test set has 600 entries: 131 "GOOD", 333 "ALMOST GOOD", 136 "BAD". The results were evaluated using F_1-score.

The addition of new "BAD" sentences leads to an improvement in quality, regardless of the sentence generation method used. Models trained on datasets generated by different strategies display the same trend: adding up to 400 bad sentences results in a considerable increase in quality, while further addition of data only slightly improves quality.

The best-performing error generator is **crfEG**, however, **bigramEG** performs very closely. Figure 4.6 shows the results of the experiments – here for clarity we included only the results for datasets generated with the **unigramWI**. The best F_1-score of 0.49 is achieved by a model trained on the data generated with the **crfEG** error generator, which is an absolute improvement of 1.9% over the baseline.

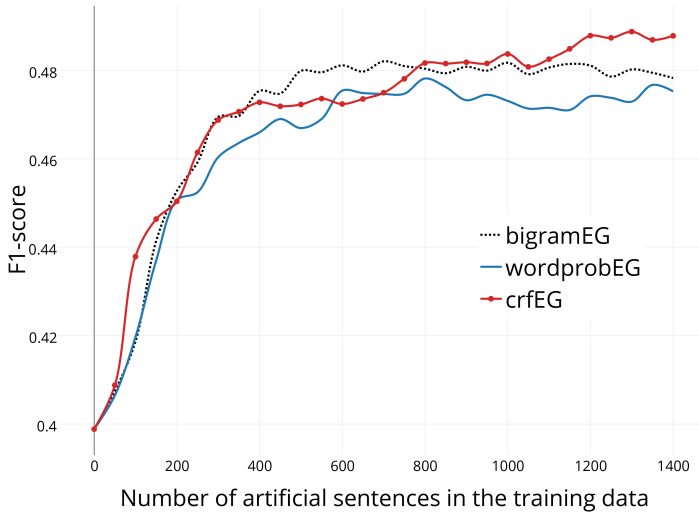

Fig. 4.6 Ternary classification: performance of different error generators.

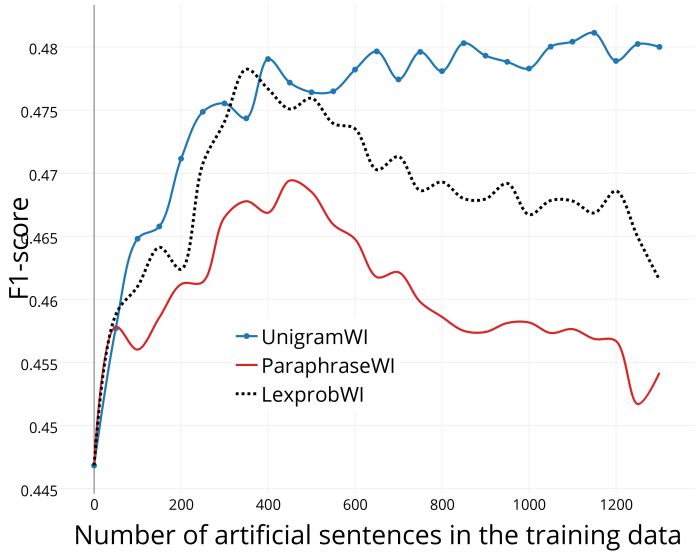

Fig. 4.7 Ternary classification: performance of different word inserters.

Surprisingly, the best word insertion strategy is the simplest one, namely **unigramWI**, which chooses words according to their frequency in a corpus. The two strategies based on lexical translation probabilities perform worse. This can be explained by the fact that random selection tends to choose frequent words more often, whereas lexical translation tables contain many low-probability translations and possibly noise. Therefore, the words selected using them often do not suit the sentences and only hamper QE performance. The **paraphraseWI** performs even worse than **lexprobWI**, because it takes the data from two lexical translation tables, which increases the probability of adding noise to the data. The comparison of word insertion strategies is plotted in figure 4.7.

However, adding only negative data makes the distribution of classes in the training data less similar to that of the test set, which might affect performance negatively. Therefore, we conducted other three sets of experiments: we added (i) equal amount of artificial data for the "GOOD" and "BAD" classes (ii) batches of artificial data for all classes that keep the original proportion of classes in the data (iii) artificial data for only the "GOOD" class. The latter setting is tested in order to check whether the classifier benefits from negative instances, or just from having new data added to the training sets. The "GOOD" sentences were taken from the corpus used for data generation, and "ALMOST GOOD" sentences were produced by our generation methods.

The results are shown in figure 4.8. We plot only the results for the **bigramEG + unigramWI** setting as it achieved the best result in absolute values, but the trends are the same for all data generation techniques. The best strategy was to add both "GOOD" and "BAD" sentences: it beats the models which uses only negative examples, but after 1,000

artificial sentences its performance degrades. Keeping the original distribution of classes (setting denoted as "All classes" in the figure) is not beneficial for this task: it performs worse than any other tested scenario since it decreases the F_1-score for the "GOOD" class dramatically.

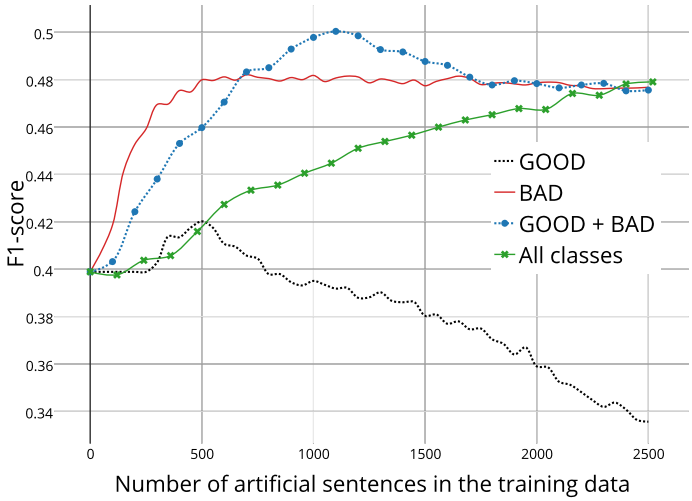

Fig. 4.8 Ternary classification: artificial examples of different classes.

Overall, the additional negative training data improves the ternary sentence classification. The addition of both positive and negative examples can further improve the results, while providing additional instances of the "ALMOST GOOD" class did not seem to be as helpful.

Also, as it was already discussed, the datasets formed by **lexprobWI** have less injected errors. It means that we need to generate more data in order to get the sufficient number of artificial negative examples. In this experiment, we could not test the **crfEG + lexprobWI** combination, because the number of errors in datasets generated in this setting was too small, and we were not able to find the sufficient number of sentences which could be classified as "BAD" (i.e. those containing 4 or more errors).

This problem also held for all datasets that used **crfEG** in conjunction with other word inserters: only 3–4% of data generated with CRF-based methods contained enough errors to be used as an instance of "BAD" class. Hence, although CRF-based methods are slightly better for generating the negative data for this task, the fact that they insert too few errors makes them impractical.

Sentence-level HTER QE task

The prediction of HTER can be more naturally modelled as a regression task, so we addressed it using the SVM regression. The results were evaluated in terms of Mean Absolute Error (MAE).

The addition of any type of artificial data leads to substantial improvements in quality for this task. The initial training dataset was very small – 896 sentences (200 sentences for test), which may explain the substantial improvements in prediction quality as new data is added.

The performance of systems depends both on error generators and word inserters used. When using **bigramEG** and **wordprobEG** we noticed, unlike the results of ternary classification task, that **lexprobWI** is the best-performing word inserter. **unigramWI** is the second best, and **paraphraseWI** has the lowest performance. However, this does not hold for **crfEG** — its combinations with all word inserters create datasets of similar quality (see figure 4.9).

Therefore, the best strategy of word selection for this task is **lexprobWI**. We compare different error generators in conjunction with **lexprobWI** in figure 4.10. The addition of data from datasets generated with **crfEG** gives the largest drop in MAE (from 0.161 to 0.138). This result is achieved by a model that uses 750 artificial sentences. Further addition of new data harms performance. The data generated by other error generators does not cause such a sharp improvement, however, it results in steady reduction of error and performs better than **crfEG** as we add more than 1,500 artificial sentences. The reason of good performance of the **crfEG** and **lexprobWI** might be that the distributions of scores in the baseline training and test sets are different: the test set has lower average score (0.26 compared to 0.31 in the training set) and lower variance (0.03 versus 0.05 in the training set). The use of artificial data with a small number of errors changes this distribution.

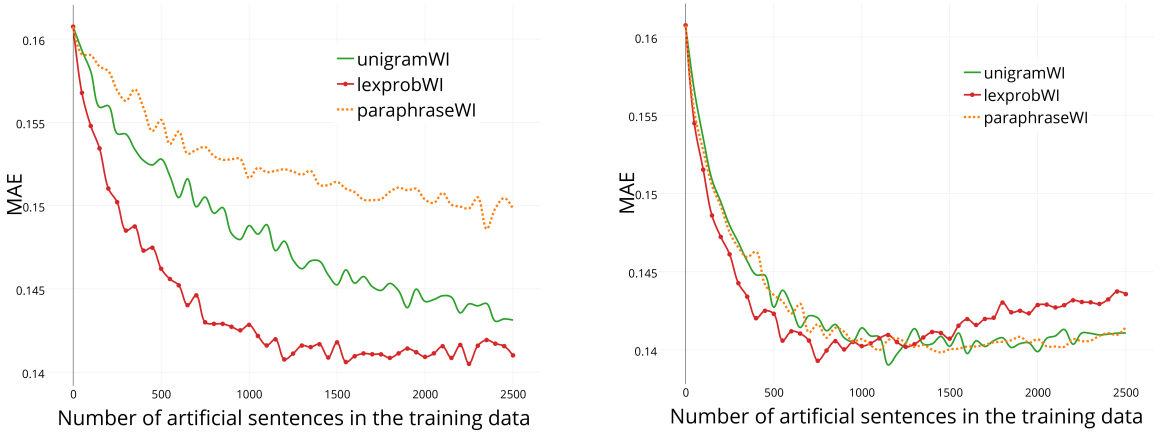

Fig. 4.9 HTER prediction: performance of different word inserters with bigramEG (left) and crfEG (right).

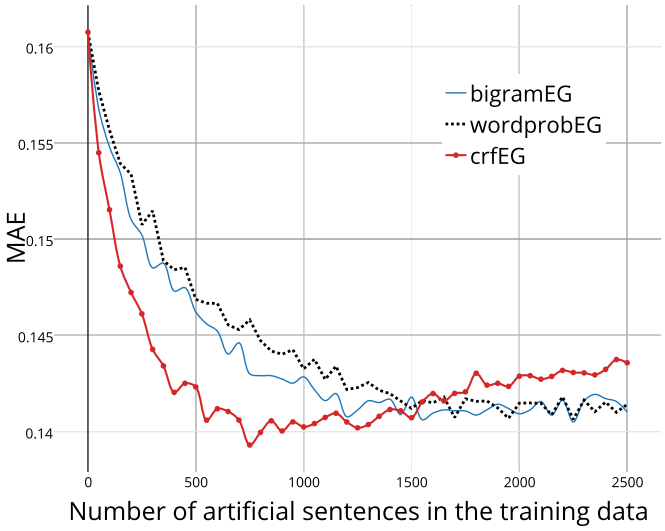

Fig. 4.10 HTER prediction: best-performing datasets (different error generators + lexprobWI)

We also experimented with training a model using only artificial data. The results of models trained on only 100 artificial sentences for each generation method were surprisingly good: their MAE ranged from 0.149 to 0.158 (compared to the baseline result of 0.161 on the original data). However, the further addition of new artificial sentences did not lead to improvements. Thus, despite the positive impact of the artificial data on the results, the models cannot be further improved without real training examples.

Word-level QE task

Here we tested the impact of the artificial data on the task of classifying individual words as "OK" or "BAD". The baseline set contains 47,335 words, 35% of which have the tag "BAD". The test set has 9,613 words with the same label distribution.

All the datasets led to similar results. Overall, the addition of artificial data harms prediction performance: the F_1-score for the "BAD" class goes down until 1,500 sentences are added, and then levels off. The performance for all datasets is similar. However, analogously to the previous tasks, there are differences between **crfEG** and the other two error generation techniques: the former leads to faster deterioration of F_1-score. No differences were observed among the word insertion techniques tested.

Figure 4.11 shows the weighted average F_1-score and F_1-scores for both classes. Since all datasets behave similarly, we show the results for two of them that demonstrate slightly different performance: **crfEG+unigramWI** is shown with solid blue lines, **bigramEG+unigramWI** — with dotted red lines. The use of data generated with CRF-based methods results in slightly faster decline in performance than the use of data generated with **bigramEG** or **word-**

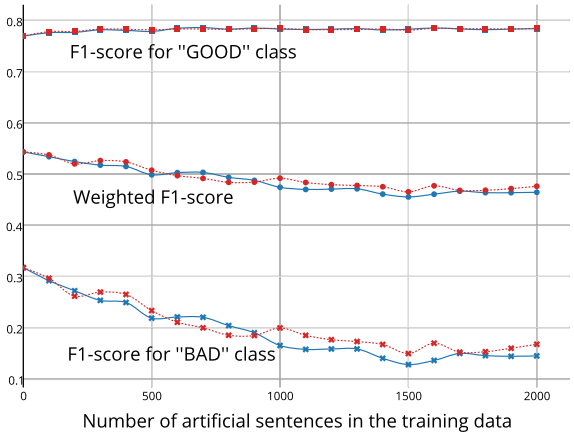

Fig. 4.11 Word-level QE. Blue solid lines – results for **crfEG**, red dotted lines – **bigramEG**

probEG. One possible reason is that the CRF-generated datasets have fewer errors, hence they have different distributions than the original tags in the training data. Therefore, test instances are tagged as "BAD" less often. That explains why the F_1-score of the "BAD" class decreases, whereas the F_1-score of the "OK" class stays at the same.

To summarise our findings for word-level QE, the strategies of data generation proposed and tested thus far do not lead to improvements. The word-level predictions are more sensitive to individual words in training sentences, so the replacement of tokens with random words may confuse the model. Therefore, we suggest that the word-level task needs more elaborate methods for substituting words.

4.3 Conclusions

In this chapter we reported on strategies for the improvement of QE models performance via the manipulations of datasets they are trained on: we experimented with filtering them and generating artificial examples from the original dataset. Some of those methods proved successful in certain conditions.

We also presented and experimented with a set of new methods of simulation of errors made by MT systems. Sentences with artificially added errors were used as training data in models that predict the quality of sentences or words. The addition of artificial data can help improve the output of sentence-level QE models, with substantial improvements in HTER score prediction and moderate improvement in sentence classification into "GOOD", "ALMOST GOOD" and "BAD". One of the limitations of our current approach is that the CRF models failed to generate sentences with the sufficient number of errors. To improve that, the model could be enriched with new features, the training error-labelled sentences

could be filtered to include only examples with enough big number of errors. Also, other sequence labelling algorithm (e.g. HMM) might be more appropriate for the task.

Another problem is the failure of our methods to model phrase substitutions: each word is substituted independently of others, whereas several adjacent errors have a high probability to be related.

The data generation can also be made more task-specific: instead of generating the training examples one can generate a set of features. If all the features used for training are numerical, a small set of real negative examples can be used to define the distributions of the feature values, and new examples can be generated using these distributions.

However, the artificial generation of the data failed to improve the predictions of word-level QE models. Therefore, in order to raise the quality of predictions we also need some conceptual changes in QE models. In the next chapter we present a new view on subsentential QE which can help enhance its accuracy and usefulness.

Chapter 5

Subsentence-level Quality estimation

This chapter contains a description of our work on the improvement of QE at subsentential level. First (section 5.1) we focus on the following conceptual improvement: we develop QE models for a new level of granularity — the level of phrases. The motivation for moving away from individual words to phrases is the fact that we aim to use predictions to improve MT, and in particular one type of MT, phrase-based SMT. There, a minimal unit of translation is a phrase (which is defined as any sequence of words) and not a word: at each step the decoder chooses a whole phrase (and not a single word) out of possible list of candidate translations. Therefore, if one word in a chosen phrase is incorrect, the entire phrase should be discarded in order to improve translation quality. We expected that operating at the same level as the decoder will allow a QE model to yield more useful predictions. Such phrase-level predictions can be more useful at decoding time: if we learn to estimate the quality of a chosen phrase on the fly, it can be scored lower and possibly rejected during decoding to improve the final translation.

Another motivation for considering phrases as atomic units is the nature of SMT errors. Words in a sentence are interdependent, and so are errors. Words can agree with each other in grammatical parameters (e.g. gender, number, case), and if one of them was translated incorrectly, it causes a chain of wrong choices. As a result, a sentence can have a big number of wrong words (i.e., words that do not match their references), although they all stem from one translation error. However, if we could group all these related words into one phrase, we could train a system that is able to identify them as a single error.

We suggest a number of phrase-level features and experiment with different models. The phrase-level features are now implemented in `Marmot` (section 5.3.2). Besides that, we organised a shared task on phrase-level QE (section 5.3) in order to encourage further research in this field and compare our models with other approaches.

Another important question which we address is the evaluation of word-level and phrase-level QE models (section 5.2): without knowing which model performs better we cannot effectively use QE in downstream applications. Many existing metrics are biased and do not reflect the real performance of models. While word-level QE is usually evaluated with F_1-score for the "BAD" class, this metric tends to overreward models which tag the majority of (or even all) words as "BAD", although such labelling is uninformative. We study a number of metrics to find a reasonable alternative for F_1-BAD score.

5.1 Phrase-level Quality Estimation[1]

5.1.1 Motivation for phrase-level quality estimation

As was described before, word-level QE has received little attention for many years because automatic quality metrics which were used to generate quality labels for the training data did not give reliable information on the quality of individual words. Word-level QE has been regaining attention since 2013, when it became a part of the WMT evaluation campaign (Bojar et al., 2013). The post-editing of MT output was used to automatically collect translations annotated for quality at the word level: a word left unchanged by a translator was labelled as "OK", while a word edited was labelled as "BAD". However, framing the subsentence-level QE task in this way has serious limitations. Notably, the fact that errors in different words are not independent from one another. For example, if two words agree in their grammatical features, changing one of them will most likely cause the need to change the other one as well.

Let us consider a more concrete example. We have an English phrase *A beautiful flower* which was incorrectly translated into French as *Un bel arbre*. Post-editor will correct this phrase into the phrase *Une belle fleur*, because *arbre* is translated from French as *tree*, not *flower*. If we count the number of errors as the number of corrected words, this phrase contains three errors. The word *arbre* is an example of a mistranslation error, whereas the words *un* and *bel* are semantically correct, but they are in the incorrect form: masculine instead of feminine. The wrong forms of a determiner and an adjective were chosen to fit the incorrect noun *arbre* they need to agree with. Therefore, we can assume that if the noun in this phrase was chosen correctly, the MT system would have picked the correct forms of the dependant determiner and adjective. So these three errors are related, and ideally they should be marked as one error which spans across three words.

[1] A part of this work was published in (Logacheva and Specia, 2015a).

Such groups of related edits were defined by Blain et al. (2011) as post-editing actions (PEAs) — minimal units that should be post-edited jointly in one action according to some pattern. The MQM (Multidimensional Quality Metrics) framework (Lommel et al., 2014) for translation error analysis also focuses on defining errors that can span phrases of any length. This leads us to the idea that QE should be done at the level of phrases, as opposed to words. Analysing groups of words jointly can provide additional information which is not available at the word level, and notifying a user that the errors in several adjacent words are related can help them use quality predictions more efficiently.

Another motivation for phrase-level QE is the fact that the most widely used MT engines are phrase-based, i.e. at each step the MT decoder extends the translation hypothesis with a phrase. In other words, decisions are made over phrases, rather than over single words. Therefore, it is likely that translation errors are generated at the phrase level. In addition, phrase-level QE models could be used to guide decoding to avoid certain errors. Let us assume we have a QE model which can predict the quality of every word in every hypothesis during decoding. Then after choosing a new candidate phrase the oracle estimates the quality of every word in it, and if at least one of words is classified as "BAD" (unsuitable for the context), then the whole phrase will be scored low. Therefore, a more natural way of framing the QE task is at the phrase level.

Previous work on word-level QE has highlighted the intuition that errors can span over entire phrases. Bach (2011) use a number of features that rely on the source phrase that generated the current target phrase. In the work of Ueffing and Ney (2005b), the word posterior probability is computed at the phrase level: it is regarded as the probability of a word being generated by a source phrase rather than by the entire source sentence. However, in previous research the quality labels are defined for every word, and thus our work represents the first effort to estimate the quality of a target phrase as an atomic unit. We identify the main challenges in this task and suggest ways of dealing with them.

5.1.2 Challenges of phrase-level QE

Various problems make phrase-level QE a difficult task to tackle. The main one is that we do not have the **information on the phrase borders** of an arbitrary machine-translated sentence. We do not know what should be considered as a phrase in this case. The goal is to improve an SMT system by guiding the decoder, so we should use phrases in the decoder sense, but the linguistic definition of a phrase as a sequence of syntactically related words will make more sense for the task.

Another problem is the lack of resources for **phrase-level labelling**. The majority of datasets for automatic translations labelled for quality provide only word-level labels, none

of them considers phrase as a minimal unit. Although there exist datasets labelled for errors at the phrase level (e.g. using the MQM framework (Bojar et al., 2014), they do not provide a segmentation that can be used directly for the task. Only erroneous sequences of words are labelled in such datasets. They can be used as "BAD" phrases without changes. However, in practice, only a small fraction of words are erroneous. So if all the unlabelled words are considered "OK", many of the sequences of "OK" words can be very long. If we train a classifier based on such data to discriminate between good and bad phrases, it is very likely to be biased by a phrase length and to classify shorter phrases as bad and longer phrases as good regardless of their actual quality. In addition, if the phrase segmentation is done based on the reference labels, we have no way of segmenting unseen data — the test data to evaluate the model's performance.

And finally, there are no established **features** we can use to represent the quality of phrases. Phrases combine properties of sentences and words: like sentences, they may contain different number of words. On the other hand, phrases should be much shorter than sentences, so many sentence-level features will not apply.

Therefore, training of a phrase-level model involves three sub-tasks: we should segment sentences into phrases, retrieve phrase-level labels, find a suitable feature set and a learning algorithm for this data.

5.1.3 Segmentation and labelling

Phrase-level QE relies heavily upon appropriate sentence segmentation. One of the main difficulties involved in the segmentation task is the lack of a strict definition of what a *phrase* is for this purpose. In linguistics, *phrase* is a unit where words are connected by dependency relationships. In statistical MT, phrases are simply sequences of words that frequently co-occur and are translated jointly into a corresponding sequence of words in the target language. An example of such phrase segmentation and translations of phrases is shown in table 5.1. Note that these MT-defined phrases can violate some constraints imposed on linguistic phrases. For example, words in them are not necessarily connected by a syntactic relation, on the other hand, syntactic relations can be broken by a phrase border (e.g. see the phrase 3 in the table 5.1, where the word *aus* appears in the target phrase, although it is a part of a verb *auswählen* and should be a part of the first phrase). Such phrases also contain punctuation, because punctuation marks are considered as ordinary tokens by most of SMT systems. Translations of such phrases are not necessarily accurate translations *per se* (e.g. the word *aus* in the phrase 3 is not a part of a translation of the original phrase "*, and*"), but translations which are appropriate within a particular context.

Select one or more	‖	characters	‖	, and	‖	choose	Select >	‖	All
⇓		⇓		⇓		⇓			⇓	
Wählen Sie ein oder mehrere	‖	Zeichen	‖	aus , und	‖	wählen Sie	Auswählen >	‖	Alle

Table 5.1 Correspondence of source and target phrases defined by an SMT system. Borders between phrases are denoted with "‖".

Given the prevalence of statistical MT systems nowadays, for this work we assume the notion of phrase used in SMT and use the segmentation produced by a statistical MT decoder. Since we do not always have access to the MT system that produced the translations, we may need to resort to strategies such as re-decoding the source data with a statistical MT system and reproducing its phrase segmentation. We are not guaranteed that this segmentation will match the original one, i.e., the one that generated the target data. However, if the two MT systems are very similar, we should get similar segments. We suggest two ways of segmenting sentences into `Moses`-like phrases: segmentation of both source and target sentences jointly with a source-target MT system, and independent segmentation of target sentences.

Source segmentation

The datasets we use for QE models training have source sentences and their automatic translations. When we have access to the MT system which generated the translation, we can reproduce the original segmentation accurately by simply re-translating the source sentences. However, such MT models are rarely made available, and we are not guaranteed to get the same output using another MT system, even if it trained on the same data.

One possible solution is to constrain the decoder to use only phrases that appear in the target sentence. In order to do this we need to provide source sentence which needs translation and target sentence which should be generated by the MT system. Then the decoder will search its phrase table for source phrases which have only substrings of the provided target sentence as translations. It should then construct translation from phrases whose concatenation matches the given reference. In this case the aim of the MT system is not to generate translation (because we already have it) but to segment both source and target sentence into corresponding phrases, i.e. to find source and target substrings which are translations of each other. Thus, we input a pair of sentences, and get the two sets of aligned phrases, as shown in table 5.1.

Constrained decoding can be performed in `Moses` system. However, constrained decoding is often unable to fully reach the translation provided, usually because of lack of suitable

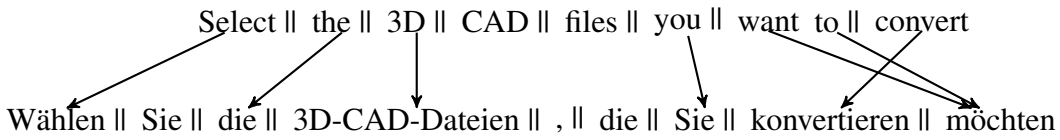

Fig. 5.1 Phrase segmentation where every target word is considered a separate phrase, and source parts of phrases are restored based on alignments of source and target sentences.

phrases in the phrase table. In order to supply the system with this information we extracted an additional phrase table for the data to be decoded (i.e. phrase-level QE data), and produced translations using both phrase tables. In other words, we trained a small translation model on our QE dataset, and then used it to translate this QE dataset, hereby making sure that all words that we need to translate occur in the translation model.

Despite this additional data-specific phrase table, a small percentage of sentences still could not be decoded. In those cases we considered each word of the sentence a separate phrase, and the source part of such phrase was constructed from all source words aligned to the target word. Therefore, for some "phrases" of such sentences, the source phrase will be empty. This happens because some target words are not aligned to any source word. Analogously, since some source words are not aligned to any target word, the source phrases are not guaranteed to cover all source sentence.

Figure 5.1 shows an example of phrase segmentation of a sentence for which constrained decoding could not retrieve the translation. Here every target word is considered a separate phrase, and source parts of phrases are restored based on alignments of source and target sentences. Every target phrase consists of one word, because we did not have any phrase segmentation for this sentence, and had to assume that every word constitutes a phrase. The only two-word phrase *want to* occurs in the source, because the target word *möchten* is aligned with two words.

The alignments were retrieved by an external alignment model, that is why they are incomplete: there are unaligned words in both source and target sentences. Unaligned target tokens ",", *Sie* and *die* form separate phrases, and source parts of these phrases will be empty because the words have no correspondence in the source sentence. Unaligned source tokens *CAD* and *files* are not included to any phrase, because are not aligned to any target word.

However, the cases when we have to fall back to one-word phrases are not common: they occur in at most 10% sentences. In the majority of cases this strategy allows to obtain matching phrase segmentations for the source and the target, and these segmentations cover all words in the sentence pair.

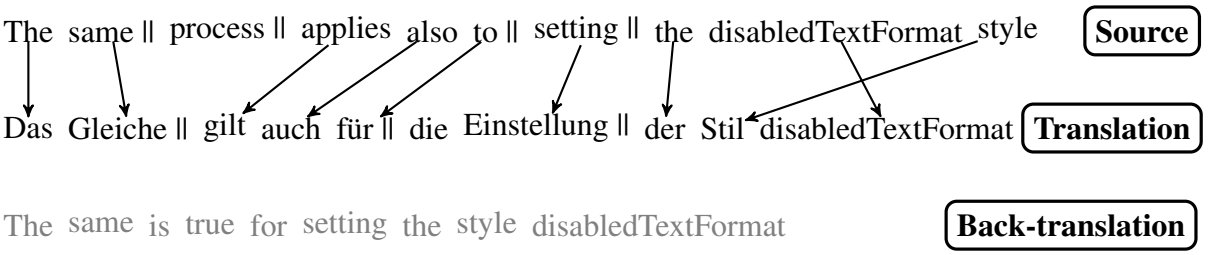

Fig. 5.2 Target sentence is translated with a target-source MT system, its segmentation is kept. The source segmentation is acquired by combining the segmentation of the target sentence with source-target alignments.

Target segmentation

An alternative technique consists in segmenting only the target sentence with an MT system which translates from the target language into the source language. We translate the target sentence with no constraints and retrieve the phrase segmentation for it. The actual translation will not match the source side of our data, which is not an issue as we will discard this translation. We obtain the source segmentation by combining the target segmentation and source-target word alignments: for each target phrase, the corresponding source phrase is composed of all source words aligned to the words in the target phrase.

The figure 5.2 shows an example of such segmentation. Initially we have two sentences: the source and the translation. We translate the translation back into the source language and get back-translation which is discarded. The only outcome of this back-translation which is useful for us is the phrase segmentation of the initial translation. Then we align the source and the translation with a third-party alignment tool. Finally, we retrieve source phrases from the alignments and the target phrases. For each target phrase take all source words that are aligned to words of this target phrase, and form a source phrase out of them. Thus, we get the phrase *the same* for the target phrase *das Gleiche*, the phrase *applies also to* which corresponds to the target phrase *gilt auch für*, etc.

Note that this segmentation is not guaranteed to cover all source words: they are not included to any phrase if they are not aligned to any target words (see the word *process* in figure 5.2). On the other hand, a source word can be included in several phrases in case of multiple alignments.

However, despite its drawbacks, this approach has advantages over **source segmentation** technique described before. Constrained decoding performed with a phrase table trained on the same data is likely to produce too long phrases. Conversely, when segmenting only the target sentence, we can get shorter and more natural phrases.

Phrase labelling

After having segmented sentences into phrases we need to label these phrases for quality. As we have no datasets manually labelled for quality at the phrase level, we need to retrieve phrase-level labelling from the word-level one.

Datasets with post-edited machine translations can be labelled at the word level by comparing the automatic translations with its post-edited version. This can be done with edit distance metrics such as HTER implemented in the `TERCOM` tool (Snover et al., 2006). This tool identifies an edit operation (substitution, deletion, insertion, shift) which needs to be performed on a word to make the automatic translation match its post-edit. The word labels could thus be the edit operations which need to be performed on words to improve the sentence translation, as in the dataset created for the WMT-13 QE shared task (Bojar et al., 2013). Other datasets have incorrect words manually labelled with fine-grained error classes (grammatical error, mistranslation, etc.) (Bojar et al., 2014). Since the number of errors is relatively small (10-30% for different datasets), in order to reduce sparsity, binary ("OK"/"BAD") labels are often used (Bojar et al., 2014, 2015). They indicate simply whether a word fits the context or needs to be edited. However, both these types of labels are defined over words only. When segmenting a sentence with one of the techniques described above, we are likely to face a situation where words put together in a phrase have different tags. Thus we need to combine word labels to get to a single phrase label.

The most obvious combination strategy is *majority labelling*, i.e. to assign the most common label of the words in the phrase to that phrase. However, such a strategy is likely to further increase the discrepancies between the number of occurrences of "BAD" and "OK" labels. The majority tagging strategy can reduce even more the number of "BAD" tags, which will in turn make learning harder. We propose three alternative labelling strategies to mitigate this issue:

- optimistic — if more than half of words have the label "BAD", the phrase has the label "BAD" (majority labelling),
- pessimistic — if 30% words or more have the label "BAD", the phrase has the label "BAD",
- super-pessimistic — if any word in the phrase has the label "BAD", the whole phrase has the label "BAD".

The latter strategy is motivated by the possibility of using phrase-level QE to support phrase-based MT decoding. At each step of the search process the decoder chooses a new phrase, and the best candidate phrase should contain only "OK" words. If one of the words does not fit into the context, the entire phrase should be considered unsuitable.

We deliberately do not use any additional information (e.g. linguistic features of words such as POS-tags) for phrase labelling. The reason for that is that we tried to keep the approach close to vanilla SMT scenario that uses unlabelled parallel data. Although linguistic information can in principle be included into SMT pipeline, it might be inaccessible for some languages. Conditioning phrase labelling on the availability of linguistic markup would make the phrase-level QE less universally applicable. However, we assume that POS-tags can improve phrase labelling if phrase labels are assigned based on the importance of "BAD" and "OK" words that constitute them: e.g. labels of content words should have larger weight than those of function words.

Joint target+data segmentation

Instead of changing the word-level labels, we can get rid of phrases with ambiguous tags if we combine the phrase borders identified by the decoder with the borders of "OK" and "BAD" spans in our data. Let us consider the following example. The target phrase and its original edit distance-based tagging:

¿	Sabes	lo	que	voy	a	hacer	,	sin	embargo	?
OK	OK	OK	OK	BAD	BAD	BAD	BAD	BAD	BAD	OK

create the following segmentation:

[¿ Sabes lo que]	[voy a hacer , sin embargo]	[?]
OK	BAD	OK

The **target segmentation** procedure for the same sentence returns a different segmentation with ambiguous tags:

[¿ Sabes]	[lo que voy a hacer]	[,]	[sin embargo]	[?]
OK	OK/BAD	BAD	BAD	OK

However, if we combine two sets of borders, we convert one phrase with ambiguous tagging (*lo que voy a hacer* — 2 "OK", 3 "BAD" words) into two unambiguous phrases:

[¿ Sabes]	[lo que]	[voy a hacer]	[,]	[sin embargo]	[?]
OK	OK	BAD	BAD	BAD	OK

Note that we can join the phrase borders with the label span borders only for the target segmentation, because the source segmentation has the corresponding source phrases, which cannot be segmented into "BAD" and "OK" segments.

Evaluation

The evaluation of phrase-level QE models is an open question. On one hand, as a new level of granularity it needs a new evaluation procedure. Since the phrase-level QE includes two subtasks — segmentation and labelling — we would like to evaluate both of them. Unfortunately, we have no test set with gold standard phrase segmentation, so we cannot evaluate the correctness of segmentation strategies. As evaluation of labelling is concerned, an obvious solution would be to compute F_1-scores for "BAD" and "OK" classes at the phrase level analogously to word level.

Since we do not have any test sets segmented into phrases, in order to perform phrase-level labelling evaluation we first need to segment the test set. Then we convert its word-level labels to phrase labels using the same schemes as the ones used for training of a model ("optimistic", "pessimistic" or "super-pessimistic"). However, in this case we will be a phrase-level QE model on data which was partially generated by the same QE model, i.e. segmented with one of the segmentation strategies. The test set cannot thus be considered gold standard any more: it will be biased towards the model which generated its labelling. Letting every model perform the test set segmentation itself is not correct either, because the test sets will not be comparable.

Therefore, if we have a test set without segmentation, a more objective way of evaluation seems to be computation of word-level scores. In such cases we segment the test sentences into phrases and label them, and then we propagate the phrase labels onto all words of the phrase. After that the test output can be evaluated at the word level.

5.1.4 Features

Sequence features Some of features used in word-level QE models cannot be applied to phrases. However, most of the sentence-level features are suitable for any sequence of words, not only full sentences. For our experiments we transferred the list of 79 *black-box* sentence-level features to phrases. Some examples of these features are:

- LM features: language model (LM) score of source and target phrases under source and target LMs,
- POS features: numbers of verbs, nouns and other parts of speech in the source and the target phrases,
- Features that indicate the number of tokens from different closed classes: numbers, alphanumeric tokens, punctuation marks in the source and target phrases,
- Average number of translations of source words with different translation probability thresholds as given by source–target alignment,

- Average number of n-grams in different frequency quartiles.

The full set of these features is listed in the section A.3.1 of appendix A.

Vector representation features Another set of features we use relies only on monolingual information, namely vector representations of words generated with `word2vec` tool[2]. `word2vec` assigns every word a fixed-size vector of numbers that encodes information on the contexts in which the word is used. Therefore, similar words should have similar vectors (for a detailed description of `word2vec` see work by Mikolov et al. (2013)). The vectors are word-level, but unlike other word-level features they can be easily combined for phrases that are longer than one word. We can use two vector operations to combine two or more vectors of the same size while keeping the dimensionality of these vectors: element-wise sum or average of the vectors. According to our preliminary experiments, models trained on summed vectors showed higher performance than those that used averaged vectors, so in the experiments reported below we use the sum of vectors.

5.1.5 Training algorithms

Most word-level QE approaches rely on sequence labelling algorithms. One of the best-performing sequence labelling techniques is **Conditional Random Fields** (CRF) (Lafferty et al., 2001), which has been used by many word-level QE models (Camargo de Souza et al., 2014; Luong et al., 2014b). However, a CRF model might be less helpful for phrase-level QE. The errors in words may be dependent on each other, and thus the labels of neighbouring tokens can influence each other. Linear chain CRFs are well suited for modelling this type of dependency. However, in phrase-level QE the relatedness of word-level errors is already captured by the phrases. In other words, if the segmentation is accurate, it encapsulates related errors in one unit. While there are no constraints on labels of adjacent phrases (i.e., two or more OK/BAD phrases can occur consecutively), these labels are not expected to be as closely related as those in word-level QE. Therefore, we also explore a standard classifier, a **Random Forest** classifier (Breiman, 2001), which showed good performance in our previous experiments on word-level QE (see section 4.1.1).

5.1.6 Experiments

We performed a set of experiments to test how phrase-level models compare to previous work on word-level QE and to find the optimal parameters for the phrase-level training. We tested performance varying the following parameters:

[2]https://code.google.com/p/word2vec/

- Segmentation: target segmentation, source segmentation, target+data segmentation;
- Phrases labelling: optimistic, pessimistic or super-pessimistic;
- Feature set: sequence features, combined `word2vec` word vectors, both sets of features;
- Models: CRF or Random Forest.

We conducted our experiments on two datasets used for the QE shared tasks in 2014 and 2015, so we can compare the performance of our models with state-of-the-art results.

Systems

We trained three distinct models on three datasets:

- **phrase-wmt-14**: trained on the WMT-14 dataset labelled with error types (Bojar et al., 2014).
- Two models were trained on fractions of the WMT-15 dataset (Bojar et al., 2015). This dataset has 11,000 post-edited automatic translations. However, the majority of them contain too few errors, and QE models trained on the full dataset tend to perform overly optimistic labellings. Therefore, following our experiments on manipulating the QE training dataset in section 4.1.1 we use only sentences with the highest HTER score (i.e. largest number of errors normalised by the sentence length):
  - **phrase-wmt-15-2000**: trained on the 2,000 worst sentences from WMT-15,
  - **phrase-wmt-15-5000**: trained on the 5,000 worst sentences from WMT-15.

We also compare our models with the following representative models that participated in the WMT-14 and WMT-15 QE shared tasks at the word level:

- Models from the WMT-14 shared task:
  - **Baseline-all-bad** — trivial baseline strategy that assigns the tag "BAD" to all words. No other model could beat it in terms of F_1-BAD score.
  - **FBK-UPV-UEDIN** (Camargo de Souza et al., 2014) — model with features from word posterior probabilities and confusion network descriptors computed over 100,000-best translations. Tagging was done with bidirectional long short-term memory recurrent neural networks. This was the best model in WMT-14.
  - **LIG** (Luong et al., 2014b) — model represented the data with 25 black-box features and was trained with CRF. It was the 3rd best model in WMT-14.
- Models from the WMT-15 shared task:

- **Baseline** (Bojar et al., 2015) — model that was used as a baseline at the WMT-15 word-level QE task.
- **Baseline-all-bad** — the same "all-bad" strategy.
- **UAlacant** (Esplà-Gomis et al., 2015) — model that used features drawn from pseudo-references (automatic translations of the source sentence) generated by different MT systems, and baseline features released for the task. Best best-performing model in WMT-15.
- **Shef-word2vec** (Shah et al., 2015) — model that used word vector representations as features and performed labelling with a CRF model. This model was ranked 3rd.

Tools and datasets

Besides the training and test sets, a QE model requires various resources and tools for feature extraction:

- The word alignment model was trained on the Europarl corpus (Koehn, 2005) using the `fastalign` tool.
- LM and n-gram count features were extracted using trigram LMs trained on the Europarl corpus using `SRILM`[3].
- POS features were extracted with `TreeTagger`.
- The translation probability features were computed using lexical probability tables trained with `Moses` system on the Europarl corpus.
- The word vector representations were computed with `gensim` (Řehůřek and Sojka, 2010) — Python implementation of `word2vec` models. The training data for the vectors is the concatenation of Europarl, News-commentary[4] and News crawl[5] corpora. The vectors are 500-dimensional.

Segmentation properties

The segments produced by two segmentation strategies differ substantially. The main difference is the distribution of phrase lengths: while the **target** segmentation tended to segment the sentences into shorter phrases, the majority of phrases used by the **source** segmentation are 5-word long (see figure 5.3). This is explained by the fact that the former strategy uses an independent translation table, whereas the latter decodes the sentences with a translation table trained on the same sentences, so it contains longer phrases.

[3] http://www.speech.sri.com/projects/srilm/
[4] http://statmt.org/wmt15/training-parallel-nc-v10.tgz
[5] http://statmt.org/wmt14/training-monolingual-news-crawl/

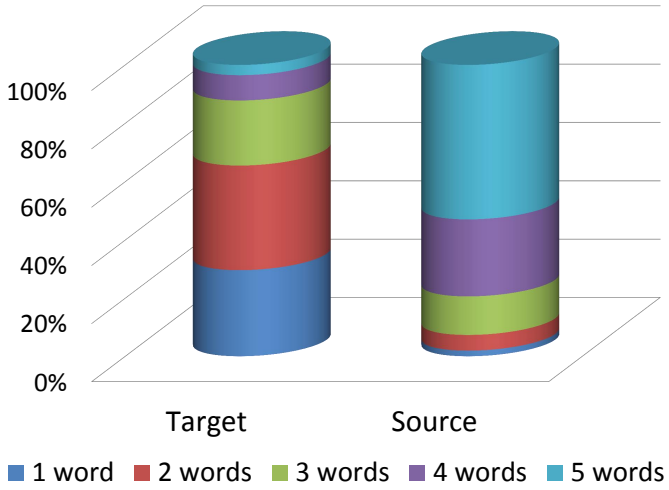

Fig. 5.3 Phrase length frequencies for different segmentation techniques.

We also looked at the amount of word labels that were modified by different labelling strategies under the target-based and source-based segmentation types. Figure 5.4 shows the percentage of words in different datasets that needed to change the label from "OK" to "BAD" and vice-versa. Under source segmentation all labelling techniques become more aggressive, i.e. they change more words. The "optimistic" strategy changes zero or few words from "OK" to "BAD", whereas the "super-pessimistic" strategy does not change words from "BAD" to "OK". Datasets converted with the "pessimistic" strategy contain both types of conversions, but tend to add "BAD" labels rather than "OK" labels.

Selection of optimal parameters

Here we study which parameters we should use to achieve the best prediction quality for our datasets. We found that most of the parameters depend on datasets and values of other parameters. In addition, the performance of a model is difficult to define: as the F_1 score for the "BAD" class (primary metric for the word-level QE task used for models comparison by Bojar et al. (2015)) grows, the F_1 score for the "OK" class drops. In order to account for both of them we plot the F_1-BAD with respect to F_1-OK scores. In each plot we compare models that differ in one parameter. They are usually shown as items of different colours and shapes. Some items of the same configuration can lie quite far apart. That happens because other parameters of a given pair of models influenced their performance.

The performance of models that use different **labelling** schemes follow a certain pattern: the F_1-BAD grows as more negative data is added, while the F_1-OK score drops. Thus, the 'optimistic' labelling scheme is almost always inferior to the other two strategies. The

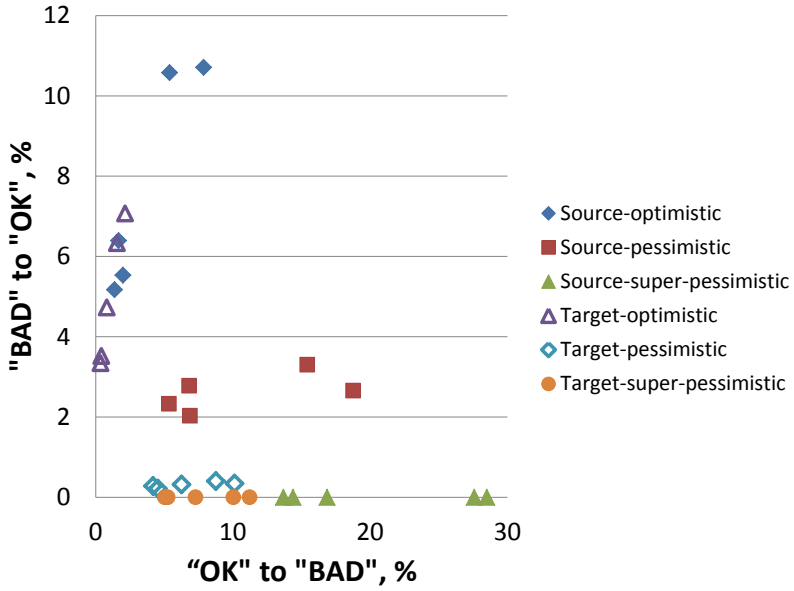

Fig. 5.4 Percentage of word labels modified for datasets segmented with different segmentation techniques (source/target) and re-labelled with either of the labelling strategies (optimistic/pessimistic/super-pessimistic).

‘pessimistic’ and ‘super-pessimistic’ schemes perform closer, but the latter returns higher F_1-BAD scores for most settings (figure 5.5).

This can also be attributed to the source **segmentation** strategy, which generates longer phrases and therefore requires more words to change tag from “OK” to “BAD”. Figure 5.6 shows the comparison of different segmentation strategies and **training algorithms**. It can be seen that CRF produced the best-performing as well as the worst-performing models depending on the type of segmentation: the source-segmented data achieves high F_1-BAD score, whereas target segmentation does not perform well in terms of F_1-BAD. On the other hand, the models trained with the Random Forest classifier do not discriminate between the segmentation types. In addition to that, these models proved very unstable, whereas CRF always returned the same results for a given configuration. In order to get more meaningful results, we ran the Random Forest classifier 20 times for each configuration and averaged the results.

The different sets of **features** do not lead to as much variance in performance as the other parameters. However, we can notice that models with `word2vec` features are more stable and less dependent on other parameters: all models which use these features perform closely. The use of sequence and `word2vec` features in combination can lead to the improved performance, whereas models using only sequence features are the least stable.

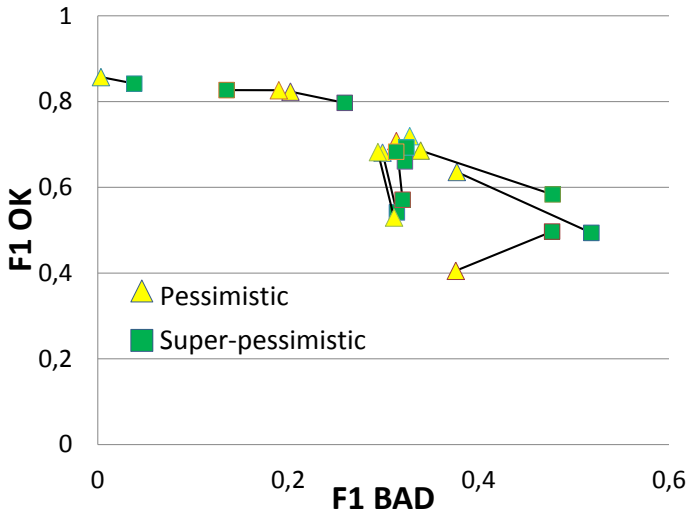

Fig. 5.5 Results for models with pessimistic and super-pessimistic phrase tagging schemes. Results of pairs of models that differ only in terms of tagging strategy are joined with a line.

The settings that returned the highest F_1-BAD scores for all the datasets were similar: **source segmentation**, **super-pessimistic labelling**, models trained with **CRF** (see yellow star in figure 5.6). The optimal feature sets differ for different datasets. All the figures in this section show the performance of models trained on 5,000 sentences from the WMT-15 dataset, but the the trends hold for the rest of the models.

Comparison to word-level models

We trained our phrase-level models on datasets used in the WMT-14 and WMT-15 QE shared tasks, so that we can compare them with word-level models for the task. The WMT-14 models used sequence features, the WMT-15 model with 2,000 sentences — `word2vec` features, the WMT-15 model with 5,000 sentences — the combination of sequence and `word2vec` features (although for both WMT-15 models all feature sets performed closely). The rest of the parameters were fixed for all the datasets: source segmentation, super-pessimistic labelling, CRF.

System	F_1-**BAD**	F_1-OK	Weighted F_1-score
phrase-wmt-14	62.76	39.07	56.80
Baseline-all-bad	52.52	0.0	18.7
FBK-UPV-UEDIN	48.72	69.33	61.99
LIG	44.47	74.09	63.54

Table 5.2 Performance on WMT-14 test set, models are ranked from best to worst with respect to F_1-BAD, our model in bold.

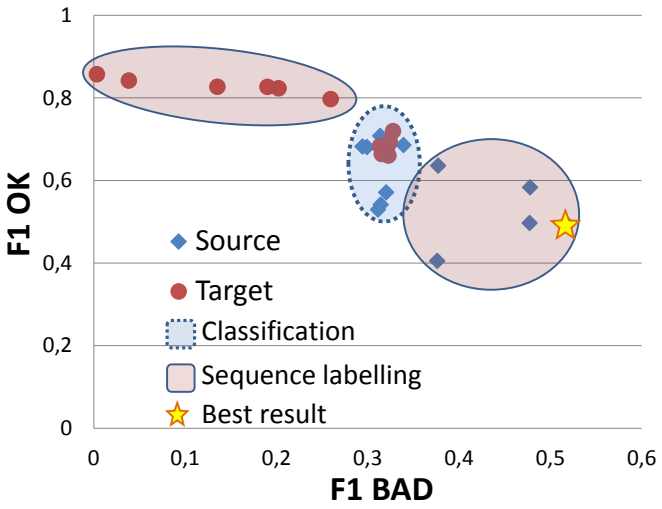

Fig. 5.6 Differences between target and source segmentation and between classification and sequence labelling for phrase-level models.

System	F_1-**BAD**	F_1-OK	Weighted F_1-score
phrase-wmt-15-5000	51.84	49.38	51.08
phrase-wmt-15-2000	51.57	49.05	50.79
UAlacant	43.12	78.07	71.47
SHEF-word2vec	38.43	71.63	65.37
Baseline-all-bad	31.75	0.0	5.99
Baseline	16.78	88.93	75.31

Table 5.3 Performance on WMT-15 test set, models are ranked from best to worst with respect to F_1-BAD, our models in bold.

Table 5.2 shows the performance of models trained and tested on the QE dataset released for the WMT-14 shared task. Our model is the only model which beats the trivial all-bad baseline strategy in terms of F_1-BAD score. The same trend is seen in table 5.3, which shows the performance of models trained on the WMT-15 data. Both our models outperform all other models including the winner. They achieve very close scores, which confirms that sentences with less errors do not contribute much for word-level QE. However, we noticed the low performance of all our models in terms of F_1-OK score. This means that the models are too "pessimistic", in other words, they tend to label too many words as "BAD". This feature of phrase-level models can potentially limit their application in downstream tasks.

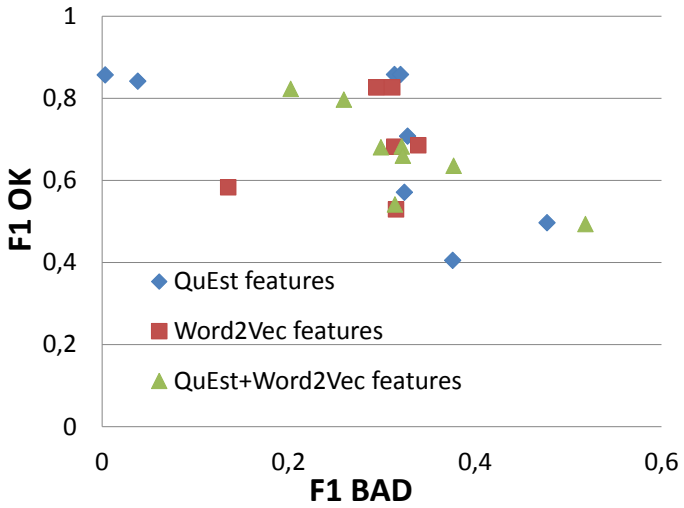

Fig. 5.7 Performance of models with different feature sets.

5.1.7 Discussion

Phrase-level QE of MT is a new field of research. We proposed the first strategies for the task, highlighted some of its challenges. However, the presented experiments could be extended: we only experimented with two ML techniques (CRF and Random Forest), which might be not the best models for the task. We believe that phrase-level QE can benefit from more advanced algorithms that take into account the segmentation of a sentence in subsequences. For example, Semi-Markov CRFs (Sarawagi and Cohen, 2004) are designed to solve segmentation and labelling tasks jointly, and higher order CRFs (Ye et al., 2004) explicitly consider relations between non-adjacent words which can be useful for modelling phrase errors.

An issue with phrase-level QE is that all available datasets are annotated only at the word level. Another direction for future work will thus be the development of a dataset of automatic translations annotated for quality at the level of phrases. The work by Blain et al. (2016) demonstrates an example of such an annotation on a small dataset. From an application perspective, we assume that the phrase segmentation should be guided by segments in statistical MT rather than linguistic properties of the data. However, it would also be interesting to test the usefulness linguistically-informed segmentation.

Besides that, further research is necessary to design features that are specific for phrase-level QE. Phrases combine properties of sentences and words: they are sequences, like sentences, but can be quite short (or even consist of a single word), so sentence-level features may be uninformative. Moreover, a phrase, unlike a sentence, can have the empty source part if the segmentation was performed with the target segmentation technique (section 5.1.3).

This makes all the source-side features harder to apply. The usefulness of linguistically motivated features in particular needs to be tested: as the phrase segmentation performed by an MT decoder does not take into account linguistic information, features indicating whether a phrase is valid based on linguistic information may not suit the task. On the other hand, linguistic information can be useful as it is often unknown to the MT system.

Finally, we noticed the bias of our phrase-level models towards the "BAD" label which results in too "pessimistic" (and therefore less reliable) result and lower F_1-OK score. Although this score is not considered in the QE evaluation campaigns, the big number of "BAD" words can be harmful for further use of phrase-level QE models, therefore, the accuracy of prediction of "OK" labels should also be improved.

5.2 Evaluation of Word-level QE models[6]

The emergence of a large variety of QE approaches requires reliable ways to compare them. The evaluation metrics which are currently used to compare the performance of systems participating in QE shared tasks[7] have received many criticisms.

Word-level QE is commonly framed as a binary task, i.e., the classification of every translated word as "OK" or "BAD". This task is evaluated in terms of F_1-score for the "BAD" class, a metric that favours 'pessimistic' systems — i.e. systems that tend to give the "BAD" label to most words. A trivial baseline strategy that assigns all words the label "BAD" can thus receive a high score while being completely uninformative (Bojar et al., 2014). We saw a similar result when compared our phrase level QE models against word-level QE models (section 5.1.6). There, phrase-level models outperformed word-level ones in terms of F_1-BAD and were deficient in terms of F_1-OK. According to the present rules of word-level QE shared task our phrase-level models would win, although they are strongly biased towards the "BAD" label and usefulness of their predictions is questionable. Such a result shows that F_1-BAD score is not an objective metric for word-level QE, because it gives too high scores to such "pessimistic" models.

However, no analysis of the word-level metrics' performance has been done and no alternative metrics have been proposed that are more reliable than the F_1-BAD score. In this section we compare existing evaluation metrics for word-level QE and suggest two alternative metrics: F_1**-mult** and **Sequence Correlation**. We show that F_1-mult leads to more objective and reliable results.

[6]This is a joint work with Michal Lukasik (Logacheva et al., 2016c). M.Lukasik suggested the design of experiment with synthetic datasets.

[7]http://statmt.org/wmt15/quality-estimation-task.html

5.2.1 Metrics

One of the reasons word-level QE is a challenging problem is the fact that "OK" and "BAD" labels are not equally important: we are generally more interested in finding incorrect words than in assigning a suitable category to every single word. An ideal metric should be oriented towards the recall for the "BAD" class. However, as was shown, F_1-BAD metric, which combines precision and recall for the "BAD" class, gives too high scores to models with prevailing "BAD" labels. So, in order to be useful the metric should not favour "pessimistic" labellings (i.e., all or most words labelled as "BAD"). Below we describe possible alternatives to F_1-BAD score.

F_1-score variants

Word-level F_1-scores. Since F_1-BAD score is too pessimistic, an obvious solution would be to balance it with F_1-score for the "OK" class. However, the widely used weighted average of F_1-scores for the two classes is not suitable as it will be dominated by F_1-OK due to labels imbalance. Instead, we suggest the **multiplication of F_1-scores** for individual classes: it is equal to zero if one of the components is zero, and since all of them are in the interval [0, 1], the overall result will not exceed the value of any of the multipliers.

Phrase-level F_1-scores. One of the features of MT errors is their phrase-level nature. Errors are not independent: one incorrect word can influence the choice of its neighbours. If several adjacent words are tagged as "BAD", they are likely to be part of an error which spans over a phrase.

Therefore, we also evaluate word-level F_1-scores and alternative metrics which are based on correctly identified erroneous or error-free spans of words. The phrase-level F_1-score we use is similar to the one used for the evaluation of named entity recognition (NER) systems (Tjong Kim Sang and De Meulder, 2003). There, precision is the percentage of named entities found by a system that are correct, recall is the percentage of named entities present in the corpus that are found by a system. For the word-level QE task the precision is the percentage of spans of erroneous (or correct) words found by a system.

In order to take into account partially correct phrases (e.g. a 4-word "BAD" phrase where the first word was tagged as "OK" by a system and the rest were correctly tagged as "BAD"), we compute the number of true positives as the sum of percentages of words with correctly predicted tags for every "OK" phrase. The number of true negatives is defined analogously.

Other metrics

Matthews correlation coefficient. MCC (Powers, 2011) was used as a secondary metric in WMT14 word-level QE shared task (Bojar et al., 2014). It is determined as follows:

$$MCC = \frac{TP \times TN + FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

where TP, TN, FP and FN are true positive, true negative, false positive and false negative values, respectively.

This coefficient takes the values in the interval [-1, 1]. If the reference and hypothesis labellings agree on the majority of examples, then the whole number is dominated by the $TP \times TN$ quantity, which gets close to the value of denominator. The more false values the predictor produces, the lower the value of the numerator.

Sequence correlation. A variant of this metric was used as an additional metric in WMT-15 word-level QE shared task (Bojar et al., 2015). Here we report its refined version. The motivation for this metric is the phrase nature of MT errors, which is disregarded by the existing evaluation metrics.

Let us consider an example. If we translate the English phrase *My dear friend* into French, an SMT will most probably return the phrase *Mon cher ami*. However, the correct translation might be *Ma chère amie* if *friend* references a female. Here all the words are incorrect, although the cause of error is the wrong choice of gender of a word *amie*. If a QE system evaluates words *mon* and *ami* as "BAD" and *cher* as "OK", this labelling will receive quite high score under F_1-BAD. Nevertheless, it is clear that the system failed to identify the fact that these errors are related. We designed the sequence correlation metric to penalise such cases, but none of the existing metrics can do that.

Therefore, the metric penalises the hypotheses where the number of error spans differs from that of the reference. In the following example:

```
Reference:  OK BAD BAD BAD OK OK OK
Hypothesis: OK  OK   OK   OK  OK OK OK
```

the reference has three spans: ["OK", "BAD BAD BAD", "OK OK OK"], and the hypothesis has only one span which includes the whole sequence. Instead of the number of spans we compute the number of borders between spans, so that it equals to zero for sequences with one span:

$$brd(\mathbf{x}) = \sum_{i=1}^{n-1} I(x_i \neq x_{i+1})$$

where $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ is a sequence. and the function I returns 1 if its argument is true and 0 otherwise.

The penalty is then defined as the ratio of the number of borders in the hypothesis and the reference:

$$r(\mathbf{y}, \hat{\mathbf{y}}) = \min(\frac{brd(\mathbf{y})}{brd(\hat{\mathbf{y}})}; \frac{brd(\hat{\mathbf{y}})}{brd(\mathbf{y})})$$

where $\mathbf{y} = \{y_1, y_2, ..., y_n\}$ is a true labelling, $\hat{\mathbf{y}} = \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_n\}$ is a hypothesis. This ratio is 1 if the number of spans is equal for the hypothesis and the reference, and less than 1 otherwise.

The sequence correlation metric is based on the accuracy score which is always defined at the sentence level. In order to prevent the score from being dominated by the "OK" class we use a weighted version of the accuracy score which includes coefficients for "BAD" and "OK" labels so that TP and TN contribute at most half of the overall accuracy score:

$$ACC_w(\hat{\mathbf{y}}, \mathbf{y}) = (\lambda_1 TP + \lambda_0 TN)/(P + N)$$

where P and N are number of positive and negative labels in the gold standard, respectively.

If $TP = P$, $\lambda_1 TP = 0.5$, the same holds for $\lambda_0 TN$ if $TN = N$. Therefore, λ_t should be inversely proportional to the number of instances of tag t in the reference:

$$\lambda_t = \frac{\#\ words\ in\ \mathbf{y}}{\#\ words\ with\ tag\ 't'\ in\ \mathbf{y}}$$

The sentence-level score is produced as follows:

$$SeqCor(\mathbf{y}, \hat{\mathbf{y}}) = r(\mathbf{y}, \hat{\mathbf{y}}) \cdot ACC_w(\hat{\mathbf{y}})$$

SeqCor for the whole dataset is in this case defined as the average of sentence-level scores. We can also define a document-level version of the metric by computing ACC_w and r quantities at the level of document:

$$SeqCor_{doc}(\hat{\mathbf{Y}}, \mathbf{Y}) = r_{doc}(\hat{\mathbf{Y}}, \mathbf{Y}) \cdot ACC_w(\hat{\mathbf{Y}}, \mathbf{Y})$$

where r_{doc} is a document-level version of r:

$$r_{doc}(\hat{\mathbf{Y}}, \mathbf{Y}) = \min(\frac{\sum_{\mathbf{y} \in Y} brd(\mathbf{y})}{\sum_{\hat{\mathbf{y}} \in \hat{Y}} brd(\hat{\mathbf{y}})}; \frac{\sum_{\hat{\mathbf{y}} \in \hat{Y}} brd(\hat{\mathbf{y}})}{\sum_{\mathbf{y} \in Y} brd(\mathbf{y})})$$

5.2.2 Metrics comparison

One of the most reliable ways of comparing metrics is to measure their correlation with human judgments. However, for the word-level QE task, asking humans to rate a system labelling or to compare the outputs of two or more QE systems is a very expensive process. A practical way of getting the human judgments is the use of quality labels in downstream human tasks — i.e. tasks where quality labels can be used as additional information and where they can influence human accuracy or speed. One such a downstream task can be computer-assisted translation, where the user translates a sentence using automatic translation as a draft, and word-level quality labels can highlight the incorrect segments of a sentence. Improvements in productivity will show the degree of usefulness of the quality labels. However, such experiment is also very expensive to be performed. Therefore, we look for indirect ways of comparing the metrics' reliability, based on pre-labelled gold-standard test sets. We compare the performance of six metrics:

- F_1**-BAD** — F_1-score for the "BAD" class,
- F_1-**mult** — multiplication of F_1-scores for individual classes,
- **phrase-**F_1**-BAD** — phrase-level F_1-score for the "BAD" class,
- **phrase-**F_1**-multiplied** — multiplication of phrase-level F_1-scores,
- **MCC** — Matthews correlation coefficient,
- **SeqCor** — Sequence Correlation.

Comparison on real systems

One of the purposes of different systems comparison is to identify the best-performing one. So, we expect a good metric to be able to distinguish between systems in the best possible way. As such, one of metrics' quality criteria will be the number of significantly different groups of systems each metric can identify. Another way of metrics evaluation is to compare the real systems' performance with synthethic datasets for which we know the desirable metric behaviour. If a metric gives the expected scores to all artificially generated datasets, it detects some properties of the data which are relevant to us, so we can expect it to work adequately on the real datasets.

We compare the performance of metrics with respect to how well they can group a set of QE systems into groups by statistically significant difference, i.e. how many significantly different groups of systems they can identify. We performed the ranking of all systems submitted to WMT-15 word-level QE shared task. 8 teams submitted 16 QE systems to the task (2 systems per team), word-level baseline was the 17-th system. The official ranking was done according to the systems performance in terms of F_1-BAD which was the primary

metric for the task. It identified 9 significantly different groups of systems. The official results are shown in table 5.4. In addition to the QE systems, we test the performance of the metrics on a number of synthetically created labellings that should be **rated low**:

- **all-bad** — all words are tagged as "BAD".
- **all-good** — all words are tagged as "OK".
- **optimistic** — the majority of words are tagged as "OK", but there exists small number of "BAD" labels: this system should have high precision (0.9) and low recall (0.1) for "BAD" label.
- **pessimistic** — the majority of words are tagged as "BAD": high recall for "BAD" label, low recall for "OK" label.
- **random** — labels are drawn randomly from the label probability distribution.

System ID	weighted F_1 All	F_1 **Bad** ↑	F_1 GOOD
English-Spanish			
• UAlacant/OnLine-SBI-Baseline	71.47	43.12	78.07
• HDCL/QUETCHPLUS	72.56	43.05	79.42
UAlacant/OnLine-SBI	69.54	41.51	76.06
SAU/KERC-CRF	77.44	39.11	86.36
SAU/KERC-SLG-CRF	77.4	38.91	86.35
SHEF2/W2V-BI-2000	65.37	38.43	71.63
SHEF2/W2V-BI-2000-SIM	65.27	38.40	71.52
SHEF1/QuEst++-AROW	62.07	38.36	67.58
UGENT/SCATE-HYBRID	74.28	36.72	83.02
DCU-SHEFF/BASE-NGRAM-2000	67.33	36.60	74.49
HDCL/QUETCH	75.26	35.27	84.56
DCU-SHEFF/BASE-NGRAM-5000	75.09	34.53	84.53
SHEF1/QuEst++-PA	26.25	34.30	24.38
UGENT/SCATE-MBL	74.17	30.56	84.32
RTM-DCU/s5-RTM-GLMd	76.00	23.91	88.12
RTM-DCU/s4-RTM-GLMd	75.88	22.69	88.26
Baseline	75.31	16.78	88.93

Table 5.4 Official results for the WMT-15 word-level QE shared task. The winning submissions are indicated by a •. Submissions whose results are statistically different from others are grouped by a horizontal line.

We rank the systems using each of the metrics and compute the level of significance for every pair of systems with randomisation test (Yeh, 2000) with Bonferroni correction (Abdi, 2007).

In order to evaluate the metrics' performance we compute the system distinction coefficient d — probability of two systems being significantly different (defined as ratio of numbers of significantly different pairs of systems and all pairs of systems). We also compute d for the top half and for the bottom half of the ranked system list separately in order to see how a metric discriminates between better performing and worse performing systems.[8] (synthetic datasets are not considered). Higher d is better because it means that a metric can tell apart larger number of systems. The highest value of d is 1.0 which means that all differences between a metric's values of a given set of systems are statistically significant. However, this value can be infeasible, because some of the systems can be almost indistinguishable (especially when they are variants of one system).

The results are presented in table 5.5. It contains the d, d_{top} and d_{bottom} statistics for every metric. For every synthetic dataset it also shows the number of real system outputs that were rated lower than this dataset; the rightmost column contains the sum across all the synthetic sets. The full result of this experiment (scores, rankings and statistical significance of all systems in terms of all metrics) can be found in the section B.1 of the appendix B.

We can see that three metrics are more robust to synthetic datasets: SeqCor and both multiplied F_1-scores. In the case of SeqCor this is explained by the fact that it favours longer spans of "OK" and "BAD" labels (if the reference labelling also has them) and penalises labellings that insert labels randomly, and multiplications of F_1-scores have two components which are complimentary. This assumption is confirmed by the fact that F_1-BAD scores become too pessimistic without the "OK" component: they both favour synthetic sets with prevailing "BAD" labels, phrase-F_1-BAD even puts them to the top of the systems rank (**all-bad** and **pessimistic** sets outperform 16 out of 17 systems under this metric).

MCC is, in contrast, too 'optimistic': the **optimistic** dataset is rated higher than most of system outputs. In addition to that, it is not good at distinguishing different systems: its system distinction coefficient is the lowest among all metric. SeqCor and phrase-F_1-multiplied, despite identifying artificial datasets, cannot discriminate between real systems: SeqCor fails with the top half systems, phrase-F_1-multiplied is bad at finding differences in the bottom half of the list.

Overall, the F_1-multiplied score is the only metric which performs well both in task of identifying synthetic datasets and in discrimination between real systems, although none of

[8] d_{bottom} is always greater than d_{top} in our experiments, because better-performing systems tend to have closer scores under all metrics and more often are not significantly different. When comparing two metrics, greater d does not imply greater d_{top} and d_{bottom}: we use Bonferroni correction for which the significance level depends on the number of compared values, so a difference which is significant when comparing 8 systems, can become insignificant when comparing 16 systems.

	d	d_{top}	d_{bottom}	all-bad	all-good	optimistic	pessimistic	random	total
F_1-BAD	0.79	**0.61**	**0.81**	4	-	1	4	1	10
F_1-mult	0.81	0.57	0.75	-	-	2	-	2	4
phrase F_1-BAD	**0.86**	**0.61**	0.78	16	-	1	16	-	33
phrase F_1-mult	0.75	0.54	0.47	-	-	1	-	-	1
MCC	0.63	**0.61**	0.34	-	-	15	-	-	15
SeqCor	0.77	0.39	0.75	-	-	1	1	2	4

Table 5.5 Statistics for metrics. Numbers in synthetic dataset columns denote the number of system submissions that were rated lower than the corresponding synthetic dataset.

its d scores is the best. However, F_1-BAD does not fall far behind: it has high values of all d scores and can identify synthetic datasets quite often.

Comparison on synthetic datasets

The experiment described above has a notable drawback: we evaluated metrics on the outputs of systems which had been tuned to maximise F_1-BAD score only, and none of the other metrics. Although the participants were not restricted to use any particular training or tuning algorithms, they were told that the primary metric for the task would be F_1-BAD, so they aimed to make their models perform well in terms of this metric. This means that the system rankings produced by other metrics may be unfairly considered inaccurate.

Therefore, we suggest a more objective metric evaluation procedure which uses only synthetic datasets. These datasets are created in more flexible ways than the synthetic datasets used in the previous experiments. We generate datasets with different proportions of errors, compute the metrics' values and their statistical significance and then compare the metrics' discriminative power. We expect these datasets to simulate real systems. We use the following procedure for synthetic data generation:

- choose the proportion of errors to introduce in the synthetic data,
- collect all sequences that contain incorrect labels from the outputs of real systems,
- randomly choose the sequences from this set until the overall number of errors reaches the threshold,
- take the rest of sentences from the gold standard labelling (so that they contain no errors).

Thus our artificial datasets contain a specific number of errors, and all of them come from real systems. We can generate datasets with very small differences in quality and identify metrics for which this difference is considered significant.

Let us compare the discriminative power of metrics m_1 and m_2. We choose two error number thresholds e_1 and e_2 and sample random datasets for both e_1 and e_2 a relatively small (e.g. 100) number of times. We compute values of both metrics on the two sets of random samples and for each metric we test if the difference between the results for the two sets is significant. We compute the statistical significance using non-paired t-test with Bonferroni correction. Here we do not need the randomisation test any more, because it is necessary to compute statistical significance of complex metrics, such as F_1-score, and here we are comparing the differences of scores.

Since we sampled the synthetic datasets a small number of times it is likely that the metrics will not detect any significant difference between them. In this case we repeat the whole process with increased (e.g. 200) number of samples and compare the p-values for two metrics again. Thus, by gradually increasing the number of samples, at some point we will find that one of the metrics recognises the differences in scores as statistically significant, while the other one may not. This means that the former metric has higher discriminative power: it needs less samples to understand that the systems are different.

The full procedure is outlined in algorithm 1. We compare the two metrics m_1 and m_2 on the two error thresholds e_1 and e_2. We compute the metric scores for N random samples with e_1 and e_2 errors. If at some point one of the metrics identifies significant difference between the samples and another one does not, it means that this metric is more sensitive. In such a case we return this metric. Otherwise, the comparison repeats with the increased N. This process cannot go on a loop forever, because according to our experiments at $N = 10000$ every metric is able to identify a significant difference even for a pair of very close error thresholds.

Result: $m_i \in \{m_1, m_2\}$
$N \leftarrow 100$
$\alpha \leftarrow$ significance level
while $p\text{-}val_{m_1} \geqslant \alpha$ *and* $p\text{-}val_{m_2} \geqslant \alpha$ **do**
  $s_1 \leftarrow N$ random samples with e_1 errors
  $s_2 \leftarrow N$ random samples with e_2 errors
  $p\text{-}val_{m_1} \leftarrow t\text{-}test(m_1(s_1), m_1(s_2))$
  $p\text{-}val_{m_2} \leftarrow t\text{-}test(m_2(s_1), m_2(s_2))$
  if $p\text{-}val_{m_1} < \alpha$ *and* $p\text{-}val_{m_2} \geqslant \alpha$ **then**
    return m_1
  else if $p\text{-}val_{m_1} \geqslant \alpha$ *and* $p\text{-}val_{m_2} < \alpha$ **then**
    return m_2
  else
    $N \leftarrow N + 100$
end

Algorithm 1: Repeated sampling.

In our experiments in order to make the p-values more stable we repeat each sampling round (sampling of a set with e_i errors 100, 200, etc.) for 1,000 times and use the average of p-values. We used fixed sets of sample numbers: [100, 200, 500, 1000, 2000, 5000, 10,000] and error thresholds: [30%, 30.01%, 30.05%, 30.1%, 30.2%]. The significance level α is 0.05.

We need to compare 6 metrics on 5 error thresholds. This gives us 15 pairs of metrics and 10 pairs of error thresholds. Pairwise comparison of metrics is time-consuming, so we analyse the results in the following way: for every difference in the percentage of errors (e.g. thresholds of 30% and 30.01% give 0.01% difference, thresholds of 30% and 30.2% — 0.2% difference) we define the minimum number of samplings that a metric needs to see significant difference between datasets which differ in this number of errors. Table 5.6 shows the result, where numbers in cells are the minimum number of samplings. We do not show error differences greater than 0.2 because all metrics identify them well. Metrics are sorted by discriminative power from best to worst, i.e. metrics at the top of the table require less samplings to tell one synthetic dataset from another.

	0.01	0.04	0.05	0.1	0.15	0.2
F_1-mult	10000	**2000**	**2000**	**500**	**200**	**100**
MCC	10000	**2000**	**2000**	**500**	**200**	**100**
F_1-BAD	10000	5000	**2000**	1000	500	200
phrase-F_1-mult	10000	5000	5000	1000	500	200
SeqCor	10000	5000	5000	1000	500	500
phrase-F_1-BAD	10000	10000	5000	1000	500	500

Table 5.6 Repeated sampling: the minimum number of samplings required to discriminate between samples with a different error amounts.

As in previous experiment, here the discriminative power of the multiplication of F_1-scores is the highest. Surprisingly, MCC performs equally well. Similarly to the experiment on the real systems, the F_1-BAD score is below the F_1-mult score, but here their difference is more salient. All phrase-motivated metrics show worse result.

5.2.3 Discussion

Overall, we found that the multiplication of F_1-BAD and F_1-OK scores is more stable against "pessimistic" labellings and has bigger discriminative power when comparing synthetic datasets. However, no other tested metrics, including advanced phrase-based scores, could still outperform F_1-BAD. We suggest that F_1-mult should be used as a metric for evaluation and comparison of word-level and phrase-level QE models. Despite the metric's stability it is

not easily interpretable: while F_1-BAD and F_1-OK scores are intuitive and give a reader the overall understanding of a QE model's performance, F_1-mult is difficult to interpret on its own. Therefore, we suggest that QE models should be compared in terms of F_1-mult, and F_1-scores for the individual classes should be reported for clarity.

Finally, we should note that these experiments cannot be considered as an ultimate evaluation of metrics efficiency. Instead, it should be considered as a proxy of real user evaluation of word-level and phrase-level QE metrics, which can be done only on downstream task (e.g. computer-assisted translation).

5.3 Phrase-level QE shared task[9]

In order to promote and push work on the new prediction level, we introduced a phrase-level QE shared task in the First Conference on Machine Translation (WMT-16)[10].

5.3.1 Dataset and task setup

The WMT-16 QE dataset contains the English–German data on the IT domain labelled for quality. Each instance of the dataset contains:

- source sentence in English,
- its automatic translation into German performed by a phrase-based SMT model trained on around 3 million sentences of in-domain data,
- post-edit of the automatic translation performed by a professional translator,
- free reference translation of the source into German.

Table 5.7 shows the partition of the dataset into training, development and test sets.

	Sentences	Phrases	% of "BAD" phrases	% of sentences with no errors
Training	12,000	109,921	29.84	22.67
Development	1,000	9,024	30.21	21.95
Test	2,000	16,450	29.53	19.10

Table 5.7 Dataset for WMT-16 shared task on phrase-level QE.

We had the access to the MT model that generated translations, so we could use the information about the phrase segmentation made by the decoder during translation. Therefore, this data contains the real phrase segmentation with one-to-one correspondence of phrases,

[9]The findings of this task were published as a part of findings of WMT-16 (Bojar et al., 2016).
[10]http://statmt.org/wmt16/quality-estimation-task.html

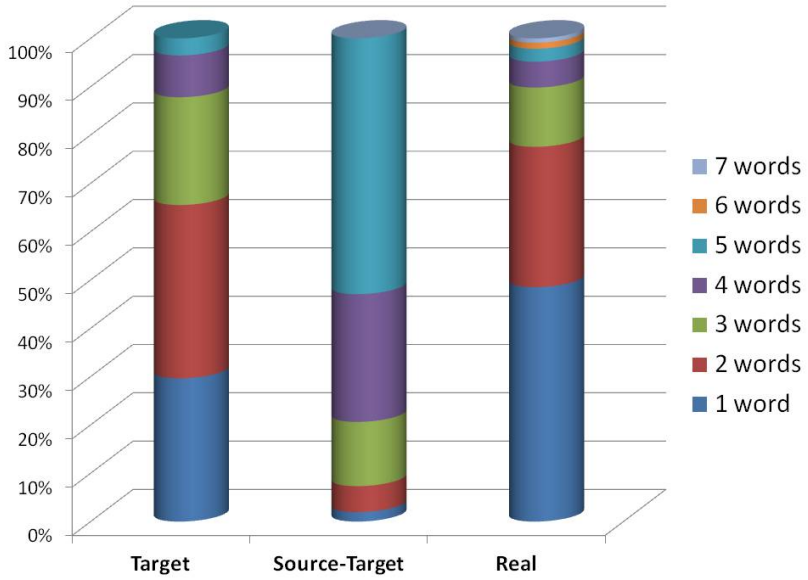

Fig. 5.8 Comparison of real phrase segmentation produced by the decoder and simulated segmentation techniques.

and we did not need to simulate the segmentation for it. The phrase length statistics here is different from the phrases produced by segmentation techniques we suggested in section 5.1.3. The distribution of phrase lengths is similar to the segmentation produced by **target** segmentation technique, but it contains bigger number of short (one-word and two-word) phrases (see figure 5.8 for comparison).

Following our work on the evaluation metrics (section 5.2) we used F_1-mult score as a primary metric for this task. We also report F_1-OK and F_1-BAD scores. The baseline phrase-level model is the most successful model from the ones tested in our initial experiments in phrase-level QE (section 5.1): it is a CRF model that uses sequence features. The baseline features were extracted with the phrase-level extension of `Marmot` (section 5.3.2).

5.3.2 Baseline phrase-level QE model: Marmot implementation

A shared task needs an established baseline model in order to provide a more informed and sound comparison of solutions suggested by participants. Based on the experiments described in the section 5.1 we trained a CRF model which used the sequence features (features originally suggested for representation of sentences and adapted for phrases in our work on phrase-level QE).

The training was performed with `CRFSuite` toolkit, and the features were extracted using the Marmot tool which was extended accordingly[11]. As it has been already described in the section 3.4.2, the majority of components of Marmot do not depend on the content of the objects they work with (i.e. the procedures can work in the same way with words, phrases or longer chunks of text), therefore, we did not need to make any major changes in the code. There were two stages of the pipeline which needed to be altered in order to accommodate phrases instead of words:

- **parsing of input files**. The data labelled at the level of phrases had the features that did not exist in previously used data formats — namely, the segmentation into phrases. Therefore, we added new functionality: a possibility to parse input files with phrase-level labelling and represent training and test examples as phrases as opposed to words, and a possibility to perform segmentation of input sentences into phrases, if phrase-level labelling is not available.
- **feature extraction**. There is a unified procedure of extraction of features from context objects. It does not access the fields of context objects, it only calls feature extractors successively. The list of feature extractors to be called is identified by a user in a configuration file, so the list of features can be modified without changing the code. However, the phrase-level features were not defined in the tool, therefore, we implemented new feature extractors for sequence features.

5.3.3 USFD submission[12]

The experiments described in the section 5.1.6 had many limitations. First of all, the models were trained and tested on datasets which had no real phrase segmentation, and the simulated segmentation produced by MT decoder was not reliable. The data was also labelled only at the level of words, so the phrase-level labels were generated using approximate labelling techniques (we used different threshold values of the percentage of "BAD" words in a phrase to label the whole phrase as "BAD"). Even the word-level labelling itself was not completely trustworthy: it was produced by non-professional translators.

In the QE dataset released for the WMT-16 shared task some of these problems were solved. The translation was edited by professional translators who were native speakers of the target language (German). Although the sentences were also labelled for quality only at the word level, we could rely on the actual phrase segmentation from the decoder. This

[11] https://github.com/qe-team/marmot/tree/phrase_level

[12] This is a part of joint work with Fred Blain (Logacheva et al., 2016a). The work described here was done by V.Logacheva.

reduced the uncertainty: while we still needed to apply the approximate labelling to convert word-level tags to phrase-level tags, the phrases themselves were already given.

New features

The set of black-box features adopted from the sentence-level QE models considers various aspects of a phrase. However, this set does not take into account the surrounding of an individual phrase, because it was originally used to represent sentences which are considered context-independent in the majority of MT models. In order to improve the representation of phrases we use a number of additional features that depend on phrases to the left and right of the considered phrase — they allow us to evaluate how well a phrase fits in the context. Here we list the 12 new features and the values they can take:

- **out-of-vocabulary words (binary)** — we check if the source phrase has words which do not occur in the SMT training corpus. The feature has value **1** if at least one of source words is out-of-vocabulary and **0** otherwise,
- **source/target left context (string)** — last word of the previous source/target phrase,
- **source/target right context (string)** — first word of the next source/target phrase,
- **highest order of ngram that includes the first target word (0 to 5)** — we checked the ngram at the border of the current and previous phrase: the combination of the first target word in the phrase and 1 to 4 words that precede it in the sentence. Let us denote the first word from the phrase $\mathbf{w}_{first}$ and 4-gram from the previous phrase $p_{-4}p_{-3}p_{-2}p_{-1}$. If the whole 5-gram $p_{-4}p_{-3}p_{-2}p_{-1}\mathbf{w}_{first}$ exists in a target LM, the feature value is 5. If it is not in the LM, ngrams of lower order (from $p_{-3}p_{-2}p_{-1}\mathbf{w}_{first}$ to unigram w_{first}) are checked, the feature value is the order of the longest ngram found in the LM. The LM is trained using the target side of the SMT training corpus.
- **highest order of ngram that includes the last target word (0 to 5)** — feature that considers the ngram $\mathbf{w}_{last}p_1p_2p_3p_4$ (where $\mathbf{w}_{last}$ is the last target word of the current phrase and $p_1p_2p_3p_4$ is the opening 4-gram of the next feature) analogously to the previous feature;
- **backoff behavior of first/last ngram (0 to 1)** — backoff behaviour of ngrams $p_{-2}p_{-1}\mathbf{w}_{first}$ and $\mathbf{w}_{last}p_1p_2$. The backoff behaviour was computed as described by Raybaud et al. (2011). This feature takes values from the $[0,1]$ range depending on whether an ngram $w_1w_2w_3$ or its parts exist in an LM. It is defined as follows:

	Feature set	
	Baseline	Extended
lbfgs	0.270	0.331
l2sgd	0.238	**0.357**
ap	0.315	0.355
pa	**0.329**	**0.357**
arow	0.291	0.314

Table 5.8 F_1-mult scores of models trained on baseline and extended feature sets using different optimisation algorithms for `CRFSuite`.

  - 1.0 if $w_1 w_2 w_3$ exists in the LM,
  - 0.8 if $w_1 w_2$ and $w_2 w_3$ exist in the LM,
  - 0.6 if only $w_2 w_3$ exists in the LM,
  - 0.4 if $w_1 w_2$ and w_3 exist in the LM,
  - 0.3 if w_2 and w_3 exist in the LM,
  - 0.2 if only w_3 exists in the LM,
  - 0.1 if all above ngrams are unknown to the LM.

- **named entities in the source/target (binary)** — we check if the source and target phrases have tokens which start with capital letters,

- **part of speech of the source/target left/right context (string)** — we check part of speech of word that precedes or follows the phrase in the sentence.

Some of these features (e.g. highest ngram order, backoff behaviour, contexts) are used because they were shown to be useful in word-level QE (Luong et al., 2013), others were included because we believe they can be relevant for understanding the quality of phrases.

We train a CRF model because it has been shown to be effective in our initial experiments on phrase-level QE as well as in other researchers' work on word-level models. We use the `CRFSuite` toolkit for training. It provides five optimisation algorithms: L-BFGS with L1/L2 regularization (lbfgs), SGD with L2-regularization (l2sgd), Averaged Perceptron (ap), Passive Aggressive (pa), and Adaptive Regularization of Weights (arow). Since these algorithms could perform differently in our task, we tested all of them on both baseline (sequence features) and extended (sequence features plus the 12 features described above) feature sets using the development set. The experiments showed that passive-aggressive algorithm performed best on both feature sets (see table 5.8). We can also see that the new features significantly improve the performance regardless of the used algorithm.

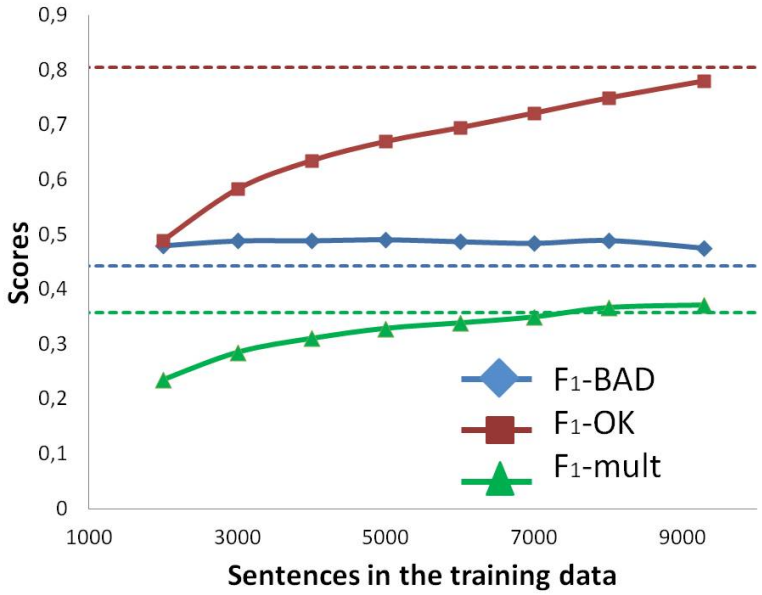

Fig. 5.9 Performance of the phrase-level QE model with different numbers of training sentences. Solid lines show the scores of models with different amount of training data. Dashed lines show the scores of a model that uses the whole training dataset.

Data filtering

As was already discussed in chapter 4, many datasets for word-level QE suffer from uneven distribution of labels: the "BAD" words occur much less often than those labelled as "OK". This also applies to the WMT-16 word-level QE shared task dataset. Since the phrase-level labels for this dataset were generated from the word-level labels, we run into the same problem at the phrase level. Here the discrepancy is not so large: the "BAD" labels make almost 30% of all labels in the training dataset for the phrase-level task. However, we believe it is still useful to reduce this discrepancy. We thus applied the filtering strategy described in the section 4.1.1 to phrase-level QE: we ranked the training sentences by their HTER score (ratio of "BAD" words in a sentence) so that the worst sentences appear at the top of the list, and trained our phrase-level QE model using only N top sentences from the training data (i.e. sentences with more errors).

Figure 5.9 shows how the scores of our phrase-level models change as we add more training data. We examine F_1-scores for both "BAD" and "OK" classes as well as their multiplication, which is the primary metric for the task. The solid lines with markers show how the scores change as we add more sentences to the QE model training data, the flat dotted lines denote the scores of a model that uses the entire dataset (12,000 sentences): red for F_1-OK, blue for F_1-OK, green for F_1-mult. It is clear that F_1-BAD benefits from

filtering out sentences with less errors. The models with reduced data never reach the F_1-OK score of the ones which use the full dataset, but their higher F_1-BAD scores result in overall improvements in performance. The F_1-mult score reaches its maximum when the training set contains only sentences with at least one error (9,280 out of 12,000 sentences), although F_1-BAD score is slightly lower in this case than with a lower number of sentences. Since F_1-mult is our main metric, we use this version of the filtered dataset for the final submission. This model was submitted to the phrase-level QE shared task as the **CONTEXT** model.

5.3.4 Performance of submitted models

Five teams of researchers participated in the shared task. They were allowed to submit up to two systems.

ID	Participating team
CDACM	Centre for Development of Advanced Computing, India (Patel and M, 2016)
POSTECH	Pohang University of Science and Technology, Republic of Korea (Kim and Lee, 2016)
RTM	Referential Translation Machines, Turkey (Bicici, 2016)
UAlacant	University of Alicante, Spain (Esplà-Gomis et al., 2016)
USFD	University of Sheffield, UK (Logacheva et al., 2016a)[13]

Table 5.9 Participants in the WMT-16 quality estimation shared task.

The performance of models was measured with the F_1-mult score as a primary metric, F_1-OK and F_1-BAD score are also reported for better understanding of models' strengths and weaknesses. We test the significance of the results using randomisation tests Yeh (2000) with Bonferroni correction Abdi (2007).

The results of the phrase-level task are presented in table 5.10. Here we cannot identify a single winner: although the F_1-mult scores of the top five systems range from 0.379 to 0.364, this difference was shown to be insignificant. However, all the winning submissions outperform the baseline.

We provide the F_1-BAD and F_1-OK scores in order to better understand the differences between the models. We can see that some models have very close F_1-mult scores, although the components may differ. For example, the F_1-mult scores of two submissions of USFD team are very close (0.367 and 0.364). However, if we decompose these scores, we will see that both F_1-BAD and F_1-OK scores of the two models have around 2% of absolute difference: the W&SLP4PT model is more "pessimistic" (i.e. it is better at labelling "BAD"

System ID	F_1-**mult** ↑	F_1-BAD	F_1-OK
English-German			
• CDACM/RNN	0.380	0.503	0.755
• POSTECH/PHR-RNN-QV3	0.378	0.495	0.764
• POSTECH/PHR-RNN-QV2	0.369	0.478	0.772
• USFD/W&SLP4PT	0.368	0.486	0.757
• USFD/CONTEXT	0.365	0.470	0.777
RTM/s5_RTM-GLMd	0.327	0.408	0.802
BASELINE	0.321	0.401	0.800
RTM/s4_RTM-GLMd	0.307	0.377	0.814
Ualacant/SBI-Online-baseline	0.259	0.493	0.526
UAlacant/SBI-Online	0.098	0.459	0.213

Table 5.10 Official results for the WMT16 Quality Estimation Task 2p. The winning submissions are indicated by a •. These are the top-scoring submission and those that are not significantly worse according to approximate randomization tests with 95% confidence intervals. The gray area indicates the submissions whose results are not statistically different from the baseline.

words), while the CONTEXT model identifies the correct words more accurately. Meanwhile, the combinations of these scores show very little difference. The situation is the same with all top 5 submissions: the big differences between F_1-BAD components are levelled off by F_1-OK components, and the values of the F_1-mult are closer than those of F_1-BAD.

This fact suggests that the F_1-mult score might not be an optimal metric for the phrase-level task. While in the phrase-level models phrases of different length are treated in the same way, the word-level metric unfolds each phrase-level label to a set of word-level labels thus giving different importance to phrases of different length.

In order to find a more suitable metric we tested another evaluation strategy. We evaluated the submissions in terms of phrase-level F_1-scores: here all phrases were considered as atomic units regardless of their lengths, and F_1-BAD and F_1-OK were computed as harmonic means of precision and recall for phrase-level of "OK" and "BAD" labels.

Table 5.11 shows the performance of phrase-level QE models measured in terms of multiplication of phrase-level F_1-scores. Except for some changes in the order of models, this ranking is very similar to the official one presented in table 5.10. Here, the order of submissions by POSTECH and CDACM teams is different from the ranking produced with the primary metric, but they are still not significantly different. On the other hand, the USFD team models are no longer best-performing under the phrase-level F_1-score. This evaluation

shows that phrase-level F_1-mult is slightly better at discriminating between models, although the top 3 models are still too similar to find the single best-performing approach.

System ID	F_1-**mult** ↑	F_1-BAD	F_1-OK
English-German			
• POSTECH/PHR-RNN-QV3	0.393	0.518	0.759
• POSTECH/PHR-RNN-QV2	0.388	0.504	0.771
• CDACM/RNN	0.378	0.500	0.756
USFD/CONTEXT	0.364	0.467	0.780
USFD/W&SLP4PT	0.363	0.475	0.764
RTM/s5-RTM-GLMd	0.331	0.413	0.802
BASELINE	0.311	0.389	0.799
RTM/s4-RTM-GLMd	0.306	0.376	0.815
UAlacant/SBI-Online-baseline	0.275	0.502	0.547
UAlacant/SBI-Online	0.146	0.456	0.320

Table 5.11 Results for the WMT16 phrase-level QE task computed in terms of phrase-level F_1-scores. The winning submissions are indicated by a •. The gray area indicates the submissions whose results are not statistically different from the baseline.

The best-performing models are deep neural networks: the Recurrent Neural Network (RNN) from POSTECH team which predicts the phrase-level labels and the CDACM system whose word-level predictions were successfully applied to phrase-level task. Two of the submitted models make use of the baseline feature set: the USFD's CONTEXT system was enhanced with context information, UAlacante experimented with features based on pseudo-reference translations coming from a number of sources.

Several teams tried taking into account the labels of other levels. While the phrase-level submission of CDACM team simply labels the phrase-level test set using word-level predictions, the phrase-level model of UAlacant team uses the probability of each word in a phrase of being labelled as "BAD" along with other external features. Similarly, USFD's W&SLP4PT system uses information on word labels within a phrase as well as information on sentence-level quality.

5.3.5 Comparison to word-level models

Word-level and phrase-level models that participated in the WMT-16 QE shared task are not directly comparable. Although they are evaluated on the same test sentences, and the labels for the test SET come from the same post-edits, they are not identical. The labels for the phrase-level test set were changed in order to comply with the phrase-level training

data. By convention, a phrase is "BAD" if any of its words is "BAD". Thus, we changed the word-level labels so that all words within a "BAD" phrase are also labelled as "BAD".

Nevertheless, we can still compare the word-level and phrase-level submissions, if we change the word-level submissions appropriately. Let us consider that a word-level QE model was used to label phrases for quality. Following the rules mentioned above we will label a pre-segmented phrase as "BAD" if our word-level QE model labelled any of words of this phrase as "BAD". After performing this transformation we can use the phrase-level gold standard to evaluate both phrase-level and (modified) word-level submissions.

While this comparison is an approximation as the submitted word-level models were not trained to predict the quality of phrases, it still allows us to make a rough comparison of word-level and phrase-level QE models. One of the purposes of the phrase-level task was to understand if the subsentence-level QE can benefit from joint labelling of groups of words, and this cross-task comparison is one of the ways of understanding that.

The table 5.12 contains the joint results of the word-level and phrase-level QE tasks. The best-performing system is the winning word-level model. Moreover, the word-level models tend to perform better in this task in general: the top 7 positions in this joint table are occupied by the word-level models. Some of the phrase-level models which performed well turn out to be not better than the word-level baseline system. Presumably, this result means that defining the quality for individual words yields better results in general.

Another observation we can make from this table is the change in the significance level of the results: some of the word-level submissions which were significantly different from the word-level baseline model in the original (word-level) task are no longer different in the phrase-level version. This can shed the light on the difficulties we had with defining the single best phrase-level system: perhaps the lack of significance of differences between the labellings is a characteristic of the phrase-level task itself. Alternatively, as it was already discussed, this can be explained by the fact that F_1-mult score is not a suitable metric for phrase-level QE.

In order to have a closer look at how the phrase-level task relates to the word-level one we can perform a different comparison. As we can see, some of the teams presented their results for both tasks, and the majority of them have similar models for both levels: they tried to adapt their original word-level system to comply with the phrase-level task. We can compare these pairs of systems to see if the adaptation was successful. This is not a strict comparison, because the models, although similar, could not be identical due to natural differences between words and phrases.

System ID	F_1-**mult** ↑
English-German	
• **word** UNBABEL/ensemble	0.517
word UNBABEL/linear	0.487
word UGENT-LT3/SCATE-RF	0.426
word POSTECH/WORD-RNN-QV3	0.399
word UGENT-LT3/SCATE-ENS	0.395
word POSTECH/WORD-RNN-QV2	0.388
word CDACM/RNN	0.381
phrase CDACM/RNN	0.379
phrase POSTECH/PHR-RNN-QV3	0.378
phrase POSTECH/PHR-RNN-QV2	0.369
word UAlacant/SBI-Online-baseline	0.369
phrase USFD/W&SLP4PT	0.367
word SHEF/SHEF-MIME-0.3	0.367
word SHEF/SHEF-MIME-1	0.367
phrase USFD/CONTEXT	0.364
word BASELINE	0.360
word RTM/s5-RTM-GLMd	0.344
phrase RTM/s5-RTM-GLMd	0.327
phrase BASELINE	0.321
word RTM/s4-RTM-GLMd	0.313
phrase RTM/s4-RTM-GLMd	0.307
word UAlacant/SBI-Online	0.290
phrase UAlacant/SBI-Online-baseline	0.259
phrase UAlacant/SBI-Online	0.097

Table 5.12 Comparison of submissions to word-level and phrase-level QE shared tasks in terms of word-level F_1-mult scores computed on test set used for the phrase-level task. Outputs of word-level models are indicated with the word "**word**", while outputs of phrase-level models — with the word "**phrase**". The winning submission is indicated with •. The gray area indicates the models which are not significantly different from the word-level baseline system, the cyan area indicates the models which are not significantly different from the phrase-level baseline.

Table 5.13 outlines the results of this comparison[14]. Here, in order to enable the direct comparison, we adapted the word-level models to phrase-level test set the same way as we did for table 5.12. It can be seen that the performance of word-level models is better than that of the analogous phrase-level models. There are multiple possible reasons for that: most

[14]The submission by CDACM team was not included in the table because the phrase-level submission of the team is the adaptation of word-level predictions to the phrase level. It was performed analogously to our word-level submissions adaptation, therefore it should not be different.

System ID	Word-level	Phrase-level
English-German		
POSTECH/RNN-QV3	0.399	0.378
POSTECH/RNN-QV2	0.388	0.369
RTM/s5-RTM-GLMd	0.344	0.327
RTM/s4-RTM-GLMd	0.313	0.307
Ualacant/SBI-Online-baseline	0.369	0.259
Ualacant/SBI-Online	0.290	0.097

Table 5.13 Comparison of methods' performance in the word-level and phrase-level QE shared tasks. The performance is evaluated in terms of word-level F_1-mult scores computed on test set used for the phrase-level task. The submissions to the word-level task are modified in order to comply with the phrase-level task.

likely the wrong choice of phrase-level features and the inability of the models originally designed for word-level QE to deal effectively with word sequences.

5.4 Conclusions

We presented our work on subsentential QE for MT: quality estimation for segments of a sentence. First, we suggested a new level of granularity for QE, namely, the level of phrases. We do not pose limitations on the notion of *phrase* for this task, but in the experiments presented we focused on phrases in the phrase-based SMT sense: as sequences of adjacent words with no linguistic constraints. We outlined the main challenges of phrase-level QE: adaptation of word-level datasets to the phrase-level task (including the segmentation of sentences into phrases and labelling of phrases based on word-level tags), selection of appropriate features, training algorithms and evaluation metrics. We suggested a number of solutions for each of these problems and compared a number of phrase-level QE models that use different solutions. In addition to that, we described a shared task on phrase-level QE which we organised within the scope of WMT-16, where we could see how other teams tackled the same problems and compare their models against ours.

Besides introducing phrase-level QE, we worked on the improvement of evaluation of subsentence-level QE. Namely, we tried to find an evaluation metric that fixes the drawbacks of the currently used metrics. We performed comparison of evaluation metrics used for word-level QE using outputs of real models and synthetic datasets which are free from biases and have a precise pre-defined number of errors. We saw that the multiplication of F_1-scores for the two classes ("OK" and "BAD") is less biased than other metrics and better at catching

the differences between synthetic datasets. This information was used in the WMT-16 QE shared task on word-level and phrase-level QE, where F_1-multiplied was used as a primary metric.

We suggested a number of improvements of QE: artificially generated training data proved effective in sentence-level QE models (chapter 4), new level of granularity suggested in the current chapter can improve predictions at the subsentence level. The final goal of our work is to incorporate QE into MT. Therefore, in the next chapters we will use QE for the improvement of automatic translation quality. One of the obvious ways of improvement of an MT system is addition of more data. However, the improvement rate can vary significantly depending on the added data, and with careful selection of new training examples a system can achieve large improvement with small amount of data. A QE model might be useful for the task of data selection: it can provide a new view of the data, namely, it can estimate texts from the point of view of the quality of their translation. In the next chapter we report our experiments on the application of QE predictions to Active Learning (i.e. learning with online selection of new data) for MT.

Part III

Applications

Chapter 6

QE-based data selection and active learning

One of the easiest and most efficient ways of improving the quality of an MT system is its enhancement with new training data. For many language pairs, monolingual data in either source or target languages (or both) tends to be abundant. However, parallel data is rarely available in large amounts. Parallel corpora have to be created by having humans translating monolingual content, which is an expensive process. If instead of translating all the available monolingual data we apply a technique which chooses only the most useful sentences to be translated, we can save human effort and produce an MT system with higher quality using less training data. This process is an application of active learning (AL) to MT.

Active learning is a technique for the selection of the most useful new training data to complement a machine learning model. The "most useful data" in general means data that brings information which was not available to the model before. However, the notion of usefulness differs substantially throughout various tasks. We claim that the most useful sentences for an MT system are those which it was not able to translate well. Quality Estimation models enable us to define the quality of automatic translations without comparing them to references (which would not be available in a realistic setting, otherwise one would simply use them directly for the MT training) or using human feedback (we need only small amount of human feedback to train a QE model) which makes the method easy to apply even in case of limited resources.

In this chapter we describe our experiments on the use of predictions of sentence-level QE models for data selection. We experiment with two scenarios: we estimate quality of automatic translations using (i) static QE models (ii) QE models which are dynamically updated with respect to an MT model. We also compare QE models with unequal performance.

6.1 Quality-based active learning technique

Active learning (AL) is a technique that proposes dynamic selection of the new training data to improve a machine learning model. The selection process is based on some properties of the model which help to define which data is more useful for the model at its current stage. There exist many ways of defining the usefulness of a new data instance (Settles, 2012). One of the popular choices is to use method which is known as *uncertainty labelling*: it assumes that if a model is uncertain about the label for a data instance, then having a true label for this instance is useful, because the instance is likely to contain some previously unseen properties. As we saw in section 2.2.5, the majority of AL methods for MT are based on uncertainty, where potential uncertainties are identified by unknown words, low-probability ngrams, etc. However, in many cases the uncertainty is caused by the differences in the domains of the baseline training data and the test data (i.e. data which needs to be translated by the MT system), and most research concentrated on solving the domain adaptation problem when performing active learning for MT.

In contrast to existing work, we address errors which do not necessarily stem from the difference in training and test domains, e.g. grammar errors or word order errors. Here the uncertainty criterion should be formulated differently: we need to select sentences in which a baseline MT system tends to make more such mistakes. Therefore, the most suitable sentences here will be the ones which are translated badly by our MT system. There are different ways of evaluating translation quality depending on the resources we have. The evaluation by a human expert would be ideal, but it is too expensive.

However, we can simulate the work of human expert by using a QE model. For a sentence-level QE model we need a relatively small amount of automatic translations which are manually labelled for quality. Then we can translate sentences in a pool of candidates with a baseline MT system, automatically estimate their quality with a QE system, and choose the ones whose quality is the lowest according to this QE system. For the selected sentences, we could then procure human translation or post-edits to add them to parallel data to improve the baseline MT system. We deliberately choose texts of the same domain for the baseline MT training data and new unlabelled data (pool) to ensure that the majority of errors will not be related to differences in domain.

Figure 6.1 shows the AL pipeline. We use three disjoint datasets in our experiments. We need a small parallel corpus which we will use for training of a baseline MT system. A QE model should be trained on a human-labelled dataset of automatic translations. This dataset consists of source sentences, their automatic translations into the target language, and the quality scores (here HTER scores) defined by an expert for each sentence. Finally, we also

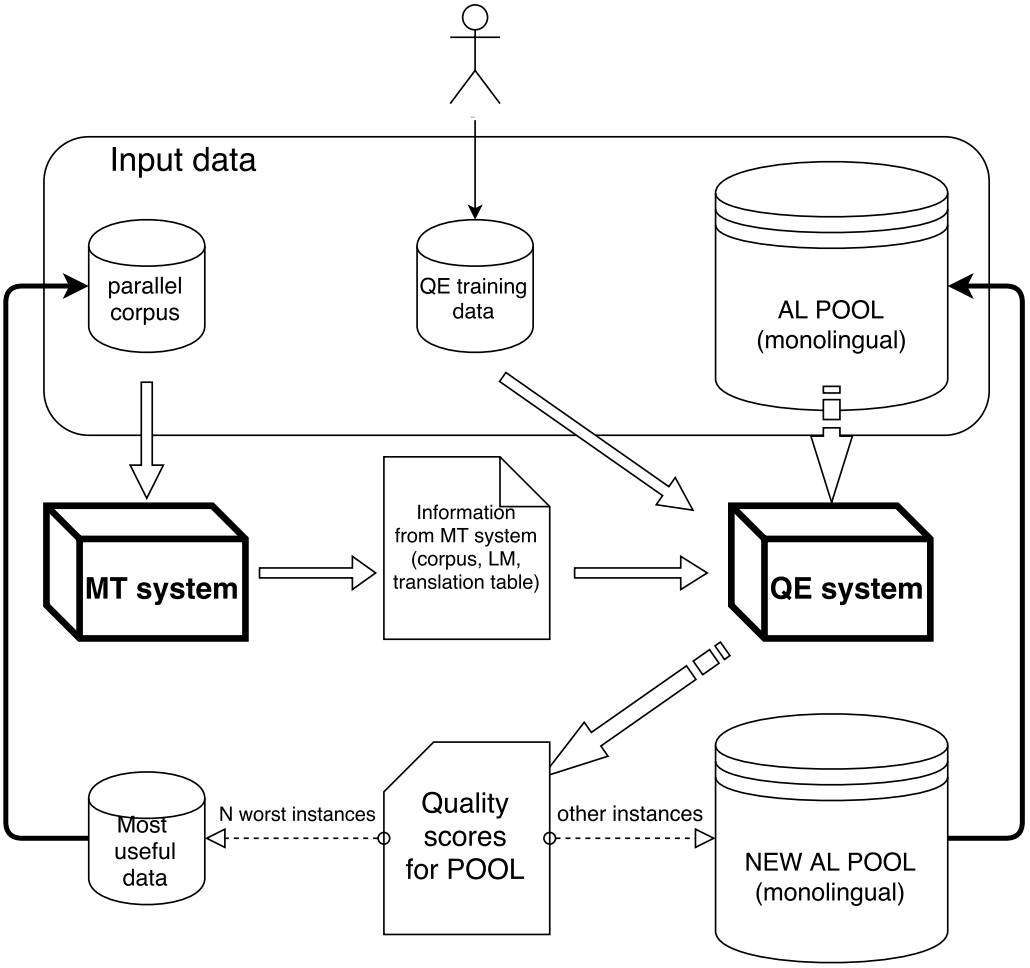

Fig. 6.1 Active learning pipeline for a Machine Translation system.

have a monolingual source corpus. This is a pool of source sentences from which we will first select the ones which are potentially more useful for MT system training, then get valid translations for them, and add them into the MT training data.

We begin with training of a baseline MT system from the parallel data we have. Then we train a QE model using the QE dataset and information from the MT system, such as corpus, language model, translation tables: a QE model needs them to extract the features. Thus, we can say that this QE model is targeted at predicting the quality of automatic translations produced by this particular MT system. After obtaining the QE model we translate the AL pool with the MT system and then predict the quality of these translations with the QE model. These scores create a partition of the pool: the worst N sentences should be translated and added to the MT training data, the remains are left in the pool for future iterations. Then the MT system is re-trained on the corpus extended with the selected worst sentences, and the features for the training data of the QE system are re-extracted using the information from the new MT system. After that the new set of the worst sentences is extracted from the pool. This process can be repeated multiple times until the pool is empty or until the budget for manual translation runs out.

6.2 Active Learning Experiments

We conducted a set of AL experiments using predictions of different QE models. All the experiments were conducted for the English–German language pair. This choice was motivated by the availability of QE training data: QE needs manually labelled datasets which are difficult to acquire and do not exist for many language pairs, so we needed to make sure to have enough data for a QE model training. A large corpus of manually post-edited translations from English into German for the IT domain was available, as well as a large parallel English–German in-domain corpus. This data was used to train a baseline MT system and to provide the components of QE models.

We cannot execute the human-in-the-loop scenario, therefore, we run simulated experiments, where the sentences in pool of candidates already have the human translations. In other words, in our experiments the pool is a parallel corpus, and after having selected new sentences from the pool we do not need a human translator to post-edit them, we just add the sentences along with their translations to the MT system training data. The pool contains 24,000 sentences. For each of them we have a manual post-edit of automatic translation (generated by a single MT system) as well as an independent reference translation. This allows us to check whether reference translations or post-edits are more useful as training data for MT systems.

6.2.1 Baseline Machine Translation model

The MT system used for the experiments is a phrase-level statistical MT system built with the `Moses` 3.0 toolkit[1] and with the following settings:

- 5-gram language models built with KenLM,
- Maximum phrase length — 7 words,
- 5-gram operation sequence model (OSM),
- Set of standard 14 decoder features (see section 2.2.1) plus 5 OSM features,
- Feature weight tuning performed with the batch MIRA algorithm.

OSM is a model which helps providing additional control over reordering in SMT. It is a lexicalised reordering model that represents the training corpus as a sequence of operations and learns a Markov model over this data. At each step a model can (i) generate both source and target words, (ii) generate only source or only target words, (iv) generate gap in the target (placeholder for a word), (v) jump forward or backward to an existing gap. OSM was shown to be effective for language pairs where translation requires long-distance reordering (e.g. German–English) (Durrani et al., 2013), therefore, we included it into our system setup. We use the batch MIRA tuning algorithm (Hasler et al., 2011) instead of commonly used MERT because it was shown to be more stable than MERT and often yield better translation quality (Cherry and Foster, 2012). The other settings are standard.

Our preliminary experiments (Logacheva and Specia, 2014) showed that the additional training data can improve the performance of an MT system significantly only if its size is comparable to the size of the original training data. Therefore, since our pool of candidates contains only 24,000 sentences, we had to limit the size of the baseline training data to 30,000 sentences. The table 6.1 contains the statistics of the MT training dataset. The evaluation of all trained systems was performed with `Multeval` tool[2] which computes BLEU, METEOR and TER scores and conducts significance test using approximate randomization technique (Clark et al., 2011). We used METEOR with paraphrase matching (Denkowski and Lavie, 2010).

6.2.2 Quality Estimation models

Dataset

For the training of our QE models we used the dataset released for the WMT-16 QE shared task. Each instance in this dataset consists of:

[1] http://github.com/moses-smt/mosesdecoder/tree/RELEASE-3.0
[2] https://github.com/jhclark/multeval

	Number of sentences	Number of words	
		English	German
Training (bilingual & monolingual)	30,000	518,375	529,077
Development	2,000	33,010	35,084
Test	2,000	33,832	36,841

Table 6.1 Statistics of the dataset used for training of the baseline MT system in the AL experiments.

- a source (English) sentence,
- its automatic translation into the target language,
- the post-edit of automatic translation performed by a professional translator,
- a reference translation (human translation done from scratch).

The automatic translations for this dataset were generated using a phrase-based SMT system trained on a 3M-sentence in-domain parallel corpus. This is a superset of the data used to train our baseline MT system, so we can say that the human feedback is to a certain degree relevant for our MT system, because it is based on the texts translated by a system with the architecture similar to ours, and the training corpora for these systems were similar. We aligned human post-edits with the automatic translations with `TERCOM` tool (Snover et al., 2006). This gave us sentence-level HTER scores (proportion of words in a sentence changed by a post-editor) and word-level binary scores (indication of whether a word was changed by a post-editor or its automatic translation was kept). We tested our QE models on the development set released along with the training data. The dataset statistics is outlined in table 6.2

	♯ sentences	Avg.sent. length	HTER	% of unedited sentences
Training	12,000	17.58	21.40	22.66
Development	1,000	19.49	19.54	19.10
Test	2,000	17.27	19.31	21.95

Table 6.2 Statistics of the English–German quality-labelled dataset.

The full QE training set consists of 12,000 sentences, however, we used half of it for the AL pool. Therefore, for the active learning experiments we used QE models trained on 6,000 sentences randomly selected from the WMT-16 QE training data.

Sentence-level vs word-level scores

For the active data selection we need sentence-level scores, because we are selecting entire sentences for MT retraining. The sentence-level scores can be acquired either by making

predictions with a sentence-level QE model, or by averaging the word-level predictions for words of a sentence. On one hand, our intuition is that sentence-level QE models should be more accurate at predicting scores for a sentence because they are targeted at this type of prediction and use sentence-level features that can be more informative for this type of task. On the other hand, an ideal word-level QE model that correctly identifies all the erroneous words in a sentence should by definition produce a correct sentence-level HTER score as well, because HTER score is essentially the proportion of erroneous words in a sentence.

We can directly compare performance of word-level and sentence-level QE models by producing the sentence-level HTER scores from binary word-level labels. The QE shared task held within the scope of WMT gives us a possibility to compare many models, because it contains both word-level and sentence-level tasks, and they use the same datasets. At the WMT-16 shared task all participants were provided with the same training, development, and test QE datasets as the ones we use for our experiments, so we used the word-level and sentence-level submissions for comparison.

This comparison is shown in table 6.3. Despite our intuition about the superiority of sentence-level models, the comparison did not show a clear answer to what models are better at predicting the sentence-level HTER score. The best-performing word-level model has higher Pearson correlation (official sentence-level metric) with the true HTER than the top sentence-level model. However, it should be noted that this word-level model has also outperformed all other participants of the word-level task by a large margin. Word-level models on average tend to perform worse than the sentence-level ones: around a half of word-level models cluster at the bottom of the table, and their scores are lower than those of any sentence-level model. Nonetheless, at least four word-level models (by Unbabel and Postech teams) outperform the majority of sentence-level models. Conversely, the analogous comparison of models for the WMT-15 QE shared task clearly showed the superiority of sentence-level models over word-level ones (see table 6.4).

However, for our experiments we do not use word-level QE models to generate sentence-level scores for a number of reasons. First of all, the word-level models which produced the best HTER scores are not feasible to replicate: they use complex features and advanced neural networks that are computationally expensive to retrain and re-run over multiple iterations of AL. Secondly, our aim was to show the potential use of QE for the improvement of Machine Translation rather than produce the best possible QE models for a particular task. Finally, there can be mismatches between internal evaluation of QE models and their performance in downstream tasks: a QE model that produces the highest score on a test set is not guaranteed to be optimal for AL.

	Model	Pearson r
word	UNBABEL/ensemble	0.539
sent	YSDA/SNTX+BLEU+SVM	0.525
word	UNBABEL/linear	0.481
sent	POSTECH/SENT-RNN-QV2	0.460
sent	SHEF/SVM-NN-both-emb-QuEst	0.451
sent	POSTECH/SENT-RNN-QV3	0.447
word	POSTECH/WORD-RNN-QV2	0.432
word	POSTECH/WORD-RNN-QV3	0.430
sent	SHEF/SVM-NN-both-emb	0.430
sent	UGENT/SVM2	0.412
word	UGENT-LT3/QE-TASK2-RF	0.411
word	UAlacant/2-word-level-SBI-Online-baseline	0.399
word	UGENT-LT3/QE-TASK2-ENS	0.389
sent	UFAL/MULTIVEC	0.377
sent	RTM/RTM-FS-SVR	0.376
sent	UU/UU-SVM	0.370
sent	UGENT/SVM1	0.363
sent	RTM/RTM-SVRb	0.358
sent	BASELINE	0.351
word	CDACM/RNN	0.327
word	BASELINE	0.300
word	RTM/s4-RTM-GLMd	0.269
word	RTM/s5-RTM-GLMd	0.261
word	SHEF/SHEF-MIME-0.3	0.242
word	SHEF/SHEF-MIME-1	0.237
word	UAlacant/word-level-SBI-Online	0.207

Table 6.3 Performance of word-level (shown in gray rows and marked with the prefix "word") and sentence-level (marked with the prefix "sent") models submitted to the WMT-16 QE shared task on the task of sentence-level HTER prediction.

	Model	Pearson r
sent	DCU-rtm-svr	0.550
sent	FBK-UPV-UEDIN-wp	0.540
sent	DCU-rtm-tree	0.518
sent	DFKI-svr	0.501
sent	USHEFF	0.432
sent	SHEFF-lite-sparse	0.428
sent	FBK-UPV-UEDIN-nowp	0.414
sent	Multilizer	0.409
word	SAU/KERC-SLG-CRF	0.399
word	SAU/KERC-CRF	0.395
word	HDCL/QUETCHPLUS	0.367
sent	DFKI-svr-xdata	0.349
word	HDCL/QUETCH	0.317
word	UGENT-LT3/SCATE-HYBRID	0.314
word	UAlacant/OnLine-SBI-Baseline	0.312
word	UAlacant/OnLine-SBI	0.297
sent	baseline	0.283
word	DCU-SHEF/BASE-NGRAM-5000	0.253
word	UGENT-LT3/SCATE-MBL	0.231
word	SHEF1/QuEst++-AROW	0.230
word	RTM-DCU/s4-RTM-GLMd	0.214
word	RTM-DCU/s5-RTM-GLMd	0.208
word	BASELINE	0.207
word	SHEF2/W2V-BI-2000.ext	0.198
word	SHEF2/W2V-BI-2000-SIM.ext	0.182
word	DCU-SHEF/BASE-NGRAM-2000	0.160
word	SHEF1/QuEst++-PA	0.070
sent	SHEFF-lite	0.052

Table 6.4 Performance of word-level (shown in gray rows and marked with the prefix "word") and sentence-level (marked with the prefix "sent") models submitted to the WMT-15 QE shared task on the task of sentence-level HTER prediction.

On the other hand, a bad-performing QE model is unlikely to be useful in a downstream task. Therefore, we tried to improve upon the sentence-level QE models which were available in current tools. We used some of the features of well-performing sentence-level QE models,data manipulation strategies suggested in chapter 4 and feature selection techniques that had been shown to improve the performance of QE models.

Black-box features

The baseline feature set for sentence-level QE used to in the "sent baseline" systems from tables 6.3 and 6.4 (Callison-Burch et al., 2012) offers a trade-off between informativeness and simplicity: it does not require any advanced information, so it can be extracted for virtually any language pair, and at the same time it provides enough information for meaningful QE results. On the other hand, the performance of a QE model can be improved if we add more information.

QuEst++ has a set of 79 *black-box* features (see section A.1.2): features that are extracted only from source and target sentences themselves (possibly using third-party tools or datasets) and do not require any information from the MT model. This is a superset of the 17 baseline features. Table 6.5 shows the performance of QE models with 17 and 79 features on the development set.

Feature set	Pearson r
17 features	0.386
79 features	0.420

Table 6.5 Performance of QE models with baseline and full black-box feature sets on the development set

Syntactic features

The QE model presented by the Yandex team (Kozlova et al., 2016) was the best-performing sentence-level model for the English–German language pair, so we decided to include some of their features in our model. We expected the new features to have the same effect on the model's performance as the one reported in the paper, because their model is essentially the same as the one we use: it is trained with the scikit-learn implementation of SVR (Support Vector Regression). Their feature set consists of baseline features plus syntactic, pseudo-reference, back-translation and advanced LM-based features.

However, we did not use some of the features reported by Yandex for different reasons. Several of them were not available to us: these were scores from large in-house LMs built on

web resources. In our opinion, such features should not be used in a work like ours, which is supposed to be replicable in languages other than the ones used in the experiments: even if we had such a resource for English–German language pair, we are not guaranteed to find data of the same amount and quality for a pair of lower-resourced languages, and this will make our results inapplicable to them.

The pseudo-reference features improved the score of the Yandex model by 10%, but we left them out deliberately as in our opinion they contradict the idea of QE. A pseudo-reference is a translation of the input data by some third-party MT model. However, if we consider it as a reference, it means that we expect this alternative MT system perform better than ours. In a real-world scenario in such case we would simply use the best available MT model to generate translations and not to provide features for QE. Conversely, if an alternative MT model is expected to perform worse than ours, information from it could be harmful for QE. Analogously, we left out back-translation features for the same reason.

The only subset of features from the Yandex model that we could use are syntactic features. They are not as universal and resource-independent as the black-box features, but they can still be extracted for most languages. We did not use the full set of syntactic features reported by Kozlova et al. (2016), because some of them were language-specific, and we wanted our feature set to be easily transferable into other languages. We extracted the following syntactic features for both source and target sentences (in total, 22 features):

- maximum depth of dependency tree,
- average depth of dependency tree,
- maximum width of dependency tree (number of dependants of the root),
- proportion of internal nodes of the tree (nodes with dependants, not root),
- number of subjects,
- number of verbs with dependant subject,
- number of dependant clauses,
- number of verbs,
- number of nouns,
- number of conjunctions,
- sentence starts with a verb (0/1).

The use of the syntactic features in a QE model along with the *black-box* feature set improved the Pearson correlation score of the model from **0.420** to **0.433** on the development set.

Sentence embeddings as features

In order to incorporate some less explicit information about the context we experimented with vector representations of sentences. Vector representations of words which encode the information about the words' contexts (Mikolov et al., 2013) have been shown to be effective for a number of NLP tasks including word-level QE (Shah et al., 2015). Later, vector representations were extended to sentence and document levels (Le and Mikolov, 2014), which enabled generation of fixed-length vectors for texts of arbitrary length.

The difference of word (or sentence) embeddings from other groups of features is the fact that they can make use of large unlabelled monolingual corpora. The only way of using such data resources that has been available to us so far is the extraction of LM-based features (e.g. probability of a sentence under an LM trained on a large dataset). However, unlike LMs, embeddings are able to express the similarity between units. Moreover, an LM provides only one score, whereas embeddings can have any number of dimensions (usually, vectors of 100 to 1,000 dimensions are used) and are potentially more informative.

We trained sentence vector representations with category2vec tool[3] on the 3M English–German in-domain corpus. We also concatenated the source and machine-translated parts of the QE training and development sets to the 3M corpus in order to generate the representations for them. We trained 400-dimensional vector representations for sentences of the source and the target parts of the QE data and then used them as features of our classifier. The size of vectors was chosen following the original work on sentence and paragraph vectors by (Le and Mikolov, 2014), where 400-dimensional vectors were shown to perform well on sentiment analysis task. However, in our case a model which used only sentence vectors performed poorly, so when we mixed the vectors with the baseline feature set, the performance deteriorated compared to the baseline.

Feature selection

We started with using the full set of *black-box* sentence-level features available in QuEst++ and a set of syntactic features from the Yandex team. The full set of black-box features contains 79 features many of which are correlated, and therefore can harm a model. We thus turned to selecting the best-performing feature subset. We performed feature selection as described by Shah et al. (2013). The method uses Gaussian Processes to rank features by their usefulness. We filter out features which have the same value for all sentences and rank other features according to the length scale parameter returned by the Gaussian Process model. We then train systems using subsets of n features with the highest rank gradually

[3] https://github.com/rakuten-nlp/category2vec

increasing n from 1 feature to the full set of features. The selected feature subset should be the one which achieved the highest quality on a development set. According to Shah et al. (2013), n varies from 10 to 25 for different datasets.

Nevertheless, for the English–German IT dataset the result is different from the one described in the research for other datasets. Figure 6.2 shows the results of our feature selection experiment for the two feature sets: black-box feature set and black-box features joined with syntactic features[4]. Here, we see the Pearson correlation score gradually growing and reaching the performance of a model trained on the full feature set only when we use most of the available features. We get some improvement over the full set performance, but it is marginal. Table 6.6 outlines the results of the feature selection experiments.

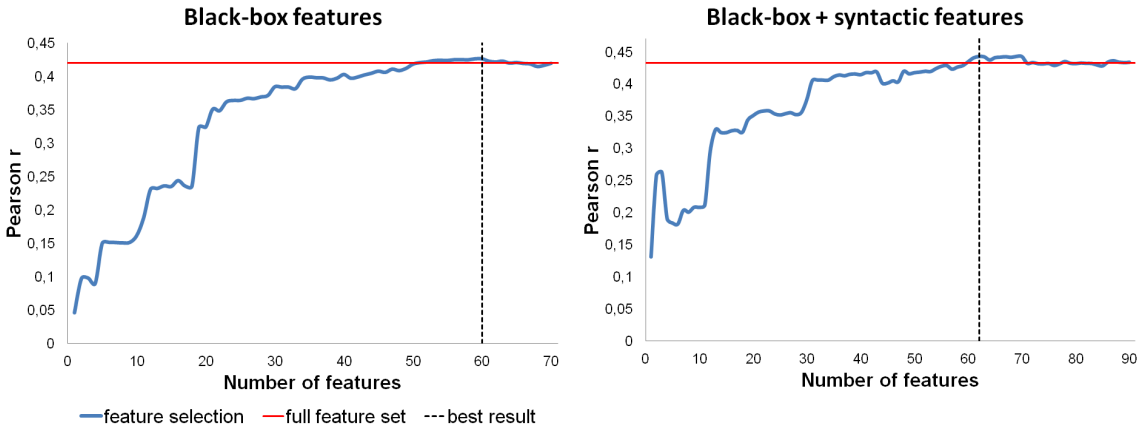

Fig. 6.2 QE model performance for different numbers of features used on development set. The blue lines show how the Pearson correlation scores for QE models change as new features are added. The red lines show the Pearson correlation of models that use full feature sets. The dashed lines show the performance of models with the optimal number of features.

	Number of features		Pearson r for development set	
	Full set	Reduced set	Full set	Reduced set
Black-box features	79	60	0.420	0.426
Black-box + syntactic features	101	62	0.432	0.442

Table 6.6 Results of feature selection experiments for different sets of features.

Some features that were filtered out were not decreasing the performance, but did not contribute to the predictions because their values were the same for all sentences of the training set. These were the features that check mismatches between the source and the target

[4]The maximum numbers of features shown in the figures are lower than the number of features in the sets, because some of the features had the same value for all instances of the training data and were left out before the feature selection.

sentences or within a sentence: mismatched brackets and quotation marks, differences in the number of full stops, question and exclamation marks in the source and the target data. These features had the value of 0 for all training sentences because there were no typographic errors in the data.

Moreover, the list of *black-box* features had a big number of features that were correlated: e.g. the number of distinct ngrams of different orders and their average probabilities, number of translations with different probability thresholds. During feature selection many of those features were removed because they were not contributing new information in the presence of other similar features. This holds for both feature sets we performed feature selection on (79 features and 79 + 22 features). However, we did not notice any pattern in feature selection in the two reduced feature sets. The selection of the most informative feature out of a list of correlated features was quite arbitrary: for example, in the 60-feature set has the information on the percentage of distinct bigrams in a sentence, whereas in the 62-feature set there is only the information on the percentage of distinct trigrams instead. We should also note that the 62-feature set contains less features from the *black-box* feature list (50 features), and feature selection kept 12 out of proposed 22 syntactic features. These 12 features contain source and target syntactic features in approximately equal proportions, so both source and target syntactic structures are informative for the task.

Note that we did not apply feature selection to the set of sentence embedding features. This was done because these features, unlike 79 black-box features, do not represent individual properties of source or target sentences. The whole vector of 400 dimensions is an object that encodes information about the sentence in an undecomposable shape. Therefore, it is impossible to select the most useful subset of the vector.

Overall, the performance of the sentence-level QE model can be improved via feature selection, but only marginally. From here on, we will be using filtered feature sets in order to reduce the size of QE models.

Performance of models

The performance of all the tested models is summarised in table 6.7. The use of more black-box features and incorporation of syntactic information improves the result, whereas sentence vector representations used as features do not give any positive effect. The feature selection marginally improved the scores for the development set, but on the test set the full black-box and black-box+syntactic feature sets performed slightly better.

As it was mentioned before, we used only half (6,000) of the original training data to train our QE models. However, we tested the performance of different feature sets on both full

Feature set	Trained on 12,000 sentences		Trained on 6,000 sentences	
	Development set	Test set	Development set	Test set
Baseline				
17 baseline features	0.386	0.364	0.375	0.360
Black-box & syntactic features				
79 black-box features	0.420	0.379	0.387	0.373
79 black-box + syntactic features	0.433	**0.406**	0.413	**0.393**
Vector representations				
Sentence vectors	-0.034	-0.011	-0.027	-0.045
79 black-box + sentence vectors	0.359	0.348	0.324	0.325
Feature selection				
60 black-box features	0.427	0.385	0.404	0.381
62 black-box + syntactic features	**0.442**	0.403	**0.435**	0.384

Table 6.7 Performance of sentence-level QE models using different features.

and half-size training corpora for comparison. We see that the trends hold for both datasets and that the reduction of performance is not large on the smaller dataset.

The baseline and other black-box features were extracted with `QuEst++`. The dependency trees for the extraction of syntactic features were generated with `Stanford CoreNLP` toolkit (Manning et al., 2014a) for English and `ParZu` toolkit (Sennrich et al., 2009a) for German using pre-trained models which are distributed with these tools. The sentence vector representations were trained with `category2vec` tool.

All the QE models were trained using `scikit-learn` implementation of SVR (Support Vector Machine Regression) with RBF kernel using the parameters chosen with grid search. The metric used for this task is Pearson correlation coefficient.The feature selection was performed with the `GPy` (GPy, 2012) implementation of Gaussian Processes.

If we compare performance of our best sentence-level QE model (black-box features + syntactic features) with the systems presented at the WMT-16 sentence-level QE shared task, we will see that it has the 7-th best result out of 15 systems (this holds for both 12,000 and 6,000 datasets) and is outperformed by models that predicted quality using neural networks (Kim and Lee, 2016; Shah et al., 2016) or information that was not available (Kozlova et al., 2016).

Artificial data

As it was shown in section 4.2, the use of additional artificial data for training can improve the performance of a sentence-level QE model. The initial goal of data generation was to mitigate the problem of insufficient data. Although this is not a major issue for our current

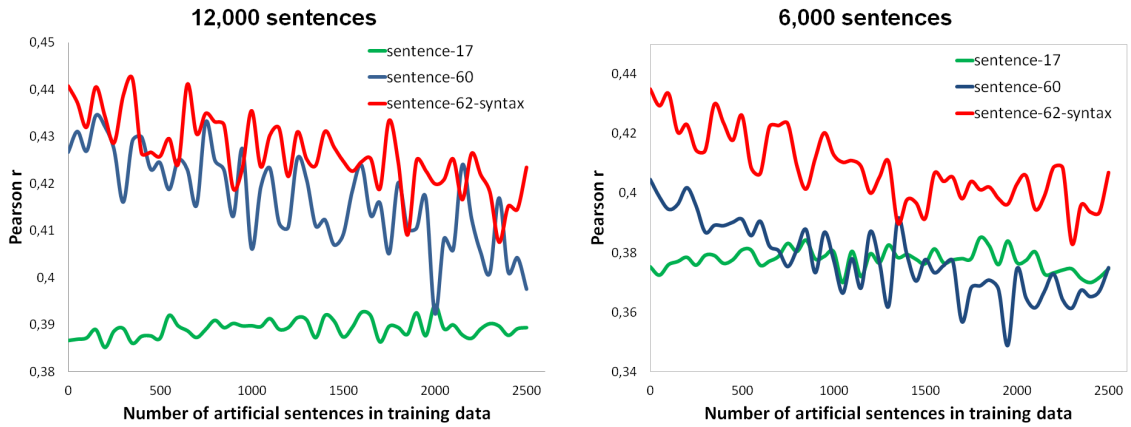

Fig. 6.3 Performance of QE models trained on **12,000** (left) and **6,000** (right) sentences and extended with artificial data.

QE dataset, we still experimented with adding some artificial data to the training set and see if it increases the model's quality.

We used the best-performing artificial data generation schemes: the error sequence was generated using a CRF model trained on our QE dataset (see **crfEG** in section 4.2.2), and the words for replacement were chosen using their translation probabilities (**lexprobWI** described in section 4.2.2). The error generator needs the training data — a dataset labelled with word-level edit operations (substitution, deletion, insertion). We used the same English–German dataset as that for training our QE models. The word inserter needs a lexical probability table to pick words from there. This table was generated with the 3M-sentences parallel corpus. The performance of QE models with artificial data was measured on the WMT-16 development set.

We experimented with blending some artificial data into training sets and find which amount of synthetic sentences is optimal for different QE models. We tested the performance of 3 QE systems: the one using 17 baseline features, 60 black-box features, and 50 black-box + 12 syntactic features. They are denoted as **sentence-17**, **sentence-60** and **sentence-62-syntax**, respectively. The left part of figure 6.3 shows the results for all three models. The baseline model is insensitive to the artificial data: its quality score only slightly fluctuates from 0.385 to 0.394. The extended models' fluctuations are larger: the quality changes from 0.396 to 0.426 for **sentence-60** and 0.407 to 0.442 for **sentence-62-syntax**. For both extended models we see a clear downward trend as more artificial data is used.

The ineffectiveness of artificial data in this case can be explained by the fact that the training corpus is large enough and cannot be improved with the addition of artificially generated examples. The original experiments with artificial data were conducted on much

smaller datasets: the dataset for the prediction of HTER scores used in our experiments in section 4.2.3 contained only 900 examples. Therefore, we experimented with the reduced dataset of 6,000 sentences. However, it also turned out to be too large for artificial data: as shown in the right part of figure 6.3, the same trends hold for the half-size QE models.

Final Quality Estimation models

It was shown that the best performance on the development set was demonstrated by the model trained on 50 black-box plus 12 syntactic features. However, we do not know how our QE models will behave on downstream tasks and we want to look into the relationship between the models' performance on internal (labelling of the test set) and external (active learning) tasks. Therefore, in our experiments we will be using three sentence-level QE models:

- **sentence-17** — model trained on 17 baseline features,
- **sentence-60** — model trained on 60 black-box features which were selected from the full set of 79 black-box features with the feature selection algorithm,
- **sentence-62-syntax** — model trained on 50 black-box and 12 syntactic features which were selected from the full set of 79 black-box and 22 syntactic features.

We use the versions of these models trained on 6,000 sentences from the WMT-16 dataset. The full sets of features for these models are described in the sections A.1.1, A.1.4 and A.1.5 of the appendix A.

6.2.3 Results

Preliminary work

The quality-based AL strategy we suggest here is an extension of our previous work (Logacheva and Specia, 2014) where we conducted similar experiments on a different dataset. Despite the similarity in the method this work was significantly different from the experiments we report here. First of all, the design of the experiment changed. The previous experiments were not AL experiments in the full sense, because the selection of data lacked the adaptation of the selection criterion (here a QE system) to the changing MT model. In the early experiments, we used the following procedure:

1. a baseline MT model is trained,
2. a sentence-level QE model is trained, using the output of the baseline MT model,
3. sentences of the AL pool are labelled for quality with the QE model,

4. N sentences with the worst predicted scores are selected and removed from the pool,
5. a new MT model is trained on the training data of the baseline MT model and the sentences selected from the AL pool,
6. the new MT model is used as baseline in the next iteration.

with the repetition of stages 4-6 until the pool is exhausted (note that all iterations use the same QE predictions made for the pool at the stage 3). This cannot be considered an application of AL to MT, because the selection criterion (QE model) is static and does not reflect changes in our MT model. In contrast to that, in our current work we (i) decode pool with the current version of the baseline MT model and (ii) extract features for QE training data and pool using the resources (corpus, LM, translation table) from the current baseline to adapt the QE model to the current MT output. However, in our current experiments we also reproduce the setup from the previous ones in order to compare the effect of static and dynamically changing QE models. We denote the static experiments **data selection**, as opposed to **active learning** experiments that re-train the QE model at every step.

Another notable difference with respect to the previous work is the data we used in our preliminary experiments: it had different size, belonged to a different domain and a different language pair. We used the LIG corpus (Potet et al., 2012b) which contains automatic translations of French sentences into English, their manual post-edits and free reference translations. The sentences of the corpus were taken from News Commentary corpus released for WMT-13 (Bojar et al., 2013). 1,800 sentences from the LIG corpus were used to train a QE model, and the rest (9,000 sentences) served as AL pool. The experiments showed that the additional training data has a noticeable effect on the MT quality when its size is comparable to the size of the baseline MT model training data. Therefore, the small size of the pool limited the size of baseline MT training data to 10,000. For the baseline MT system we used 10,000 sentences from the French–English part of News Commentary (we made sure to use sentences which do not occur in the LIG data). Another feature of the LIG corpus is the fact that it was post-edited by non-native speakers of English, which reduces its reliability.

In the previous work we compared the quality-based data selection with the random selection of the data and selection of sentences based on their real quality (HTER scores). We found out that the QE-based selection returns slightly higher scores than random selection 6.4. Surprisingly, selecting sentences by their real scores performs much worse than both QE-based and random techniques — which shows that the quality of sentences is not a good criterion of usefulness for an MT system, but a QE model can capture some features of the data which makes it useful for MT. We also found that the sentence-level QE predictions have a strong length bias, i.e., a QE model tends to give lower scores to longer sentences.

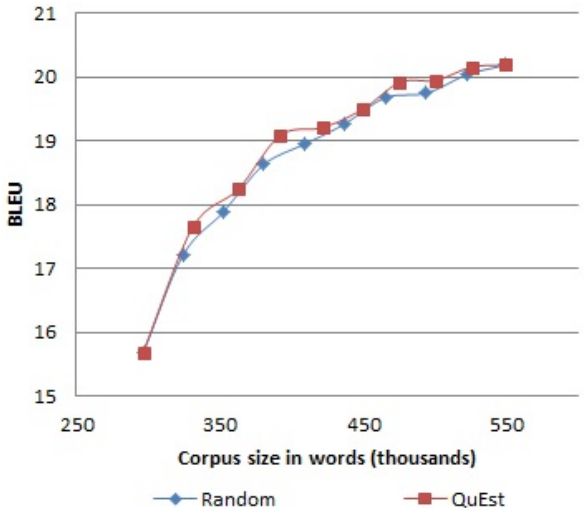

Fig. 6.4 Performance of QE-based AL technique on the French–English LIG dataset: comparison of the QE-based technique with random selection of training data.

Therefore, in subsequent experiments we will compare different MT models in terms of the number of words in their training data rather than in terms of sentences.

The LIG corpus contained two versions of target sentences: reference translations and post-edited automatic translations, so we could compare their impact on the quality of MT models enriched with new data. According to these experiments, the post-edited sentences are much more useful than the references: adding them to the MT training data yields much faster growth of BLEU scores 6.5. Since post-edits tend to be closer to original sentences than free translations (this observation was done by Potet et al. (2012b) when they analysed the properties of the LIG corpus), they generally produce better source-target alignments, which helped to improve the quality of phrases extracted by the MT system. In our current experiments we will also compare the MT models trained on references and post-edits for the AL-selected sentences.

Experimental setup

In our new experiments we would like to use the lessons we learned from the previous experiments and compare the results on the new data. Therefore, our main goals for the new experiments are the following:

- explore how the QE-based AL strategy works on data from a different domain (IT domain),
- compare the performance of static and dynamic QE models, i.e. compare *data selection* and *active learning* scenarios,

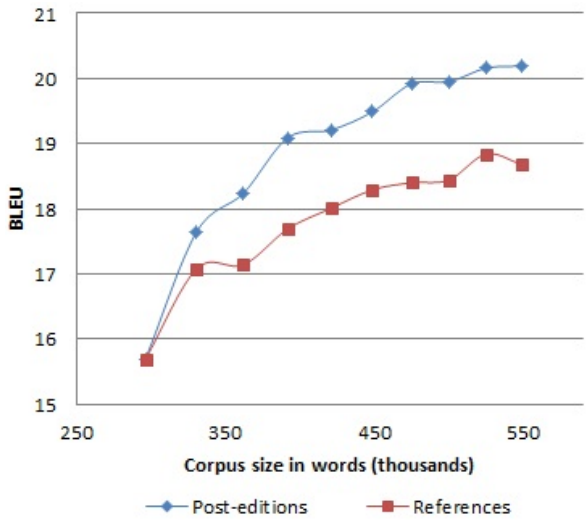

Fig. 6.5 Performance of QE-based AL technique on the French–English LIG dataset: comparison of post-edits and references.

- compare the performance of AL experiments on larger data,
- compare the performance of different QE models.

Therefore, our experiments will be run with the following settings:

- **QE models**: we use 3 sentence-level QE models of different performance (**sentence-17**, **sentence-60**, **sentence-62-syntax**). For the lack of post-edited data we used only 6,000 sentences from WMT-16 QE dataset for the training of models, the rest 6,000 along with 1,000 sentences of development set and 2,000 sentences of test set were added to the pool (see section 6.2.2).

- **MT models**: the baseline model is trained on 30,000 sentence pairs of IT domain (see section 6.2.1).

- **AL Pool**: the AL pool contains 24,000 instances, every of them consisting of English source, German automatic translation, manual post-editing done by professional translator who is native speaker of German, and free reference translation into German (the structure of the pool data is analogous to that of the WMT-16 QE corpus). Analogously to the previous work, we compare the performance of MT systems extended with **references** and **post-edits** (the data statistics of the experiments are outlined in table 6.8).

- **One epoch** of an experiment consists of:

  - selection of a new batch of **1,000** sentences according to some selection criterion (e.g. selected randomly, selected according to QE prediction, etc.);
  - training of a new MT model whose training data consists of the training data of the baseline system concatenated with the selected batch of new data;
  - this MT model becomes the baseline for the subsequent epoch.

- **Active learning techniques**:
  - **data selection** — new sentences are selected according to the predictions of a static QE model,
  - **active learning** — a new QE model is trained at every epoch. Features for the QE training data and the AL pool are extracted using the data from the current baseline MT system. The translation of the pool is performed at every epoch by the current baseline MT system.

- **Alternative strategies**:
  - **oracle**: sentences are selected based on their real HTER score.
  - **random**: sentences are selected randomly.

- The **Evaluation** of MT models is reported in terms of BLEU score, but all the trends between different models also hold for the METEOR score.

	Number of sentences	Number of words
Baseline MT model		
Training	30,000	518,357
Development	2,000	33,010
Test	2,000	33,832
QE model training	6,000	100,873
Pool	24,000	406,112

Table 6.8 Data statistics for the Active Learning experiments.

Quality-informed vs random selection

We start with the comparison of the performance of the QE-based AL strategy with random data selection. Figure 6.6 shows the three runs of quality-based AL technique with different QE models compared with random data selection. We plot the performance of individual MT models at different AL epochs. The blue line shows the performance of the **random** technique: it shows BLEU scores of MT models with additional 1,000, 2,000, 3,000, etc.

sentences. The yellow line shows the same for AL technique that used the **sentence-17** QE model, and so on. The better-performing techniques are the ones whose curves are above others — it means that they achieved some level of performance with less data.

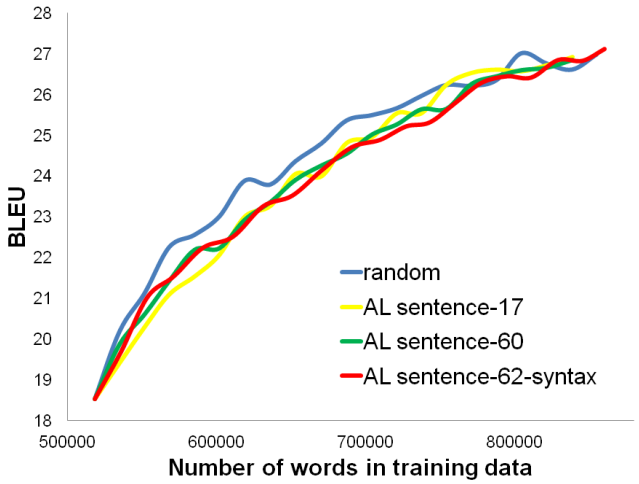

Fig. 6.6 Active learning: comparison of random data selection and QE-based selection technique with different QE models.

The figure shows the first difference from our previous experiment: the **random** strategy outperformed the quality-informed AL. It is important to note that the advantage of **random** technique over quality-informed AL remains only until we add around 250,000 new words to the training data. This number makes around 60% of the pool and corresponds to 13–15 epochs for different methods. We noticed this effect in our previous work: the performance of MT models that used almost all (85% and above) sentences from the pool became almost identical. However, in that experiment we had a pool of smaller size and had to use all of it, whereas now with the larger pool we can look at the behaviour of MT models before using all the AL pool data planned for addition. Figure 6.6 demonstrates a part of models with only 80% of the pool used, because the addition of small batches of data does not affect their performance any more, so the curves flatten. We see that after adding 60% of the data difference between the methods becomes insignificant. This shows that AL in general should be used when the size of available pool exceeds the size of needed data by several times. Otherwise, any selection strategies will equalise with random selection of the data.

It could be argued that this result might not reflect the changes in MT models' performance because of the way they were evaluated. The QE-based selection techniques selected new sentences based on their (predicted) HTER score, whereas the evaluation was conducted in terms of BLEU score. BLEU might be unable to detect the improvements induced by the data selected with a different objective. In order to check that we plot the results of the AL

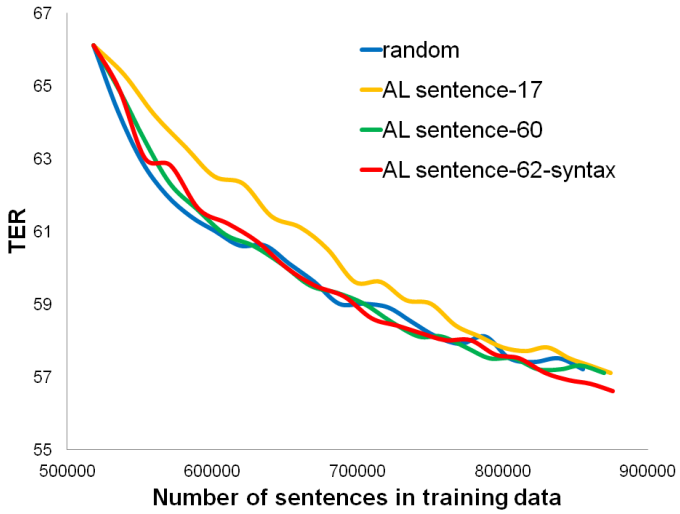

Fig. 6.7 Active learning: performance of random data selection and QE-based selection technique according to the TER score.

experiments in terms of the TER score (see figure 6.7). For the TER score better performance is reflected by lower values, so the performance curves are upside down compared to the ones formed by BLEU scores, and the best-performing technique is the one whose curve is lower, i.e. closer to the zero point.

Despite the differences between TER and BLEU, the TER score agrees with BLEU in the fact that random data selection was not outperformed by quality-informed AL technique. However, in figure 6.7 we see that AL performs very closely to random when advanced QE models are used, and the baseline QE model selects much less useful data than **sentence-60** and **sentence-62-syntax**, along with the **random** technique.

Baseline vs advanced QE models

The QE models tested perform differently in AL experiments: MT systems that use data selected by **sentence-60** and **sentence-62-syntax** models are significantly better than those trained on **sentence-17** data (see figures 6.6 and 6.7).

The comparison of predictions of these models showed some differences in their behaviour. Figure 6.8 shows the correlation between a model's predictions made at first epoch and the same model's predictions made at subsequent epochs. We see that all the predictions by the **sentence-17** model correlate very well. It means that the model does not change much. In contrast to that, the correlation of **sentence-62-syntax** outputs gradually falls over epochs. Decreasing correlation means that the model is adapting to the MT system it takes the data from. Conversely, **sentence-17** predictions fail to adapt to the changing MT systems

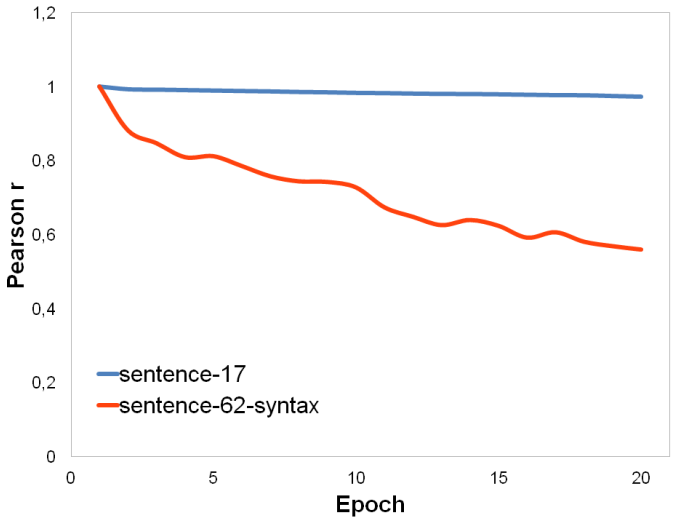

Fig. 6.8 Correlation of QE predictions made at first epoch with predictions of subsequent epochs.

— hence the worse results. The comparison of the average prediction values in different epochs (see figure 6.9) of the **sentence-17** and **sentence-62-syntax** QE models shows the same: while the average for **sentence-17** decreases smoothly, **sentence-62-syntax** averages fluctuate which means that they re-adjust to the new MT system at each epoch.

We should also highlight that this advantage of the feature-rich QE model holds only for 5 epochs (5,000 added sentences and approximately 100,000 words), which confirms our reasoning about the relative sizes of the AL pool and the selected data.

The application of the **data selection** technique gives a different result than **active learning**: here we cannot see any difference between the QE models (see figure 6.10), **sentence-17**, **sentence-60** and **sentence-62-syntax** perform almost identically. Moreover, the static QE models used for **data selection** experiments are better at selecting data for MT training: although they are still unable to outperform **random** selection, in the **data selection** experiments differences between **random** and quality-informed selection of the data become insignificant.

We also analysed the sentences selected by the QE models from the perspective of their usefulness: we compared the number of unique trigrams per token (the number of unique ngrams divided over the number of tokens in a text) in the batches selected by different QE models (see figure 6.11). This quantity reflects the data repetitiveness: the more different ngrams there occur per certain number of words, the less repetitive the data is. Here we also see the advantage of more advanced QE models over the baseline one: **sentence-17** model always chooses sentences with smaller numbers of unique ngrams. Note that this parameter

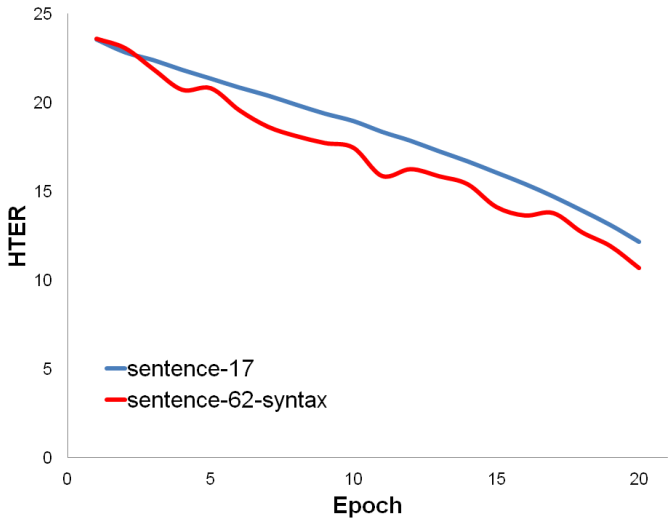

Fig. 6.9 Average prediction values in different epochs.

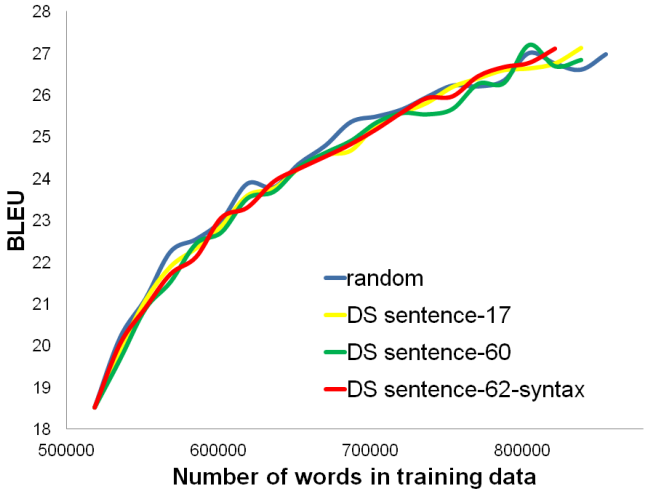

Fig. 6.10 Data selection: comparison of random data selection and QE-based selection technique with different QE models.

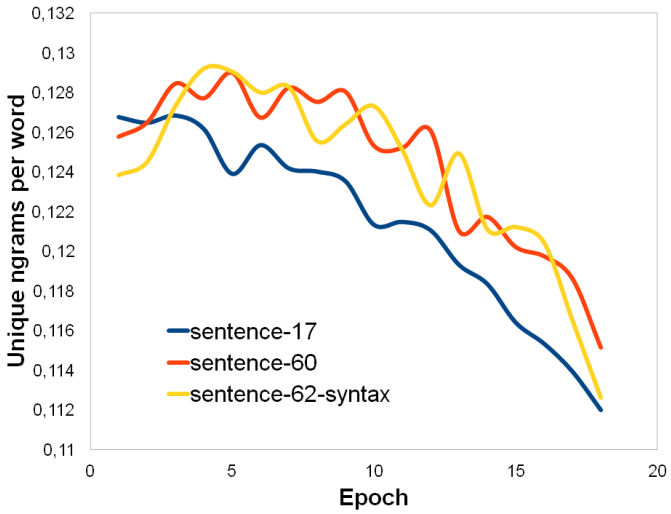

Fig. 6.11 Data selection: comparison of random data selection and the QE-based selection technique with different QE models.

is length-independent, so it is not influenced by any preferences of longer/shorter sentences that the models might have.

Furthermore, there is a pattern which is common to all QE models: they tend to choose more diverse sentences (sentences with more unique words and ngrams) first. This is clearly seen in figure 6.11: the number of unique ngrams gradually falls over epochs for all the models. This means that QE models predict worse scores to more diverse sentences, in other words, they consider those sentences worse translated (and probably more difficult to translate).

Active learning vs data selection

We compared the performance of different QE models in **active learning** and **data selection** tasks. We would also like to know how the two tasks compare with each other: does the tuning of a QE model to a particular MT model help, or can we use a single QE model to generate predictions? Figure 6.12 shows the results of **active learning** (AL) and **data selection** (DS) experiments with the best-performing QE model (**sentence-62-syntax**) and the **random** and **oracle** experiments. As we already saw in previous figures, none of our quality-informed techniques can outperform random data selection.

Moreover, the AL and DS techniques perform differently: it cannot be seen in figure 6.12 where we compare AL and DS runs which used **sentence-62-syntax** models, but the comparison of active learning and data selection performed with **sentence-17** (baseline QE

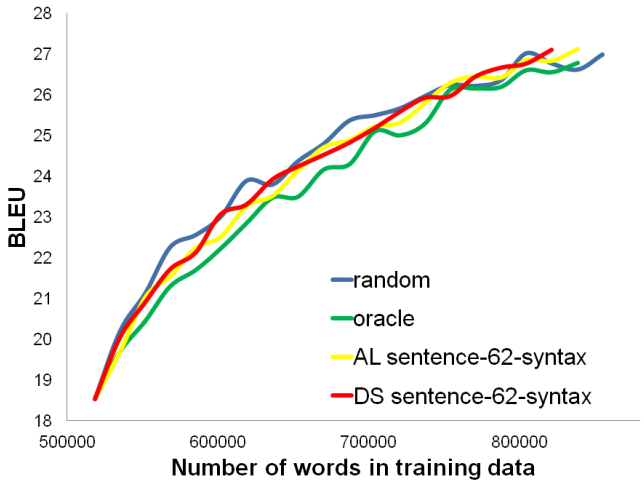

Fig. 6.12 Comparison of data selection different techniques.

model) shows that QE predictions from a static QE model were more useful for the task (see figure 6.13).

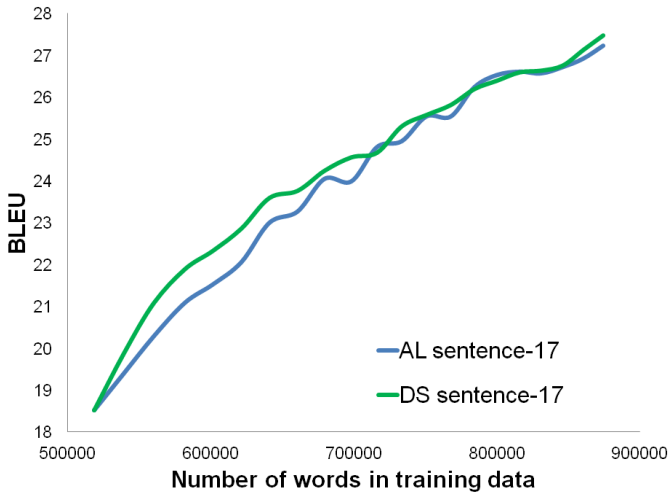

Fig. 6.13 Comparison of the results of *active learning* and *data selection* experiments with **sentence-17** QE model.

We also computed the correlation of predictions made by static QE models and their dynamic counterparts. Figure 6.14 shows that it is quite high for the first epochs of the AL experiment (0.7 to 0.78) and gradually falls throughout epochs.

The reason for such behaviour might be the following. In our dynamic QE models (used in AL runs) we use the data from the baseline MT system (corpus, translation table, LM) to extract QE features, as opposed to static QE model that uses resources built from large

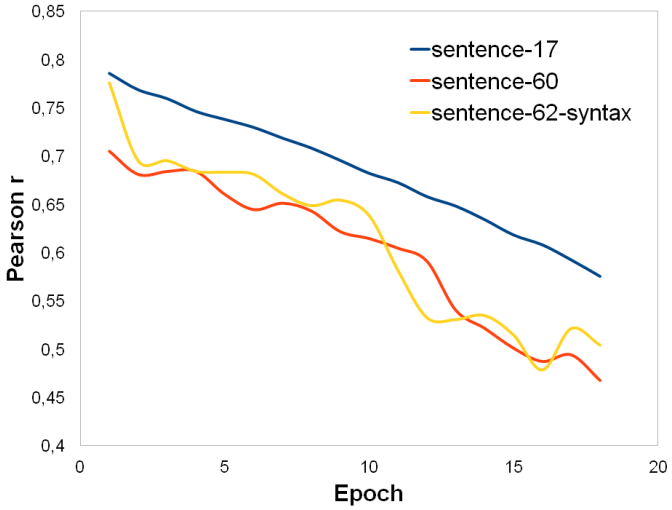

Fig. 6.14 Correlation between the predictions static and dynamic QE models over AL epochs.

datasets. Therefore, the dynamic models have only limited information on translation quality. Their predictions still correlate with those of better (static) models while they operate on the whole pool which contains good and bad sentences. However, as we remove the worst sentences from the pool to add them to the MT training data, the pool becomes less diverse in terms of quality. While the QE models of the first epochs needed to distinguish between very bad and very good sentences, the subsequent models' task is to catch fine-grained differences, which is more difficult and results in worse correlation with the static models' output. If we look at how the distribution of QE predictions changes throughout AL experiment, we can see the confirmation of the reasoning given above: as we remove worse sentences from the pool, QE predictions for the remaining ones get less diverse and slip towards zero (see figure 6.15). Since the target value for our QE models is HTER score, the value of zero means no errors.

Quality Estimation vs oracle

The negative results demonstrated by the **oracle** strategy corroborate the observation made during our previous work with the French–English data, where the oracle selection of sentences produced MT models with the lowest quality. However, this result contradicts the earlier shown observation that better-performing QE models are better at selecting data: while more accurate predictions provide a better usefulness criterion than worse predictions, the oracle scores (i.e. the most accurate quality predictions) turn out to deteriorate the downstream performance. A possible explanation for that is that sentence quality is an unsuitable criterion to measure usefulness to improve MT systems, however, QE models

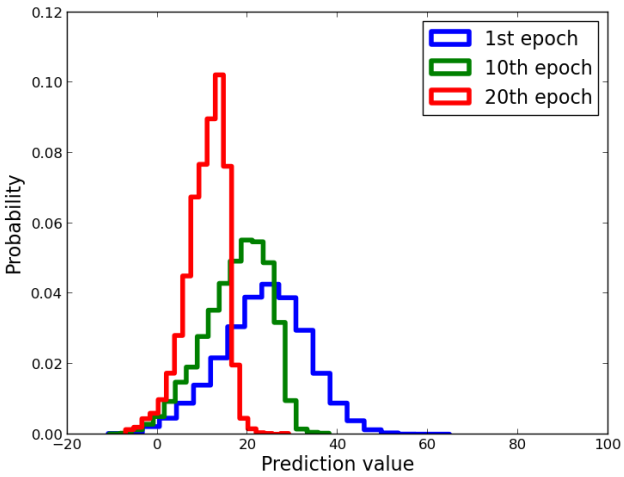

Fig. 6.15 Change of QE predictions distribution over AL epochs.

might capture some other features of sentences that make them useful. As we have already seen, QE models tend to predict worse scores to sentences with higher diversity. Therefore, bad (predicted) translation quality correlates with diversity of the data, probably, with different aspects of diversity: large number of unseen ngrams, unseen pairs of source–target words.

On the other hand, we can also suggest that the sentence quality does not reflect its usefulness, but the *approximate* quality does. Therefore, better-performing QE models might improve the sentences selection accuracy, but only up to some threshold. A QE model showing accuracy above threshold will deteriorate the performance of actively trained MT systems. Unfortunately, the we do not have enough data to verify this hypothesis: the reported experiments showed that better-performing QE models can be more useful than worse models in the **active learning** scenario, however, we did not observe situations where worse models outperformed better ones. Therefore, none of our QE models seems to have reached this threshold.

References vs post-edits

Analogously to our previous experiments we conducted two sets of runs for each combination of method and QE model: in one run we re-trained SMT systems with post-edited sentences selected from the AL pool, and in the other we used free reference translations for the sentences selected from the pool as the target side of the parallel corpus. The previous experiments showed that post-edited sentences were more effective for MT system training. However, this time we see the opposite result: as shown in figure 6.17, MT models that use

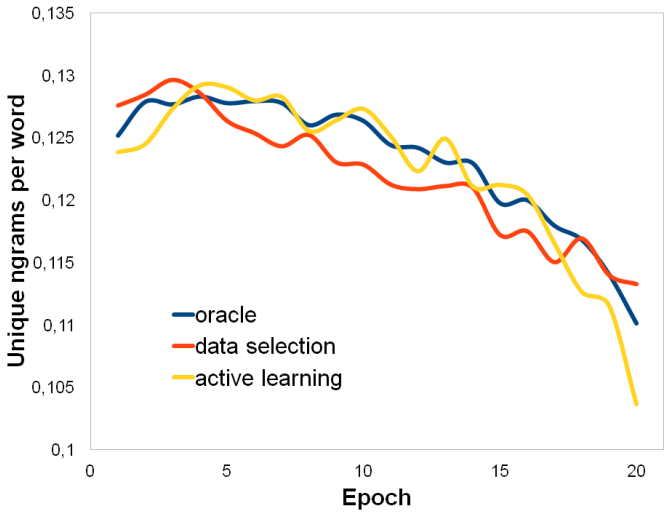

Fig. 6.16 Number of unique ngrams per running word in a batch: change over epochs.

reference data for training outperform the ones which were trained on post-edited data. This holds for all QE models and data selection techniques.

In order to understand the reasons of this difference we looked into the corpus statistics for the entire AL pool. We discovered that reference translations are slightly longer — 454,000 words versus 448,000 words in post-edits, with average sentence lengths of 18.93 and 18.65, respectively. On the other hand, post-edits are superior in terms of vocabulary: their vocabulary is slightly larger and has better coverage of vocabulary of the test set. However, if we examine the number of ngrams in the two datasets, we see a different picture: while the post-edited pool contains more unique words, the number of unique bigrams and ngrams of higher orders is larger in reference data (see figure 6.18). The same holds for the number of unique ngrams that occur in the pool *and* the test set but not in the baseline training data — i.e. ngrams which directly help improving the MT systems' scores. Figure 6.19 shows these numbers for post-edits and references and also demonstrates the dominance of reference data in this respect.

Therefore, the superiority of post-edited data over reference or vice versa may vary for different corpora. It is not defined by the similarity of post-edits and translation outputs, but rather depends on the lengths of post-edits and references, homogeneity of the domain and the skills of translators who performed translation and post-editing.

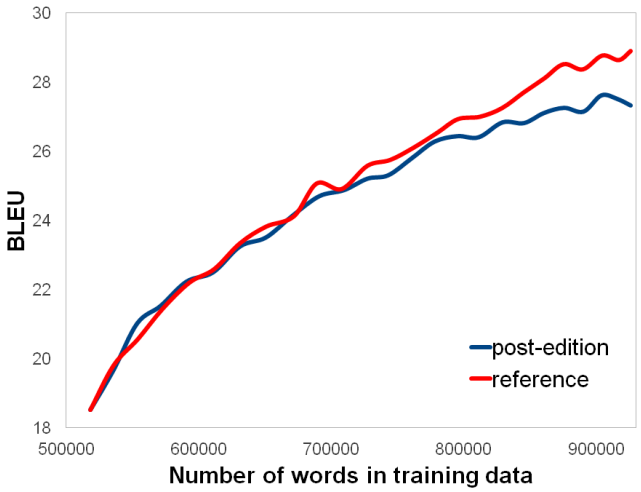

Fig. 6.17 Comparison of MT models trained on post-edits and references.

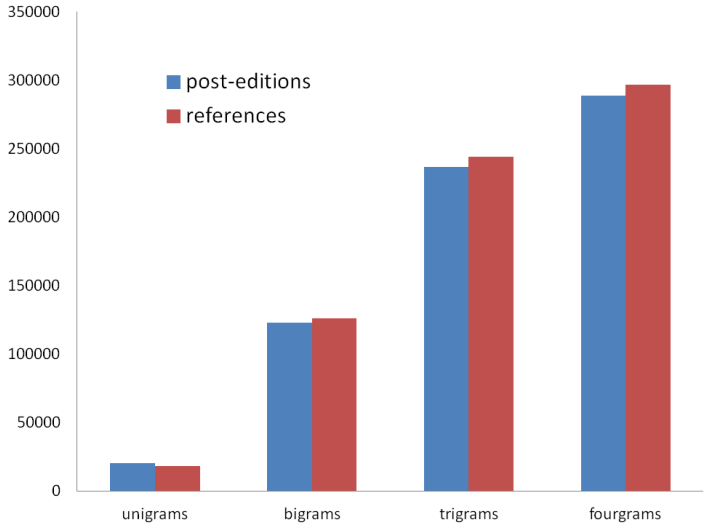

Fig. 6.18 Number of unique ngrams in reference and post-edited parts of the pool.

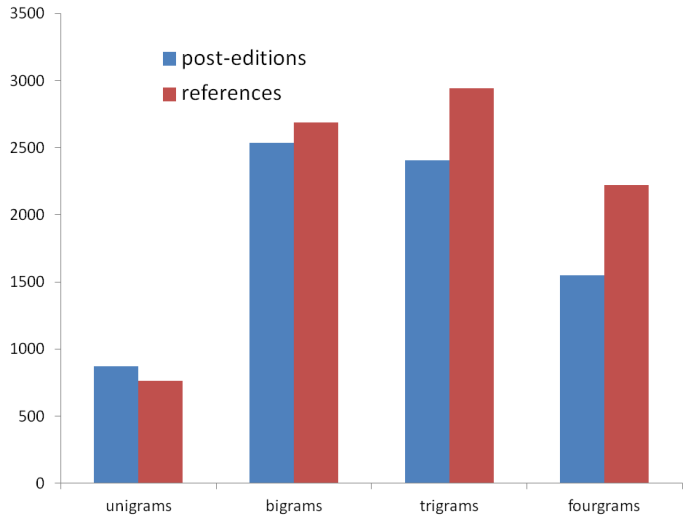

Fig. 6.19 Number of unique ngrams in references and post-edits that occur in the test set of MT model and do not occur in the baseline training set.

6.3 Conclusions

In this chapter we made an attempt to improve the performance of MT models by selecting new data for their training using quality predictions as a criterion of usefulness. We used three QE models with different performance (in isolation) to predict the sentence-level quality of translations and trained MT systems on the sentences which were predicted as having the worse translations by the baseline MT systems.

The data selected with any of the tested QE models proved less useful than the data selected randomly. However, we were able to see some regularities which suggest that our quality-based AL technique could be successfully applied. We saw that the dynamic QE models (models trained using MT system-specific data for feature extraction) adapted to a particular MT system, and better-performing QE models were better at such adaptation, resulting in higher quality of MT systems. Our adaptation was quite simple and probably not enough influential to bring larger improvements. Other ways of dynamically adapting QE models to the new data coming in a stream (de Souza et al., 2015) could be used in future work.

On the other hand, we found that static QE models where feature extraction uses all available resources (corpora, language models, etc.) and is not bound to resources of a particular MT system, in some cases perform better than the corresponding dynamic QE models. In this case there is no difference between the performance of baseline and advanced static QE models on the data selection approach. This means that this approach cannot

benefit from the use of better-performing QE models. Given that all of the tested static QE models could not outperform the random data selection, this makes the static QE-based data selection task ineffective.

The suggested strategy of data selection based on the quality of its translation is questionable altogether. In addition to our QE models we performed oracle selection — selection of sentences whose translation quality was the worst according to HTER score computed on human post-edits, in other words, this was the best possible performance of a QE model. This oracle-based selection was not able to outperform the other techniques tested, whereas we expected it to be more effective than our QE-based strategies and to show to what extent we could improve in this task by using better-performing QE models. Such result suggests that the translation quality is not a good usefulness criterion for a sentence. It can still be used, but it should be combined with other features. One of the major problems of oracle selection was its bias towards short sentences, therefore, combination of QE-based technique with sentence length restriction could be more effective.

Finally, in our initial experiments with the French–English data we saw that references and post-edits have different effect on MT quality when used as the target part of its training data: the MT systems trained on post-edits performed much better than those trained on the same amount of reference translations. However, these experiments with the new data did not confirm this finding: here, the difference between post-edits and references was mainly seen at the last iterations and was in favour of references. Therefore, these differences are data-specific and cannot be generalised for unknown datasets.

Overall, active learning is not an effective way of incorporation of human feedback via Quality Estimation. First of all, translation quality does not reflect the usefulness of the data for MT system training. And besides, it does not use human feedback in the most efficient way: AL requires the access to the large amounts of new data in addition to human feedback. Besides, the gain is very data-dependent and unverifiable in a real-world task, in other words, if AL was used to select the new data for a system training, it is difficult to check how it would perform with a different data selection technique in practice, i.e., it is difficult to run real (not simulated) experiments.

Another potential explanation for the negative result of our experiments is the complexity of our method compared to other AL methods. Most other data selection techniques do not consider the resulting quality of the end ML system (in this case, MT system), but rather the potential regions of its uncertainty in the model (e.g. words/phrases which were not represented in the vocabulary) or some other heuristics. These methods often do not require training of machine learning models and data collection, both of which need to be performed

to build a QE system. Therefore, in the situation of limited time and/or insufficient human resources simpler active learning techniques can be beneficial.

The human feedback could be more useful if it was incorporated into MT system more directly. The QE predictions can be used to assess the quality of translations generated by an MT system at different stages, thus improving the performance without additional data. In the next chapter we describe techniques of incorporation of QE predictions at the stage of translation generation and when selecting the best translation from a list of translation variants.

Chapter 7

Incorporation of quality scores into MT

A weak point of QE-based data selection described in the previous chapter is that it probably does not use the human labelling of the data in the most effective way: the added data does not have enough influence on the overall performance of the MT system. We can extract more explicit information from the quality labelling and from the quality scores we predicted, so a tighter integration of a QE model within an MT model might lead to larger improvement of automatic translations.

Therefore, we turn to an application of QE which was the original motivation for this task: we use predicted quality scores to inform an MT model itself of its quality and guide it while producing translation. Blatz et al. (2004) were the first to combine quality predictions with MT model scores or even replace the model score with the prediction in order to get a more objective measurement of the quality of output. However, at that time there were no manually labelled QE datasets, so the QE models were trained on data automatically labelled via comparison with reference translations. This data was very noisy, which resulted in QE models being unable to improve MT performance (section 3.1). There were also some later attempts to incorporate QE scores into SMT (Luong et al., 2014a,d). They incorporated word-level QE predictions in as an additional feature in Minimum Bayes Risk decoding (second-pass decoding which operates on the graph of possible translation variants). However, this direction has not been researched thoroughly.

A new feature can be incorporated at different stages. The easiest one is the re-computation of model scores of translations with respect to a new feature after the translations have been generated. At this stage we consider a list of N candidate translations (further denoted as *N-best list*) which got the highest score under the baseline feature set. We compute the values of our new feature(s) for each of these translations and return a revised model score which takes the new information into account. Based on this score we **rererank the N-best list**.

A new feature can be incorporated at the stage of **tuning**: while the translations are generated without using this feature, it can be computed for complete sentences and then be added to the final model score. Therefore, the information from this feature can be considered by the tuning algorithm and it can have an impact on how the importance (weight) of features is defined.

Finally, a feature can be computed at **decoding** time. Unlike N-best list reranking and tuning, during decoding the new feature influences the process of hypotheses generation, which can potentially lead to larger growth in translation quality.

In this chapter we describe our experiments with these three methods for the incorporation of QE scores of different levels (sentence-level, phrase-level, word-level) into a statistical phrase-based MT model. We conduct our experiments on two MT systems of different domains (IT and pharmaceutical) and different language pairs (English–German and German–English). Our N-best list reranking experiments are described in section 7.2. The results of our retuning experiments are reported in section 7.3. Section 7.4 contains the description of our experiments on incorporation of QE predictions at decoding time.

7.1 QE models

First of all, we select QE models that will be used in the experiments.

7.1.1 Datasets

We have two QE datasets available to us — an English–German IT dataset and a collection of German–English pharmaceutical texts. The former was used for in our phrase-level QE shared task (section 5.3) and for our AL experiments in chapter 6 (see section 6.2.2 for the statistics of the dataset). Note that while in AL experiments only half of the data (6,000 sentences) was used for training, here we use the whole dataset of 12,000 sentences.

The German–English dataset is larger: it contains 25,000 sentences in the training set and another 1,000 and 2,000 in the development and test sets, respectively. The parallel data for the MT training and the data with manual quality labelling come from the pharmaceutical domain. Both datasets were created by TAUS.Analogously to the WMT-16 QE dataset used in our previous experiments, this QE dataset contains the following information for every instance:

- source (German) sentence,

- automatic translation into the target (English) language performed by an SMT system with pre-reordering module which reordered the original German sentences before training and decoding so that their word order is closer to that of English sentences,
- post-edit of the automatic translation written by a professional translator,
- reference translation into English.

This QE dataset is different from the one we used before. Besides the different domain, it presents other differences from the WMT-16 QE dataset. First of all, it is twice as large: it has 25,000 sentences as opposed to 12,000 sentences in the WMT-16 corpus. The machine translation of the dataset is of much higher quality: the average HTER value is only 15.9 as opposed to 21.4 for the WMT-16 data, and 40% of sentences were left unedited (i.e. they were correctly translated by the MT system), whereas in the IT dataset there are only 20% of such sentences. The QE dataset statistics are outlined in table 7.1.

	♯ sentences	Avg.sent. length	HTER	% of unedited sentences
Training	25,000	17.36	15.94	41.81
Development	1,000	17.39	15.48	43.60
Test	2,100	17.39	14.74	44.15

Table 7.1 Statistics of the German–English quality-labelled dataset.

7.1.2 Sentence-level models

For the English–German experiments we use the sentence-level QE models that we used in our AL experiments[1]:

- **sentence-17** — model trained on 17 baseline features,
- **sentence-60** — model trained on 60 black-box features,
- **sentence-62-syntax** — model trained on 62 black-box and syntactic features.

For the German–English experiments we conduct the selection of sentence-level QE models analogously to the one described in section 6.2.2. Analogously to our previous experiments, here we would like to compare the performance of different QE models on downstream tasks. Therefore, we train a baseline model which uses 17 features and two models with extended feature lists: the *black-box* feature list of 79 features and the *black-box* features in conjunction with 22 syntactic sentence-level features. As in previous experiments, the models are trained with `sklearn` implementation of SVR, parameters for SVR are defined

[1]The detailed descriptions of these models are given in section 6.2.2

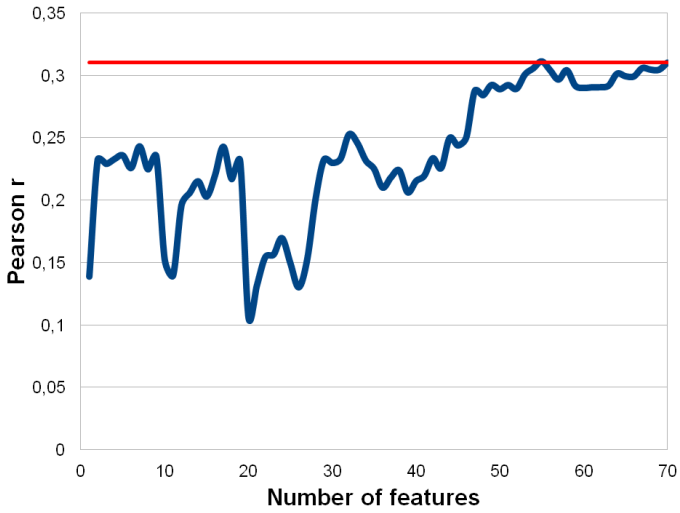

Fig. 7.1 Feature selection experiment for 79 black-box features with the German–English data on development set. The blue line denotes the performance of models with growing number of features, the red line denotes the baseline performance for the feature set (the performance of a model that uses all features).

via grid search. The scores of these QE models for the new dataset are very different from those in our previous experiments: here, the best result is achieved by the baseline model, and the use of the 79 *black-box* features or its combination with syntactic sentence-level features could not beat the baseline model.

We perform feature selection (Shah et al., 2013) on the two larger feature sets. The result is different from the one for feature selection we performed on English–German QE models (see section 6.2.2). There, we saw a slight improvement in model performance when using around 2/3 of the features. Here, the algorithm failed to select any better-performing subset of 79 *black-box* features (see figure 7.1), while its application to the set of 79 *black-box* plus 22 syntactic features yielded significant improvements when half (50) of the feature set is used (see figure 7.2). However, the 50-feature model could only reach the performance of the baseline model, but not improve upon it.

The detailed results for different QE models are presented in table 7.2. Here we see that the performance of the models is also different for the development and test sets: the baseline QE model is much better at predicting the test labels than the 50-feature models, although they perform almost identically on the development set. Likewise, the test labels are better predicted by the model that uses the full set of *black-box* and syntactic features, although on the development set labels its filtered 50-feature version performs much better. Overall, our QE model evaluation is not very conclusive. Therefore, we decided to use the three sentence-level QE models which are analogous to the ones we used in previous experiments:

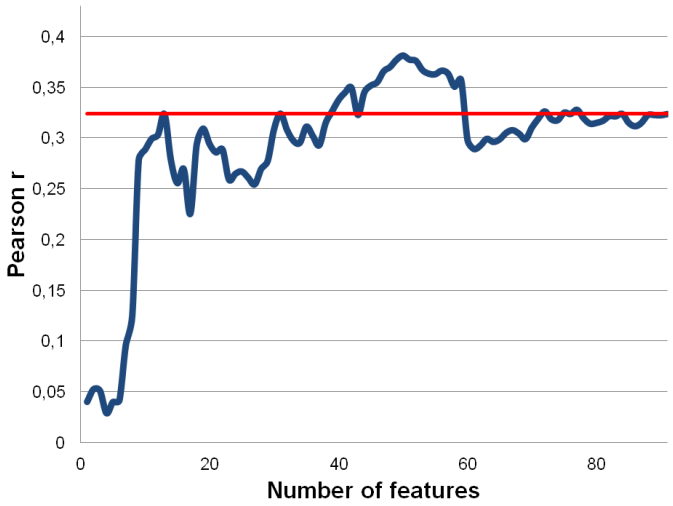

Fig. 7.2 Feature selection experiment for 79 black-box plus 22 syntactic features with the German–English data on development set. The blue line denotes the performance of models with growing number of features, the red line denotes the baseline performance for the feature set (the performance of a model that uses all features).

sentence-17 (baseline QE model), **sentence-79** (model that uses only *black-box* features) and **sentence-50-syntax** (model that uses feature selection over *black-box* and syntactic features).

	Development	Test
sentence-17: 17 baseline features	0.382	0.405
sentence-79: 79 black-box features	0.310	0.210
79 black-box + 22 syntactic features	0.324	0.321
Feature selection		
sentence-50-syntax	0.381	0.305

Table 7.2 Pearson correlation score for the sentence-level QE models trained on the German–English dataset.

7.1.3 Word-level models

Similarly to sentence-level QE, the word-level QE also has an established baseline: it is a CRF model which uses a small number (20-25) of surface word-level features (see section 3.3). The model has been used as an official baseline for the shared tasks on word-level QE at WMT-15 (Bojar et al., 2015) and WMT-16 (Bojar et al., 2016).

We use this baseline word-level QE model for N-best list reranking to provide the baseline estimate of usefulness of QE scores. However, as with sentence-level models, here we can

also extend the baseline model to yield better predictions and check how they influence the downstream performance. We improved our model using some additional features (see below) and data selection technique described in section 4.1.1. We trained our word-level QE models on the dataset released for the WMT-16 QE shared task. It is described in section 6.2.2.

Training algorithm

Most works word-level QE is cast as a sequence labelling problem and is often addressed with Conditional Random Fields (CRF) model. For our current experiments we also use CRF model implemented in `CRFSuite` toolkit. We start by selecting the training algorithm for our QE models from 5 optimisation algorithms available in `CRFSuite`:

- **lbfgs** — L-BFGS with L1/L2 regularization,
- **l2sgd** — SGD with L2-regularization,
- **ap** — Averaged Perceptron,
- **pa** — Passive Aggressive,
- **arow** — Adaptive Regularization of Weights (AROW).

We compare their performance on the set of 22 baseline word-level features. Table 7.3 shows the performance of 5 CRF models for the English–German and German–English datasets trained on the baseline feature set with the 5 different algorithms and tested on the corresponding development sets.

Analogously to our experiment with phrase-level models in section 5.3.3, we can see that the passive-aggressive algorithm works better for this task, therefore, we will use it to train all subsequent word-level QE models. The model trained on the baseline features with **pa** algorithm (row 4 of table 7.3) will be further referred to as the **baseline** word-level QE model.

Feature set

The work by Kreutzer et al. (2015) and Martins et al. (2016) show that a significant improvement can be achieved without adding new information about words, but only by manipulating the available information. Both papers describe models which use the baseline feature set enriched with features that combine values of individual features: e.g. for two features $f_i = val_i$ and $f_j = val_j$ of an instance we create a feature $f_{i,j} = val_i\_val_j$ (both val_i and val_j should be strings). A linear classifier trained on these combined features was shown to perform better than the majority of models participating in QE shared task (Kreutzer et al., 2015). Following this work, we use six combined features:

	English–German			German–English		
Algorithm \ Metric	F_1-OK	F_1-BAD	F_1-**mult**	F_1-OK	F_1-BAD	F_1-**mult**
lbfgs	0.238	0.889	0.212	0.298	0.940	0.280
l2sgd	0.212	0.891	0.189	0.270	0.940	0.254
ap	0.320	0.886	0.284	0.306	0.941	0.288
pa	0.375	0.877	**0.329**	0.336	0.937	**0.315**
arow	0.326	0.850	0.278	0.321	0.914	0.294

Table 7.3 Performance of different `CRFSuite` optimisation algorithms for word-level QE models with the baseline feature set. Performance is computed on development sets.

- target token + source token,
- target token + left context,
- target token + right context,
- target token + left source context,
- target token + right source context,
- target POS + source POS.

Another improvement of the work by Martins et al. (2016) that we adopt is the use of syntactic features. We follow Martins et al. by including the following syntactic features they use in their models:

- type of dependency between the token and its head word,
- token + type of dependency,
- token + its head word,
- POS of the token + POS of its head,
- token + token's closest sibling to the left,
- POS of the token + POS of its closest sibling to the left,
- token + token's closest sibling to the right,
- POS of the token + POS of its closest sibling to the right,
- token + its head word + head word of the head (grandhead),
- token's POS + its head's POS + its grandhead's POS.

These features are extracted for both the target token and a source token it is aligned to, which gives 20 features. If the target token is unaligned, all syntactic features for the source equal to `None`.

Table 7.4 shows the comparison of the baseline and extended feature sets for the two datasets. It is clear that the performance of word-level QE models consistently improves

	English–German		German–English	
	Development	Test	Development	Test
Baseline feature set	0.329	0.324	0.315	0.298
+ combined features	0.340	0.340	0.351	**0.336**
+ combined & syntactic features	**0.367**	**0.371**	**0.383**	0.335

Table 7.4 Influence of additional features on the performance of word-level QE models.

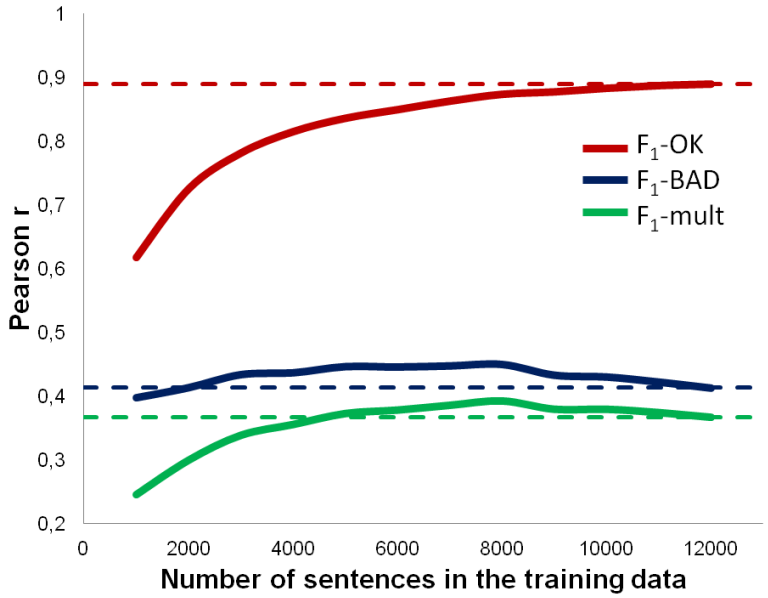

Fig. 7.3 Performance of word-level QE models for the English–German dataset with different amounts of data. The solid lines show the scores for models which use subsets of the data of different sizes, the dashed lines show the scores for the model trained on the whole dataset.

when the feature set is extended with combined and syntactic features. The only exception is the German–English test set whose labels are slightly better predicted without syntactic features.

Data filtering

Another simple yet effective strategy that we tested before is filtering of the training data: in particular, dropping the sentences that contain no or few errors: as it was shown in section 4.1.1, they skew the labels distribution towards the "OK" label, making the classifier predictions too optimistic. In order to mitigate this effect we throw out sentences which contain the smallest number of erroneous words: we sort the sentences from the worst to the best and train QE models using the top N sentences as the training data until we find which proportion of the original data gives the optimal performance on a development set.

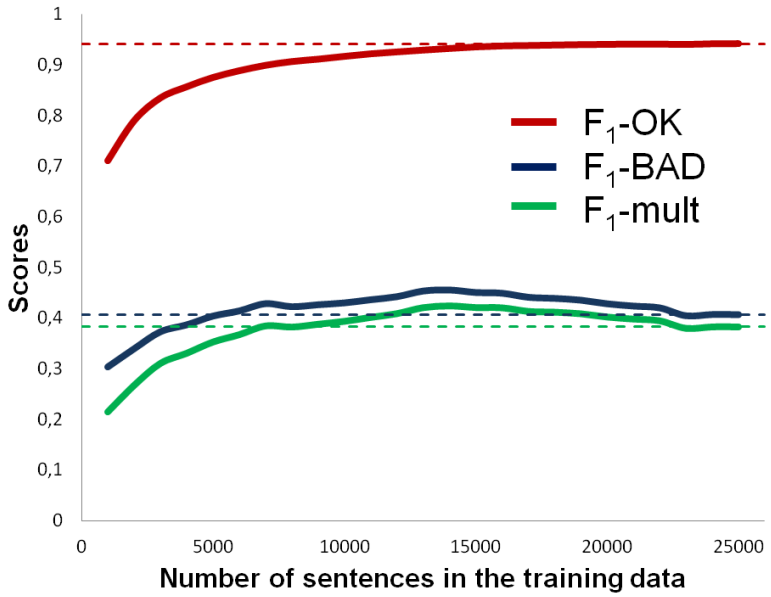

Fig. 7.4 Performance of word-level QE models trained on subsets of the training data for the German–English dataset. The solid lines show the scores for models which use subsets of the data of different sizes, the dashed lines show the scores for the model trained on the whole dataset.

Figure 7.3 shows the results of this experiment for the English–German dataset. While the F_1-OK score keeps growing as we add more data, the F_1-BAD score starts decreasing when the training data exceeds 6,000 sentences. Their multiplication reaches its maximum at 8,000 features. Therefore, using only 2/3 of the data we achieve the F_1-mult score of 0.375, which corresponds to 10% relative improvement.

The results for the German–English dataset are analogous to those of the English–German dataset. Figure 7.4 shows the performance of models trained on filtered German–English data. The baseline F_1-mult score is reached when using only 6,000 training sentences out of original 25,000. This happens because around 40% of the sentences have no errors and are thus not helpful for a word-level QE model training. The highest performance is reached when training on 14,000 sentences with the highest number of errors — in fact, these are almost all sentences that contain any errors.

Models used for the experiments

We experimented with different feature sets and data manipulation techniques. The feature extraction was performed with the Marmot tool. We also used the following third-party tools:

- `TreeTagger` (Schmid, 1994) — part-of-speech features,

	Development set	Test set
Baseline features	0.329	0.324
Baseline + combined + syntactic features	0.367	0.371
Baseline + combined + syntactic features, filtered	0.392	0.390
Top participants of WMT-16 shared task		
UNBABEL/ensemble	–	0.495
UNBABEL/linear	–	0.462
UGENT/LT3-RF	–	0.411
UGENT/LT3-ENS	–	0.381

Table 7.5 Comparison of our word-level QE models trained on the English–German data and participants of the WMT-16 QE shared task.

	Development set	Test set
Baseline features	0.329	0.324
Baseline + combined + syntactic features	0.367	0.371
Baseline + combined + syntactic features, filtered	0.392	0.390

Table 7.6 Performance of our word-level QE models trained on the German–English data.

- `ParZu` (Sennrich et al., 2009b) — syntactic features for German,
- `Stanford CoreNLP` (Manning et al., 2014b) — syntactic features for English,
- `SRILM` (Stolcke, 2002) — ngram counts for LM features,
- `NLTK`[2] — stopword lists for English and German.

The results of these experiments for the English–German dataset are outlined in table 7.5. The selection of optimal dataset was performed on the development set from the dataset released WMT-16 QE shared task. Based on the score for the test set, our best-performing model has 4-th best result out of all models submitted to the 2016 word-level QE shared task (see the comparison of our models with the top 4 participants of the WMT-16 QE shared task). The performance of the German–English QE models is outlined in the table 7.6.

For our further experiments we use the following two models:

- **word-BL** — word-level baseline QE model (row 1 in tables 7.5 and 7.6),
- **word-extended** — word-level QE model that uses baseline features, feature combinations and syntactic features, and is trained on 8,000 sentences (for the English–German data) or on 14,000 sentences (for the German–English data) with the highest HTER (row 3 in the tables 7.5 and 7.6).

We chose the baseline model which is easy to produce, and our best-performing model. By experimenting with two models of different quality we can see to what extent the accuracy

[2]http://www.nltk.org/

of quality predictions can influence downstream performance. The full lists of features used in these models can be found in sections A.2.1 and A.2.2 of appendix A.

7.1.4 Phrase-level models

For the English–German data we have a baseline phrase-level QE model (described in section 5.1.6) and a better-performing model which is based on the baseline with filtered training dataset and addition of new features (described in section 5.3.3). In our experiments we will refer to them as:

- **phrase-BL** — phrase-level baseline,
- **phrase-extended** — phrase-level QE model which uses baseline and context features and is trained on 9,000 sentences of the WMT-16 dataset with the highest HTER.

The full lists of features used in these QE models can be found in sections A.3.1 and A.3.2 of appendix A.

For the German–English dataset we also trained a baseline model and a model with the extended feature set. We conducted a data filtering experiment for the latter model. The experiment showed that the optimal number of sentences in the training for this dataset is 17,000. Figure 7.5 shows the performance of German–English phrase-level QE models trained on different numbers of sentences. The performance of the phrase-level models we will use for the experiments is reported in table 7.7.

	F_1-BAD	F_1-OK	F_1-multiplied
phrase-BL	0.432	0.913	0.394
phrase-BL + additional features	0.452	0.910	0.411
phrase-extended	0.484	0.900	0.436

Table 7.7 Phrase-level QE models used for the experiment.

For the German–English experiments we will use the same pair of models: **phrase-BL** and **phrase-extended**.

7.1.5 Interpretation of subsentence scores

Both word-level and phrase-level QE models return binary labels (“OK” and “BAD”) for individual words and phrases, respectively, and the probability of these labels. However, the N-best list reranking task requires a single label for a sentence. This means that subsentence-level scores should be combined into one score. In our experiments we will use two types of combinations for different outputs of QE models:

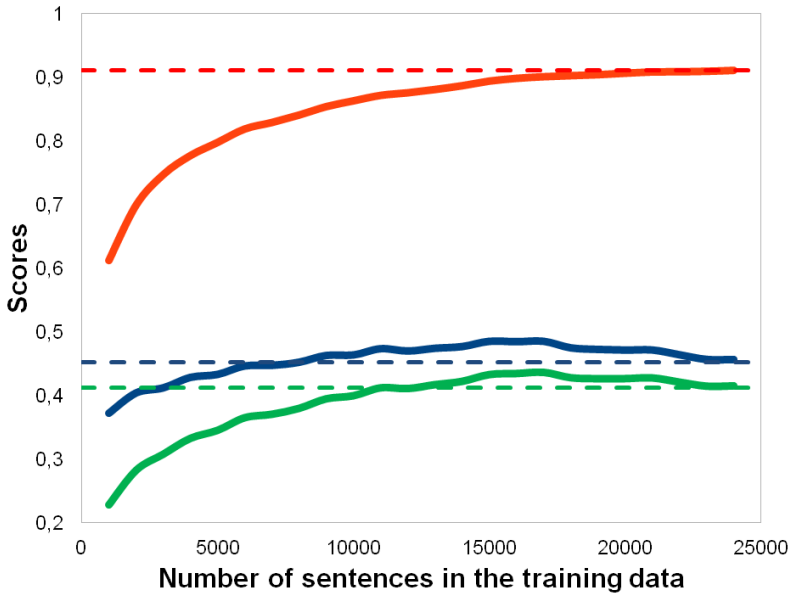

Fig. 7.5 Performance of phrase-level QE models trained on subsets of the training data for the German–English dataset. The solid lines show the scores for models which use subsets of the data of different sizes, the dashed lines show the scores for the model trained on the whole dataset.

- **probability**: average probability of words/phrases in a sentence having the label "OK",
- **score**: number of words / phrases with label "OK" divided by the number of words / phrases in the sentence — this quantity is equivalent to $1 - HTER$.

7.2 N-best list reranking

The SMT decoder places strict limitations on the features that can be used during translation: they have to be word- or phrase-level, because they have to be estimated at translation time when the sentence is not finished. That limits the use of syntactic, semantic or any other global features that can be extracted only from the complete sentences. This is also true for QE predictions, which are generated for complete translations (even if the predictions themselves are done for individual words or phrases). Thus, if we want an MT system to take into account any of this information, we need to include it at the stage when translations have already been generated. Moreover, the implementation of a new feature into an SMT system is often a laborious and time-consuming process.

Therefore, it is a common practice to apply the new feature to the list of N-best list of the test set in order to see if it can improve the quality of translations by re-ranking such a list to get a better translation at the top. The feature value is computed for every sentence from the

N-best list. Then it is combined with the model scores or other feature values via a linear combination. All the sentences from the N-best list are then reranked according to this new combined score. This process can identify a sentence in the N-best list which was not rated the best by the MT model, but is actually a better translation than the top one.

There are several examples of application of quality scores to N-best list reranking. The first such experiment was described by Blatz et al. (2004). The CE system used in the experiment failed to obtain any improvement over the baseline MT system, although the oracle experiment (selection of sentences from N-best lists according to their real quality scores under some metric) showed that there was room for improvement. The authors attributed this result to the fact that the target values in the dataset used for a CE system training were different from the metrics that were used to measure the performance of baseline and improved systems. However, these values were still correlated, because the labelling of the training data was done based on automatic metric values, and later the same metrics were used to evaluate the MT systems' performance and the effect of N-best list reranking. The poor result is probably due to the bad performance of their CE systems.

Later, quality scores were incorporated into an MT system via N-best list reranking by Luong et al. (2014d). The authors used a word-level QE model to produce binary quality predictions for every word. These predictions were then averaged to get a score for sentence. The improvement amounted to 0.5 BLEU points.

Our work extends the previous work: we conduct N-best list reranking using QE predictions of different levels (word, phrase, sentence) and experiment with a number of reranking strategies.

7.2.1 English–German experiments

All experiments described in this chapter were performed for the **English–German** language pair. The N-best list reranking experiments were performed on top of a baseline MT model trained on 3.3 million English–German sentence pairs from the IT domain produced by TAUS[3] (Translation Automation User Society). The corpus statistics is outlined in table 7.8). The training was performed with the `Moses` toolkit version 3.0 with the same settings as the ones we used in our AL experiments (see section 6.2.1). From now on all BLEU and METEOR values reported are computed for the test set using `Multeval` tool, the significance level is 0.05.

The reranking was produced on N-best lists of maximum length of 100 sentences, where all sentences in each N-best list are unique. We also tried the same experiments with the

[3] https://www.taus.net/

	Sentences	Words	
		English	German
Training	3,396,814	57,473,415	58,277,814
Development	2,000	33,010	35,084
Test	2,000	33,410	35,160

Table 7.8 Data used for training of the baseline SMT system.

larger N-best lists (up to 500 sentences), but the increased size of N-best lists did not change the results of experiments. This suggests that the best translations are already ranked correctly towards the top of an N-best list, and sentences which are rated low by an MT system's model score are unlikely to be superior.

Naive reranking

Our first set of experiments is the simple reranking of the N-best list, where the only information available to us is the set of N best translations for each sentence of a test set, and a model score for each of these translations. We can use the QE prediction instead of the model score and rerank the translations based on QE score alone, as it was done in early work (Blatz et al., 2004; Gandrabur and Foster, 2003). However, this strategy might be too aggressive: the performance of QE models is likely to be far from good enough to replace the model score. Therefore, another N-best reranking technique would be to combine the model score (*model*) and the QE prediction (*qe*):

$$score = model + qe$$

However, *qe* can be interpreted in different ways depending on the values which are predicted: word-level and phrase-level QE models predict sum of probabilities of "OK" labels or percentage of correct words in the sentence — both these values are higher for better sentence, whereas a sentence-level QE model predicts HTER scores (the higher the worse). Therefore, we replace the *qe* score with qe':

$$qe' = \begin{cases} \text{word-level and phrase-level QE:} & qe; \\ \text{sentence-level QE:} & 100 - qe. \end{cases}$$

Since the model score and QE prediction have different ranges, for the combination we will use scaled values:

$$score = model_{scaled} + qe'_{scaled},$$

where

$$X_{scaled} = \frac{X - \mu_X}{\sigma_X},$$

with μ_X and σ_X being the average value and the standard deviation value of X, respectively.

Table 7.9 shows the results of N-best list reranking with QE score and its combination with the model score. We can see that the use of QE scores as a replacement for model score deteriorates the performance, which means that QE score cannot be used as the only quality criterion. When combined with the model score, QE does not change the quality compared to the baseline. There are some exceptions: combination of model score and **sentence-17** model gives an improvement in METEOR (and deteriorates in BLEU), and both phrase-level QE models that compute probability scores are worse than the baseline and than their counterparts which return the percentage of bad phrases.

We provide oracle scores for the two metrics used (BLEU and METEOR). The oracles were generated as follows. We computed sentence-level BLEU and METEOR scores for every sentence in N-best lists and picked a translation candidate with the highest metric value for every N-best list. The oracles show the highest possible result of the N-best list reranking. Thus, the difference between the scores of oracle test sets and test sets rescored with our techniques shows the room for improvement.

Retraining of weights

The naive N-best list reranking mixes QE and SMT model scores giving them equal importance. We also performed a set of more advanced reranking experiments where the new score was generated by re-tuning weights for all features used for decoding and a QE feature. We performed weight retraining with `nbest-rescore`: the N-best list reranking tool available as part of Moses MT toolkit[4]. It first trains the weights of all features (the new ones as well as those used for decoding) on a list of N best translations for development set. Then, any other N-best list generated by the same MT system (i.e. the test set) can be rescored using a new set of weights. The training is performed with the Moses version of the K-batch MIRA algorithm. We run MIRA with the default parameters set in the `train.py` script of the `nbest-rescore` tool: it starts with all weights set to 0 and runs 300 iterations of the feature update process.

Since MIRA's performance is unstable (it outputs different feature weights in different runs) we repeated the training and reranking for 10 times and computed MT quality on concatenated outputs of the 10 runs.

[4]https://github.com/moses-smt/mosesdecoder/tree/master/scripts/nbest-rescore

System		BLEU	METEOR
Baseline		**25.4**	44.1
Only QE score			
sentence-17		22.4	44.0*
sentence-60		24.0	43.1
sentence-62-syntax		23.8	43.7
word-BL	score	24.9	43.9
	probability	24.1	42.8
word-extended	score	24.7	43.6
	probability	23.8	42.7
phrase-BL	score	24.5	43.6
	probability	23.3	42.6
phrase-extended	score	24.2	43.2
	probability	23.4	42.4
QE score + model score			
sentence-17		24.7	**44.4**
sentence-60		25.3*	44.1*
sentence-62-syntax		25.4*	44.3*
word-BL	score	25.3*	44.1*
	probability	25.4*	44.1*
word-extended	score	25.2*	44.0*
	probability	25.2*	44.0*
phrase-BL	score	25.3*	44.0*
	probability	24.9	43.8
phrase-extended	score	25.3*	44.0*
	probability	25.1	43.8
Oracle BLEU		35.0	51.7
Oracle METEOR		34.5	53.0

Table 7.9 Naive N-best reranking. Asterisk (*) denotes outputs which are **not** significantly different from the baseline.

System		BLEU	METEOR
Baseline (no reranking)		25.4	44.1
Baseline (reranking without new features)		24.4	44.9
sentence-17		25.2	44.9
sentence-60		25.6	44.7
sentence-62-syntax		25.5	**45.0**
word-BL	score	25.5	44.7
	probability	25.5	44.8
word-extended	score	25.5	44.8
	probability	25.5	44.8
phrase-BL	score	25.5	44.8
	probability	25.5	44.7
phrase-extended	score	25.5	44.8
	probability	25.6	44.8
sentence-62-syntax + word-BL score + phrase-BL score		25.4*	**45.1**
Oracle BLEU		35.0	51.7
Oracle METEOR		34.5	53.0

Table 7.10 N-best reranking with retraining of weights. Asterisk (*) denotes outputs which are **not** significantly different from the baseline.

The results of the weights retraining experiments are given in table 7.10. Unlike naive N-best reranking, this technique resulted in small yet significant improvement of quality: the predictions of some QE models were able to lead to better translations from the N-best lists.

The retraining of weights can be conducted without new features, so it is difficult to tell if the improvement of translation quality should be attributed to a new feature or to better weights of other features. In order to study the effect of re-tuning of weights in isolation, we performed N-best list reranking without new features. It returned a controversial result: the sentences chosen via this baseline reranking are significantly better than the baseline in terms of METEOR, but significantly worse in terms of BLEU. However, despite the high METEOR score, the **sentence-62-syntax** still slightly improves it. Therefore, the use of QE features for N-best list reranking can be considered an adequate method of incorporating PE in SMT.

In the results for reranking experiments with additional QE features we see the same discrepancy between the two metrics: while according to BLEU the use of QE predictions gives only minor improvements in quality (up to 0.2 points), the value of METEOR metric grows substantially for reranked test sets (up to 1 point). This means that the QE feature improves the unigram precision and recall which are the base for the METEOR metric and does not change the precision of ngrams of higher order which dominates the BLEU metric. Interestingly, the METEOR metric is better at capturing the improvements in the reranked test sets, although the reranking algorithm optimises weights with respect to BLEU.

sentence-17 is the only model that fails to improve on BLEU, the others improve over the baseline, but are not different from one another in terms of BLEU. The highest improvement in METEOR score is achieved by **sentence-62-syntax** and **sentence-17** models — they are significantly better than any other tested models. Differences between other models are not significant. This applies to **sentence-60** model and all word-level and phrase-level QE models: the results of reranking for baseline and extended models have identical weights despite the differences between them.

In addition to single QE features we experimented with combining the predictions at different levels and reranking the N-best list using better-performing models of the three levels of granularity (**sentence-62-syntax**, **word-extended**, **phrase-extended**). This experiment was done analogously to the ones with single QE features, but here instead of reranking weights with one additional features we used three additional features: one for each level of granularity. We combined subsentence-level predictions via the **score** technique, because according to our naive reranking experiments it performs better than the **probability** combination. This combination of three QE features performs closely to single features.

7.2.2 Analysis

According to the results of the N-best list reranking experiments, the retraining of weights with information from a QE model brings significant although small improvements regardless of the QE model type. However, other methods of incorporation of QE predictions at the stage of N-best list reranking are ineffective. We would like to understand what the success (or lack thereof) is attributed to.

Naive reranking vs retraining of weights

First of all, our experiments showed that N-best list reranking where all features were reweighed with respect to a new feature (quality predictions) were more effective than naive reranking where the new feature was combined with the model score with equal weights or just replaced the model score. This shows that QE models are still not accurate enough to be used for quality assessment on their own. This is expected as the model score exploits much more information.

The unsuitability of raw QE scores for ranking of translation suggestions is confirmed by correlation scores between different rankings of sentences in the N-best list and their oracle order. Table 7.11 shows the rank correlation scores between N-best lists sorted according to BLEU and METEOR scores and (i) baseline N-best list (ii) N-best list reranked via retraining of weights (iii) N-best list reranked according to the quality of sentences predicted with a QE model. The figures were close for all QE models, so we report the average scores. While the original (baseline) N-best list has low but positive correlation with both oracles, after reranking this correlation drops. Furthermore, if we compare the oracle rankings with the one produced with quality predictions, we will see almost no correlation. For the N-best list reranking task we do not need to identify the right order of the sentences — it is enough to pick a sentence which is better than the baseline top translation, but this still indicates the fact that the predictions from state-of-the-art QE models cannot be used as a replacement for MT model scores.

	BLEU	METEOR
Baseline N-best list	0.142	0.118
Reordered N-best list	0.088	0.074
QE predictions	-0.015	-0.014

Table 7.11 Rank correlation scores between original, reranked and QE-based rankings of sentences in N-best lists and the N-best lists sorted according to METEOR and BLEU scores.

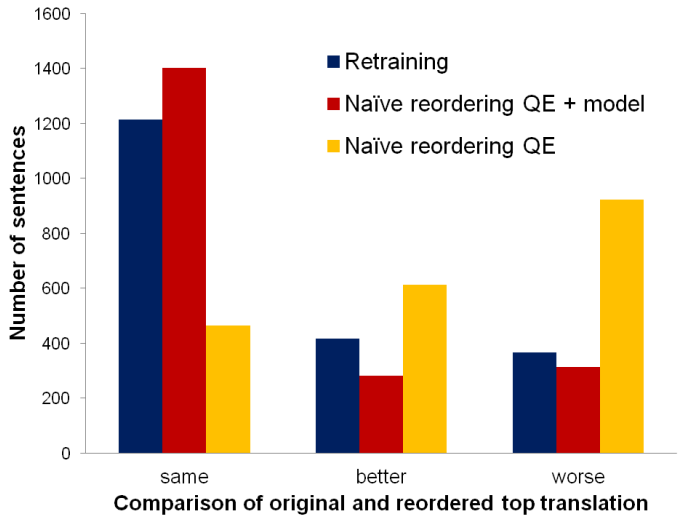

Fig. 7.6 Comparison of the numbers of improved/worsened/unchanged sentences for different reranking techniques.

However, the absence of correlation should not be taken as an ultimate evidence of the QE models' low informativeness, because the automatic evaluation metrics sometimes fail to define a better translation in the list of similar candidates, or tend to downweigh a good translation if it does not contain words from the reference. This is particularly true when adapting document-level metrics for the evaluation of individual sentences. This could instead be an indication that the different information types are complementary.

The most notable differences of three N-best list reranking techniques are the percentage of sentences of the test set that were changed and the nature of this change. Figure 7.6 gives an overview of these statistics for the three techniques. When reranking the N-best list with retrained feature weights, in around 60% of cases the order in a 2000-sentence test set is not changed — i.e. the top-1 sentence defined by the baseline MT model score is also considered the best according to the combination of re-weighted features. On the other hand, when using QE predictions *in lieu* of the original model scores, this quantity drops to just 400 sentences (20%). Finally, the combination of QE predictions with the original model scores is the least aggressive: it leaves almost 3/4 of sentences untouched. As for the cases when sentences were changed (more specifically, replaced with another translation from an N-best list), the new versions can be better or worse than the baseline top-1 translations according to sentence-level BLEU or METEOR. The comparison of quantities thereof shows that only the weights retraining strategy yields more improved sentences than those worsened. Both naive reranking strategies deteriorate translation more often than raise its quality.

This explains the differences in final scores. Retraining of weights brings small improvements because there are slightly more sentences that got better after reranking. Reordering

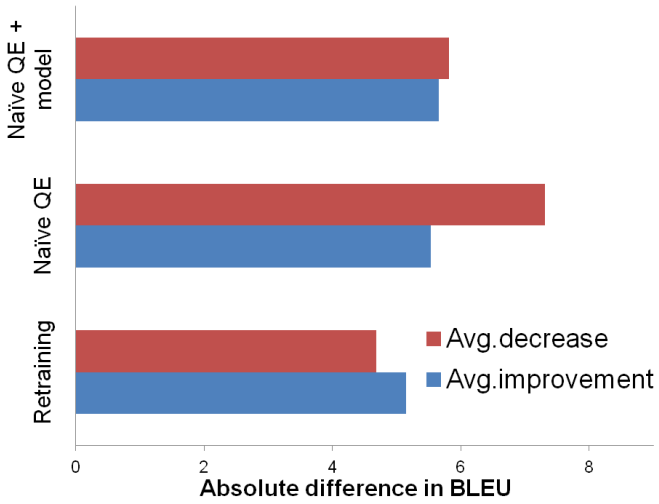

Fig. 7.7 Average absolute increase/decrease in sentence-level BLEU scores after reranking for the three N-best reranking techniques: naive reranking with QE predictions, naive reranking with a combination of QE predictions and SMT model score, reranking with retraining of feature weights.

based on only QE scores reduces the final BLEU and METEOR scores, because it worsens almost half of the sentences. Conversely, when QE scores are combined with the model scores, the numbers of improved and worsened sentences are small and very close — that is why it does not change the scores significantly in the majority of cases.

On the other hand, the number of improved and worsened sentences is not the only factor that influences the scores. The sentences can be replaced with their very similar alternatives or become completely different, thus bringing small or large increase or decrease of scores. Therefore, we compared the absolute values of increase and decrease of BLEU scores for the sentences which were improved and worsened, respectively (see figure 7.7). The comparison agrees with the result we demonstrated above: naive reranking with QE score is the most "aggressive" strategy as it leads to the largest deterioration. Analogously, naive reranking with the combination of QE and model scores damages sentences at the same rate as it improves them. Finally, the retraining of weights again proved a better reranking technique, because it is the only technique for which the degree of improvement is higher than that of deterioration.

Sentence-level vs subsentence-level QE

We used features from QE models of two types: those predicting quality of entire sentences and of smaller units — words and phrases. According to our experiments, the sentence-level QE models yield slightly higher scores in the N-best list reranking with the retraining of weights. The comparison of the number of unchanged/improved/deteriorated sentences (see

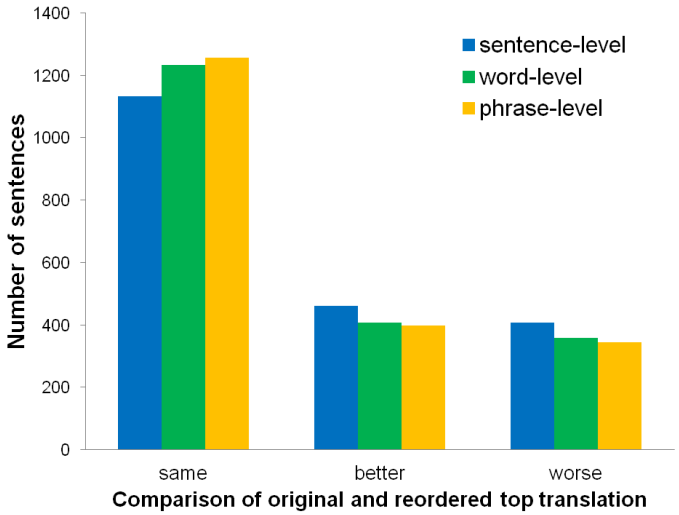

Fig. 7.8 Comparison of number of sentences which were improved/worsened/left unchanged by QE models of different levels of granularity.

figure 7.8) showed that sentence-level strategies are more "aggressive" in the sense that they leave less sentences untouched.

In all our experiments we use sentence-level scores: sentence-level QE models generate sentence-level HTER scores directly, and when working with subsentence-level models we combine scores for words (phrases) to get a single score for a sentence. We combine them in two different ways: (i) compute the average probability of words/phrases in a sentence having the label "OK" and (ii) compute the percentage of words/phrases having the label "OK". The latter way (referred to as **score**) is essentially an inverted HTER score, therefore, we can directly compare the predictions by QE models of different levels.

Table 7.12 gives such a comparison: we list average values of predicted HTER scores for sentence-level models along with the predictions by subsentence-level models combined with the **score** technique and distracted from 100 so that they show the average percentage of "BAD" words in a sentence. This comparison does not give us any clear picture. While the average values sentence-level models' predictions range from 25% to 28%, the subsentence-level models are more diverse. The baseline word-level QE model is more "optimistic", whereas its extended version approaches the sentence-level scores. The baseline phrase-level model is also more "optimistic" than its extended counterpart, but phrase-level models are more "pessimistic" in general: the extended phrase-level model predicts higher (worse) scores than the sentence-level models.

Model		Average predictions (HTER)
Sentence-level	sentence-17	25.22
	sentence-60	27.50
	sentence-62-syntax	28.20
Word-level	word-BL	15.78
	word-extended	24.27
Phrase-level	phrase-BL	27.50
	phrase-extended	36.32

Table 7.12 Comparison of average HTER scores predicted by QE models of different levels of granularity.

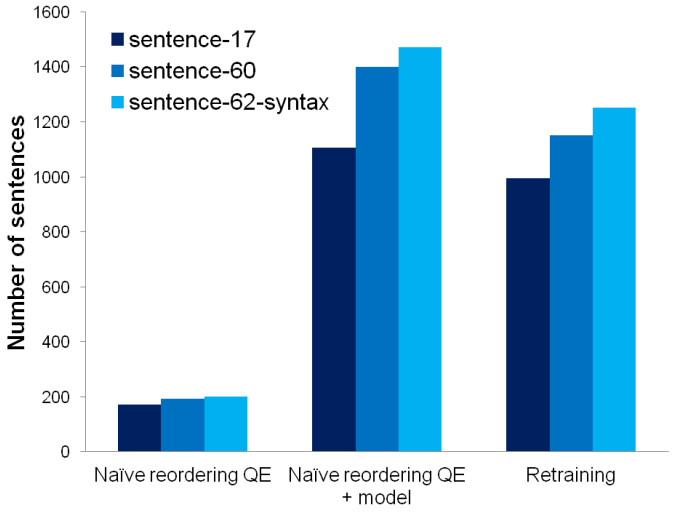

Fig. 7.9 Number of sentences which were left unchanged by reranking techniques that used sentence-level QE features.

Sentence-level models

In addition to differences in the performance of QE models of different levels, the three tested sentence-level models vary in their performance. In the N-best reranking with the reranking of weights, **sentence-60** and **sentence-62-syntax** outperform the baseline in terms of BLEU, whereas **sentence-17** is below the baseline. On the other hand, **sentence-17** along with **sentence-62-syntax** is better than **sentence-60** in terms of METEOR, although the latter is still above the baseline.

We looked into the differences between the models and discovered that the models that showed higher F_1-mult scores on a held-out test set tend to be less invasive when used in N-best list reranking experiments: they leave more sentences unchanged (see figure 7.9). This suggests that predictions from a better-performing model correlate better with the model

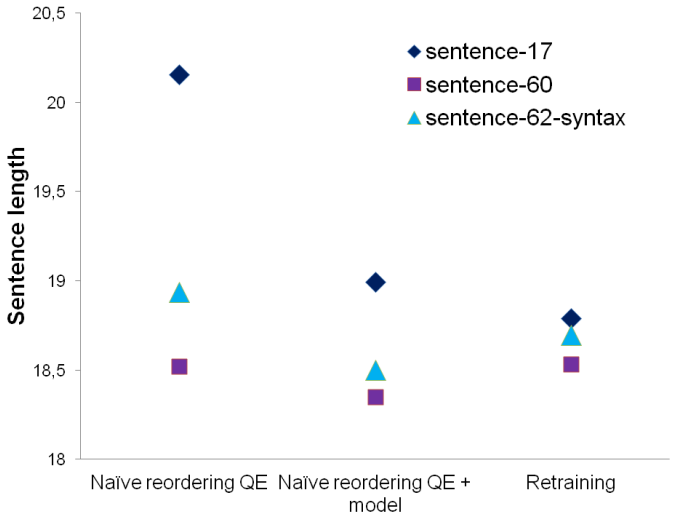

Fig. 7.10 Average lengths of sentences chosen by different sentence-level QE models.

scores which are used by MT system to sort translation suggestions. Another confirmation of this fact is the length bias that is demonstrated by the **sentence-17** model: it always tends to choose longer sentences, whereas translations picked by **sentence-60** and **sentence-62-syntax** are on average not different from the baseline translations in terms of length. Figure 7.10 demonstrates the comparison of average lengths of translations selected by the three sentence-level models.

Subsentence-level models: baseline vs extended

Analogously to sentence-level models, the word-level and phrase-level models we used for our experiments were also presented in different variants: we used baseline models and better-performing extended models. We wanted to compare their performance in N-best list reranking in order to see to what extent the differences in their quality influence the downstream task. Surprisingly, the corresponding baseline and extended QE models returned very close scores for all tasks.

The baseline and extended models were also very close in terms of various statistics: the percentage of unchanged, improved and worsened sentences, average difference in BLEU score for improved and worsened sentences. The only difference which was already reflected in table 7.12 is the fact that baseline models tend to be more "optimistic": there, they consistently predict lower HTER values. The same trend holds for predictions combined with **probability** technique. Figure 7.11 shows the distributions of predictions by different subsentence-level QE models. The distributions are presented in the form of box plots: the box is limited from the bottom and the top by the lower (25% data points) and upper (75%

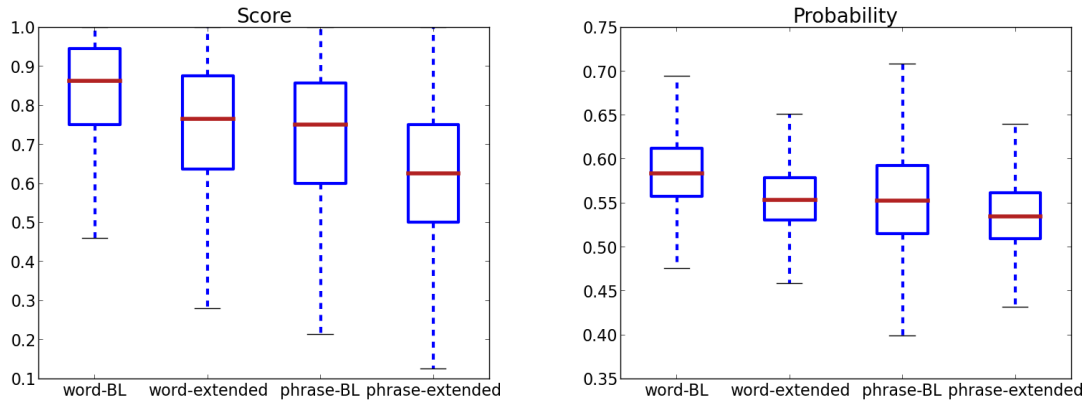

Fig. 7.11 Distributions of predictions made by baseline and extended subsentence models and combined via **score** (left) and **probability** (right) techniques.

data points) quartiles, respectively, and divided by the median (50% data points). It can be clearly seen that the baseline models give higher scores regardless of the way they were combined (here, unlike the HTER score demonstrated before, higher scores mean better quality of sentences). Besides that, the predictions of extended models are less diverse when combined with the **probability** technique.

Subsentence-level models: probability vs score

The subsentence-level models also differ in another aspect: we either convert their predictions into average probability of words in a sentence being correct (**probability**), or the percentage of correct words in a sentence (**score**). The differences between these two types of scores are greater than the differences between the baseline and extended subsentence-level models.

First, as shown by our naive reranking experiments, the **probability** QE models perform significantly worse than the corresponding **score** models. As figure 7.11 shows, the predictions made by **probability** models are less diverse: they span only from 0.4 to 0.7, whereas the predictions by **scores** models often cover more of the allowable range ($[0, 1]$). Therefore, we can assume that the **probability** models are worse at discriminating between similar sentences.

However, further analysis shows that despite their inability to distinguish between better and worse sentences, the **probability** models are more aggressive than the **score** ones: they change more sentences (see figure 7.12). While the number of improved sentences is greater compared to the **score** models, the number of deteriorated sentences is larger. The "aggressiveness" is also reflected in the improvement and and deterioration in changed

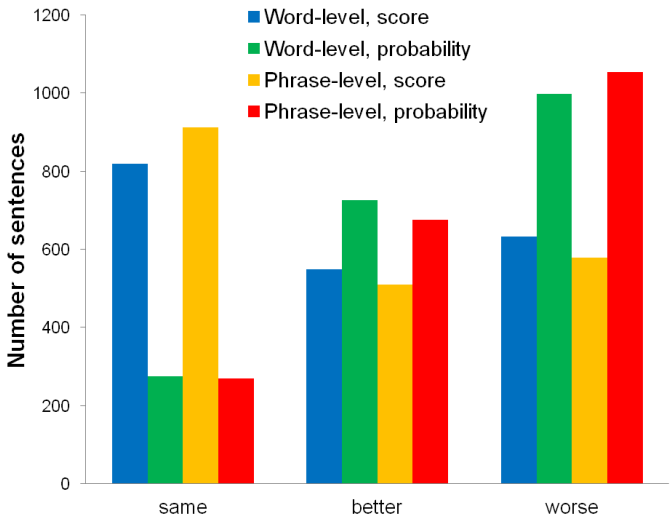

Fig. 7.12 Comparison of number of sentences which were improved/worsened/left unchanged by subsentence-level QE models which were combined using **probability** and **score** techniques.

sentences: the absolute differences between the BLEU scores of baseline and changed sentences are larger for the **probability** models (see figure 7.13).

Another difference between **score** and **probability** models is that the latter are biased towards shorter sentences: figure 7.14 shows that the average length of sentences selected with **probability** models is lower in almost all cases. Note that all these statistics (number of improved/worsened/unchanged sentences, increase/decrease in sentence-level BLEU scores) were reported for the naive reranking experiments where the sentences were reranked based on QE scores alone. The test sets generated via reranking based on the combination of QE and model scores follow the same trends, but in the experiments with QE scores they are more salient.

We should highlight that neither the "aggressiveness" of **probability** models nor their preference for shorter sentences hold for reranking of weights: we noticed that there subsentence-level QE features which contained information about probability were given very low weights (see table 7.13). In the table we also see that the features that come from phrase-level QE models are given lower weights in general, regardless of their type. This might indicate the low reliability of such features, which agrees with our previous observations of phrase-level QE models: they underperform the word-level QE models in the majority of cases.

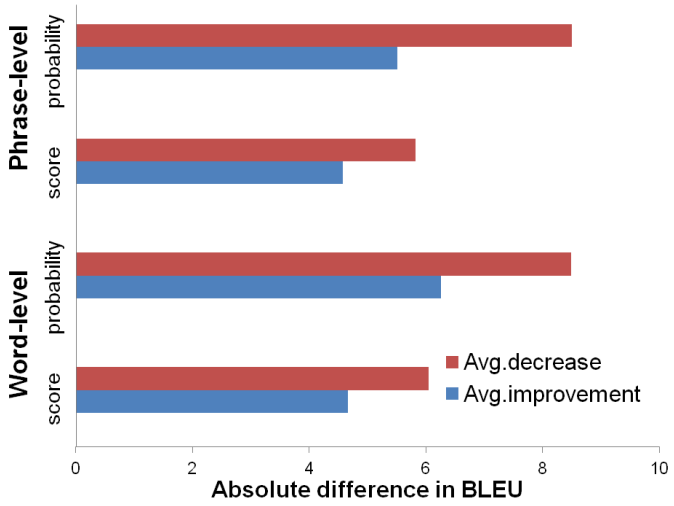

Fig. 7.13 Average absolute increase/decrease in sentence-level BLEU scores after naive N-best list reranking with QE predictions made by word-level and sentence-level **probability** and **score** QE models .

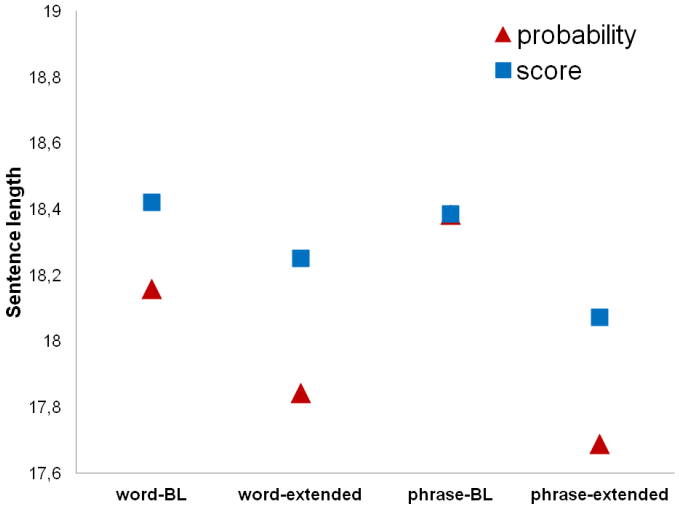

Fig. 7.14 Average lengths of sentences chosen by subsentence-level QE models which used **probability** and **score** combination techniques.

	Probability	Score
word-BL	0.025	0.204
word-extended	0.028	0.253
phrase-BL	0.003	0.022
phrase-extended	0.019	0.04

Table 7.13 Weights assigned to QE features in the N-best list reranking experiment with retraining of weights.

Experiment	BLEU	METEOR
Baseline	**25.4**	44.1
Naive reranking: only QE	22.4	44.0*
Naive reranking: QE + model score	24.7	**44.4**
Retraining of weights	25.2	**44.9**

Table 7.14 Results of reranking experiments with the QE feature generated by **sentence-17** model. Asterisk (*) denotes outputs which are **not** significantly different from the baseline.

METEOR vs BLEU

We used two evaluation metrics: METEOR and BLEU, in order to make the results more general. The two metrics are consistent in the majority of cases. However, sometimes they disagree: a low BLEU score appears alongside a high METEOR score. For example, the use of **sentence-17** model predictions for naive reranking in combination with the model score gives a BLEU score which is significantly lower than the baseline (24.7 vs 25.4) and a METEOR score which is significantly higher than the baseline (44.4 vs 44.1).

As we have already seen, the **sentence-17** model has the length bias: it favours longer sentences. Therefore, in all reranking experiments the average length of sentences it chooses is higher than that of other QE models. The combination of low BLEU and high METEOR also occurs in all experiments with this QE model (see table 7.14).

The reason for such a result is the difference in the formulations of BLEU and METEOR. The BLEU score is based on the precision of ngrams, in other words, it computes the percentage of ngrams in an automatic translation which match ngrams in the reference. Conversely, METEOR is is a combination of ngram recall scores, so it is concerned about the percentage of ngrams in the reference that are used in a translation. Therefore, increasing the length of a translation increases the probability of ngrams from the reference occuring there. So the cases where a low BLEU is accompanied with a high METEOR should be attributed to the length bias.

7.2.3 German–English experiments

In order to investigate whether our positive results on N-best list reranking generalise to other data, we replicated the N-best list reranking experiments on a different dataset: it contains medical data translated from German into English.

Our baseline MT model was also trained on a parallel German–English set of medical texts containing 1.7 million sentences. The data statistics is outlined in table 7.15. The baseline model was trained with the same settings as those which we used in our AL experiments and N-best reranking experiments with the English–German data (see section 6.2.1).

	# of sentences	# of words	
		German	English
Training	1,777,000	71,138,000	79,403,000
Development	2,100	49,000	53,800
Test	2,100	49,200	54,100

Table 7.15 Statistics of the parallel German–English dataset of pharmaceutical texts used to train the baseline MT system.

Results

In our previous experiments we saw that naive reranking (reranking based on QE score alone or its combination with model score with equal importance) is unable to improve the translation quality. Therefore, in our experiments with the German–English data we conduct only N-best list reranking with the retraining of weights.

The results of these experiments are outlined in table 7.16. They are strikingly different from the ones we achieved for the English–German data. Here, in contrast with the previous experiments, the QE models fail to improve upon the baseline. Additionally, they are not different from one another. In fact, the scores we get for all the models: 61.8 BLEU and 45.8 METEOR — are essentially the same as the scores of the test set rescored with the same technique but without additional features. This means that QE features were not helpful in these experiments, because adding them turned out to be equivalent to not adding anything.

Analysis

Low importance of QE features This observation is confirmed by the weights of QE features that are returned by the tuning algorithm. Table 7.17 shows that these weights are often very close to zero, which means that the share of QE feature in the overall model score

System		BLEU	METEOR
Baseline (no retuning)		**63.9**	**46.8**
Baseline (retuning without new features)		61.8	45.8
sentence-17		61.8	45.8
sentence-60		61.8	45.8
sentence-62-syntax		61.8	45.8
word-BL	score	61.8	45.8
	probability	61.8	45.8
word-extended	score	61.8	45.8
	probability	61.8	45.8
phrase-BL	score	61.8	45.8
	probability	61.8	45.8
phrase-extended	score	61.8	45.8
	probability	61.8	45.8
Oracle BLEU		67.6	48.7
Oracle METEOR		67.1	49.1

Table 7.16 N-best reranking with retraining of weights, German–English dataset.

is very small. We also show the feature weights from the English–German experiments for comparison — it can be clearly seen that in the majority of cases they are much greater in absolute values. Although we cannot draw any conclusions without seeing the weights of other features, such difference indicates the greater importance given to the QE feature in the English–German experiments.

As a consequence of low weights assigned to QE features, the reranking left the majority of sentences of the test set unchanged: it fluctuates from 70% to 80% for sentence-level QE models and is 80% for subsentence-level models. In the English–German weights retraining experiments these figures were lower: on average 56% for sentence-level QE models and 61% for word-level and phrase-level QE.

High quality of MT systems A notable feature of this set of experiment is the high performance of the baseline MT model: its BLEU score is considerably higher than the usual values. For example, compare the score of the baseline MT model trained on our English–German IT data (25.4) and the same score for the German–English medical data (63.9). These scores cannot be directly compared, because they were obtained for different test sets and even different language pairs. However, the range of the BLEU score for any dataset is $[0, 100]$, and the value of 60 indicates high quality of translation.

On the other hand, the METEOR score does not show the changes which we see in BLEU: its score is almost the same for our English–German and German–English MT systems (44.1

QE models		QE feature weight	
		German–English	English–German
sentence-17		0.0043	-0.0834
sentence-79		-0.0307	-0.0246*
sentence-50-syntax		0.0249	-0.0615*
word-BL	probability	0.0048	0.0255
	score	0.008	0.0424
word-extended	probability	0.001	0.028
	score	0.0259	0.0873
phrase-BL	probability	-0.0089	0.0034
	score	-0.0216	0.0226
phrase-extended	probability	-0.009	0.0191
	score	-0.0295	0.0405

Table 7.17 Weights of QE features in German–English and English–German experiments. * the **sentence-79** and **sentence-50-syntax** models were not used in English–German experiments, so here for English–German we report the weights of similar models: **sentence-60** and **sentence-62-syntax**.

vs 46.8 for the baseline systems). This difference (as well as sensibility to the length of automatic translations that we discussed before) stems from the difference in formulations of METEOR and BLEU: the former considers only unigrams, whereas the latter takes into account ngrams of the orders 1 to 4.

In order to get a more comprehensive evaluation of the systems' results we compute the value of a third metric, namely, the TER score (Snover et al., 2006). The TER scores for the baseline English–German and German–English MT systems are **58.5** and **25.3**, respectively. Note that TER scores **lower** values denote higher translation quality. Therefore, TER agrees with BLEU.

Data repetitiveness Such high translation quality is usually due to the large size of the MT training corpus or the high repetitiveness in the training data. The MT model was trained on 1.7 million sentences, which is smaller than the IT corpus used in the English–German experiment, so the most likely explanation is the high repetition rate of the medical data. We compared the type-token ratio (the number of unique words per token in a corpus) and the analogous quantity for ngrams of higher orders (the number of unique bigrams, trigrams and fourgrams per token in a corpus) for the German–English medical data and two other datasets: English–German IT corpus we used in our previous experiments and the German–English part of Europarl corpus (Koehn, 2005). The type-token ratio of the medical data is larger than that of other datasets (see table 7.18), which means that the medical data is more diverse.

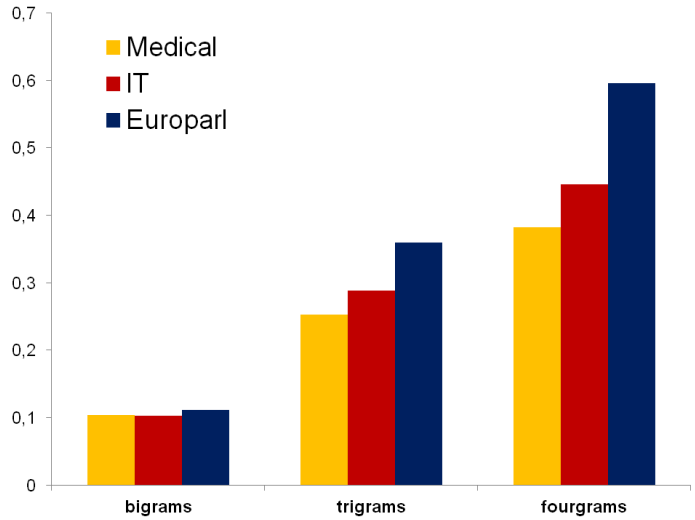

Fig. 7.15 Number of bigrams, trigrams and fourgrams per token (number of unique ngrams divided over total number of words in a corpus) in the German parts of three parallel datasets: pharmaceutical dataset, dataset of IT texts and proceedings of the European Parliament. Higher values indicate lower repetitiveness.

However, if we check the number of ngrams per token, we will see a different picture. Figure 7.15 shows the statistics for bigrams, trigrams and fourgrams for the three datasets. We see that the medical corpus is more repetitive than both Europarl and IT, with the latter being more repetitive than the former. This is likely to be a reason of the N-best list reranking having little effect on this data.

	Pharmaceutical	IT	Europarl
German	0.016	0.012	0.008
English	0.004	0.002	0.002

Table 7.18 Type-token ratio for the three datasets (higher value means higher diversity).

7.3 Retuning with an additional feature

In our N-best list reranking experiments we tuned the feature weights taking into account new features which could not be considered at decoding time. However, there the weight tuning is performed only once: the sentences of the development set were decoded with the baseline feature weights, then the MIRA algorithm generated new values for every feature weight such that the overall model score ranked sentences in all N-best lists similarly to their oracle order (i.e. their order according to an evaluation metric). Then the test sentences

(also initially decoded with the set of baseline feature weights) were re-weighed and resorted accordingly.

Such reranking has limitations: the new feature(s) can only choose the better sentences from the N-best list, but the list of translation variants stays the same, and if it does not contain good translations, the overall quality cannot be improved. On the other hand, if instead of a single update we perform tuning of an MT system, we could in theory get sentences which are closer to the ground truth. The difference from the conventional tuning process which is used in MT systems is that here we use a sentence-level QE feature that does not participate in the decoding in addition to the set of baseline features.

We run the tuning experiments only for the English–German datasets. The tuning experimental setup is similar to the one of N-best list reranking described in the previous section. Here we used the same baseline MT model as in the reranking experiments, and the retuning itself was performed with the same `nbest-rescore` scripts. Tuning essentially involves the same N-best list reranking procedure repeated multiple times.

We experimented with two initial weights settings:

- **tuned**: weights of baseline features are pre-trained with the MIRA algorithm,
- **untuned**: weights are initialised with the initial values used in Moses as a starting point. These values are:
  - WordPenalty: -1
  - PhrasePenalty: 0.2
  - Translation Model: 0.2, 0.2, 0.2, 0.2
  - Lexical reordering model: 0.3, 0.3, 0.3, 0.3, 0.3, 0.3
  - Operation sequence model: 0.08, -0.02, 0.02, -0.001, 0.03
  - Distortion: 0.3
  - Language Model: 0.5

In both cases the weight of the QE feature is initially set to 0.1.

We used the same QE models: 3 sentence-level models of different quality, 2 word-level models (baseline and extended), each having two varieties: the one that predicts average probability of words in a sentence having label “OK”, and the one combining word-level predictions into sentence-level HTER score, 2 phrase-level models with the same two varieties. For each QE model we performed two retuning experiments: we started tuning with the initial feature weights, or pre-tuned them without including the QE feature. This gives the total of 22 experiments. We present the results of all these experiments as figures for illustrative purpose. The BLEU and METEOR scores for all runs can be found in the section B.2 of appendix B.

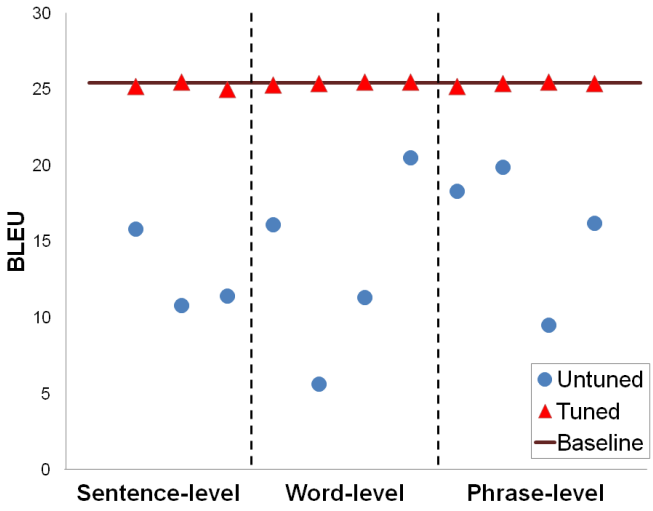

Fig. 7.16 BLEU score for the English–German MT models retuned with the new word-level, phrase-level or sentence-level feature.

Figure 7.16 shows the BLEU scores of MT models with an additional QE feature and retuned weights, figure 7.17 shows the METEOR scores of the same models. The brown lines show the performance of baseline MT systems, i.e. systems that do not use QE feature and do not have additional tuning rounds.

It can be seen that the **tuned** scenario is not different from the baseline performance or improves upon it marginally (in terms of METEOR), whereas the performance of **untuned** MT models is significantly worse than the baseline. The behaviour of the tuning algorithms in these two scenarios was also different. For cases when the tuning started with pre-tuned weights, the training stopped after 3-4 iterations; when starting from the initial weight values, the weight update process did not converge — it stopped when the maximum number of iterations was reached (30 in our experiments).

Early stopping in the case of pre-tuned weights means that the weights had already reached the optimum and could not be further optimised even with a new feature added. On the other hand, the deterioration of quality in the **untuned** scenario indicates its unsuitability. Here, we tune weights in the same manner as we do in the baseline MT system, but we repeatedly force the optimisation algorithm to take into account a parameter that does not contribute to generation of translations (QE model prediction). The QE feature does not participate in the decoding, but the weights of other features are optimised with respect to its weight as well, which makes the weights of the baseline features less effective for decoding. Therefore, the translations of the development set are repeatedly generated with suboptimal parameters, thus further degrading the quality of translations.

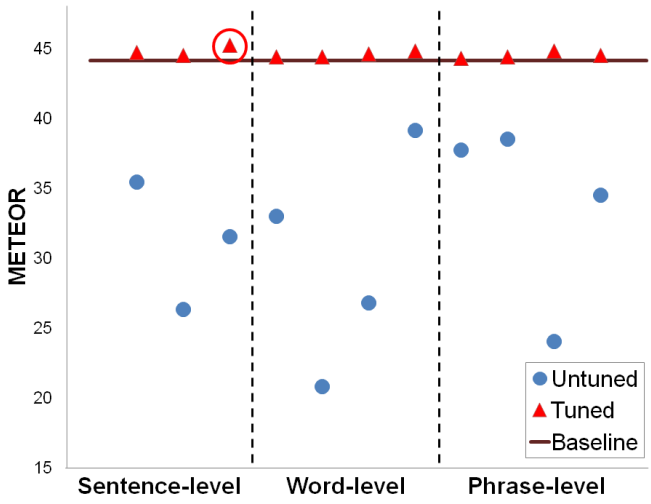

Fig. 7.17 METEOR score for the English–German MT models retuned with the new word-level, phrase-level or sentence-level feature.

We did not perform the experiments with the German–English data, because they were highly unlikely to improve the baseline scores. As we saw in the previous section, the N-best list reranking was more effective for the English–German dataset than for the German–English one. By analogy, we assume that the re-tuning on the German–English data is unlikely to give better result than on the English–German data. Besides, the retuning is produced by the same `nbest-rescore` tool that was used for the N-best reranking and did not give any positive result on the German–English data.

7.4 QE score as a decoder feature

The strategies tested in sections 7.2 and 7.3 have a major drawback: despite taking into account the quality score, they all applied it only *after* the sentence was generated. Hence, the new features cannot influence the generation process, and if translation variants which could have high QE scores are pruned during decoding, the incorporation of QE cannot improve the MT output. On the other hand, if the QE scores can be taken into account during decoding, they could help keep some good translations which would otherwise be filtered out.

The main difficulty with the incorporation of new information into an SMT decoder is the sequential nature of decoding process. While the source sentence is not necessarily translated linearly (translation can start from any source word, skip words and then come back to them), the generation of the output is always performed from left to right. This imposes constraints

on the information that can be used for the decoder features: not all the information on the translation is available at decoding time. Because of that, there are two main challenges that we need to take into account:

- When computing a word-level (or phrase-level) feature, the decoder has **only the information on the left context** of this word (phrase) because it only knows what was translated before, but not how the next words will be translated. On the other hand, the entire source sentence is available at decoding time.
- The **length of left context is limited**. During decoding, the alternative translations are stored in a translation lattice: a directed graph where each node contains a word/phrase and each path from the start to the end node of such graph is a translation variant. In order to store more variants the partially matching hypotheses can be joined into one. This process requires "forgetting" some of the features of words/phrases of the hypotheses. Keeping too long context (i.e. making the decoder "remember" features of all generated words) can result in the translation lattice not being able to store many hypotheses, because they no longer can be joined or recombined. This can dramatically decrease the MT quality, because many promising hypotheses will be discarded early due to the lack of space. There is no precise answer to the question about the longest allowable context, however, the 5-gram LMs are generally considered a good trade-off and their use does not deteriorate the MT models performance. Therefore, we suggest that a reasonable context-dependent feature should not require context longer than 4 words.

Based on that, in order to compute word-level quality scores during decoding we should make sure that:

- they do not require right target context,
- they do not require left target context of longer than 4 previous words.

7.4.1 Word-level QE model

The first requirement to the context can be accomplished via filtering out features that are not available for partial translations. As for the second requirement, we have already addressed it in section 4.1.2 where we suggested training a word-level QE model on partial sentences in order to enable it to recognise erroneous and correct words in an unfinished sentence. While our experiments showed that this transformation of the data does not improve the performance of QE models, they also showed that unfinished sentences are just as well labelled with a QE

model trained on a regular dataset. Therefore, we assume that a conventional QE model is suitable for predicting the quality of unfinished sentences provided that it does not use the right context.

We could not use syntactic features, because they require information about the whole sentence, which we do not have during decoding. Besides that, we had to remove source and target LM features because their extraction takes too much time and significantly slows down decoding since when feature extraction has to be performed millions of times (for every hypothesis). Therefore, our word-level QE model which is used for generation of prediction at decoding time contains only 15 features:

- target token,
- left context of the target token,
- source token,
- left context of the source token,
- right context of the source token,
- target token is a stopword
- target token is a punctuation mark,
- target token is a proper noun
- target token is a digit,
- target token POS-tag,
- source token POS-tag,
- target token + left context,
- target token + source token,
- target POS-tag + source POS-tag,
- target token + left token + source token.

Note that while the target sentence features can only use left context, the source features do not have such limitation because we have the access to the whole source sentence at any time.

As expected, the inability to use the majority of features affected the QE model scores: the F_1-mult score dropped to 0.309, which is below the baseline for the word-level QE task. A comparison between the reduced model and the baseline are given in the table 7.19. The changes affect mainly the model's performance on the "BAD" class: the reduced model is more "optimistic": it finds almost twice as few "BAD" words than the baseline model, and its F_1-BAD score is significantly lower, whereas F_1-OK is slightly higher.

The German–English word-level QE model uses the same reduced set of features that do not need sentence-level information or information about the right context. Analogously

	F_1-BAD	F_1-OK	F_1-mult	Number of "BAD" words
Baseline	0.368	0.879	0.324	6668 (19.3%)
Reduced	0.349	0.884	0.309	3724 (10.7%)

Table 7.19 Comparison of the word-level QE model for decoding with the baseline QE model for the English–German IT dataset.

to our previous experiments, we trained the word-level QE model with `CRFSuite` using the passive-aggressive optimisation algorithm. The performance of our reduced model and its comparison with other word-level models is presented in table 7.20.

	F_1-BAD	F_1-OK	F_1-multiplied
reduced model	0.308	0.936	0.289
word-BL	0.374	0.939	0.351
word-extended	0.455	0.932	0.424

Table 7.20 Performance of the reduced word-level QE model for the German–English dataset and its comparison with the full models.

7.4.2 Implementation of quality score features

We use Moses MT toolkit for our experiments. The baseline MT setup includes 14 decoder features (see section 2.2.1), and new features can be added. The `StatefulFeatureFunction` and `StatelessFeatureFunction` classes provide interfaces for features that use or do not use information about the previous phrase, respectively. All new feature functions can inherit from either of these classes [5]. Since the computation of quality scores involves contexts (previous words), we used `StatefulFeatureFunction`.

We implemented a new feature named `WordQEFeature`[6]. It evaluates every phrase at decoding time (i.e. when this phrase is added to the translation hypothesis). It is a *stateful* feature function, so the evaluation procedure receives the previous *state* for the hypothesis. The state for the new feature function is implemented in the class `WordQEState` and contains the following information about the hypothesis: list of words in the previous phrase, list of features for the words of the previous phrase, and alignment information for the previous phrase. All this information is used to extract the feature for the current phrase. The words of the current and previous phrases are joined into a sequence which is passed to the tagger

[5] A detailed descriptions of how to add new features can be found on: http://kentonmurray.com/blogs/addingafeaturetomoses.html

[6] The implementation is available at https://github.com/varvara-l/mosesdecoder

object. The tagger produces word-level scores. Their sum forms the feature value for the phrase.

The **WordQEFeature** can extract two types of scores for words:

- **probability**: the probability of having the label "OK";
- **score**: 0 if the word is labelled as "BAD" and 1 if labelled as "OK".

The probability and score values are interdependent. The score is represented as a binary value, and the probability takes values from 0 to 1. Therefore, a word has the score of 1 if the probability of the "OK" label for this word is greater than some pre-defined threshold (usually 0.5). However, we expect that the use of fine-grained (probability) and coarse-grained (score) scales can have different effect on the final MT quality.

The feature is added to the standard Moses configuration file in the section `[feature]` in the following form:

```
WordQEFeature name=WordQE stopwords=/path-to-target-stopwords-list
probability=1 model=/path-to-pretrained-qe-model
```

The `WordQEFeature` uses the following parameters:

- `name` — string (optional). Alias for the feature name used throughout the configuration file. If omitted, the feature name (`WordQEFeature`) is used instead.
- `stopwords` — string. Path to the list of stopwords for the target language. In our experiments we use the stopword list provided in NLTK.[7]
- `model` — string. Path to the pre-trained `CRFSuite` QE model.

In addition to that we should add the initial feature value to the section `[weight]` of the configuration file:

```
WordQE= 0.1
```

The QE model is pre-trained with `CRFSuite` toolkit. `CRFSuite` has API for C++[8] which makes it easy to use: the API provides classes for objects and features, class for the model and the possibility to load it from file. The pre-trained model is loaded into an instance of the API's `Tagger` class once when the decoder is launched. At decoding time features are saved to `Attribute` objects. A set of features for one word is saved to an `Item` object, a set of `Items` is assembled to an `ItemSequence` object which can be labelled by the `Tagger`.

[7] http://www.nltk.org/api/nltk.corpus.html
[8] http://www.chokkan.org/software/crfsuite/api/

The extraction of features for the QE training set is performed with `Marmot` toolkit (see section 3.4.2). The extraction of features for phrases at decoding time is not performed by `Marmot` — instead of that we re-implemented the set of word-level features in C++. The reason is that `Marmot` is written in Python, which is slower than C++ and other compiler-based languages, and calling Python programmes from C++ code also increases the running time. Since the decoder needs to evaluate thousands of hundreds of hypotheses, it would need to perform calls to Python for every hypothesis, which would make decoding prohibitively slow. Yet, even the C++ implementation of word-level QE feature slows down the decoder: while the generation of one sentence by our baseline MT system takes on average 0.9 seconds, the system which uses `WordQEFeature` needs 17.7 seconds. The latter figure also depends on the size of a CRF model used for the prediction. We believe that further code optimisation could reduce this time, but this was out of the scope of this thesis.

7.4.3 English–German experiments

The experiments were performed for English–German SMT system for the IT domain. For the baseline MT system we use the parallel data and settings described in 7.2.1. The only difference is that we added part-of-speech tags as an additional translation factor: every word in a translation table is represented as a pair "word|POS". This is done in order make POS-tags available at decoding time, because they are used as features. The POS-tagging of the source and the target is performed with `TreeTagger`. For the QE model training we use the English–German QE dataset described in section 6.2.2.

Decoding experiment

Our main experiment consists of end-to-end MT model training with a new feature. The only change to the training process is addition of a word-level QE model described above (see section 7.4.1), all other components of the MT model are trained as in the baseline. After having trained translation and reordering models, LM and QE model, we run tuning with the extended feature set: we add the `WordQEFeature` that computes word-level QE predictions.

Table 7.21 shows the result of adding the `WordQEFeature` to the baseline MT system. As we can see, the two variants of word-level features — **probability** and **score** — despite being related, give very different results. While the **probability** feature brings an improvement (the improvement is significant, although small), the application of the **score** feature results in a sharp decrease in quality.

We found that the **score** feature tends to have more diverse values than the **probability** feature. Figure 7.18 shows the distributions of both variants. Note that this distribution

	BLEU	METEOR	Avg.sentence length	Feature weight
Baseline	26.3	45.2	18.64	—
Probability feature	**26.7**	**45.5**	18.48	0.32
Score feature	23.0	41.7	18.64	0.09

Table 7.21 Performance of the SMT model for the English–German language pair with additional `WordQEFeature` (**probability** and **score** variants) on the test set.

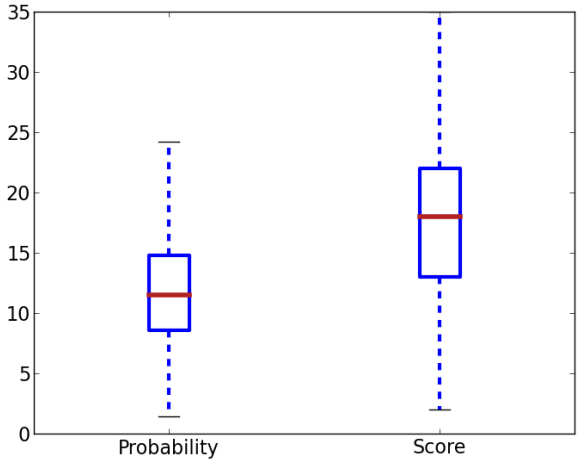

Fig. 7.18 Distribution of sentence-level values for the two varieties of `WordQEFeature`: **probability** and **score**.

for the sentence-level values, which are sums of word-level values, so they are outside the $[0, 1]$ range. We have seen analogous behaviour for **probability** and **score** in our N-best list reranking experiments (see section 7.2.2).

However, there the higher diversity of scores led to higher translation quality, whereas here we see the opposite. Therefore, when predicted at decoding time and at the level of words, the probability of a word being correct helps finding better translations. The binary score seems too strict and does not help discriminating between similar translation variants. Besides that, since word-level QE in general and particularly the QE model we are using can make wrong predictions, the errors in predicting binary score become too aggressive and disorientate the decoder. Taking this erroneous output into account results in deterioration of final translation, so the decoder learns to give it low priority. As we see in table 7.21, the weight of the probability feature trained by our MT system is higher than that of the score feature, which indicates its higher reliability and usefulness.

We claim that the use of a new feature at decoding time is potentially more useful than reranking of an N-best list with this new feature. The reason for this assumption is that

although the new feature might be relevant, a decoder which does not take it into account can eliminate hypotheses which are good according to this feature, and the feature will not be able to reward these hypotheses if used only after the translation is generated. The enhancement of the decoder with this feature is thus beneficial. However, even if good hypotheses are not pruned out, the feature is not guaranteed to bring them up to the first position in the list of possible translations. Therefore, we attempted to find even better translations according to the QE feature by reranking the N-best lists generated by our MT models with additional QE features using the `nbest-rescore` tool. However, this did not improve the results: the BLEU and METEOR of rescored test sets are not significantly different from their initial versions.

N-best list reranking experiment

In addition to conventional MT system tuning with a new feature we tried a simplified scenario for fast an easy insertion of the additional QE feature into an MT system after it has been trained. We tune an MT system without new features and then generate translation of the test set using the `WordQEFeature` with some baseline weight (in our experiments it was set to 0.1) and the tuned weights of the other features. Afterwards we retrain weights of all features with the `nbest-rescore` tool.

The table 7.22 shows the results of this experiment. For each of the three tested MT models: (i) baseline, (ii) model with the probability feature and (iii) model with the score feature — we provide two variants: the original one and the one with retrained feature weights. Retraining of weights slightly (yet significantly) improved the results for all models except the value of METEOR for the model that used the score feature.

	BLEU	METEOR	Avg.sentence length
Baseline (no reranking)	26.3	45.2	18.64
Baseline (reranking without new features)	**26.5**	45.6	18.62
QE features: no reranking			
Probability feature	25.1	45.8	19.76
Score feature	24.2	45.9	20.43
QE features: reranking			
Probability feature	25.3	**46.2**	19.77
Score feature	26.1	45.5	18.8

Table 7.22 N-best list reranking for the English–German MT system with word-level QE decoder features.

None of the models with additional feature could beat any of the baselines in terms of BLEU. However, all of them were better than both baselines in terms of METEOR. As it

was discussed in section 7.2.2, such a discrepancy can be explained by the fact that MT decoder generates longer sentences when using QE features. Here, similarly to our N-best list reranking experiments we can see that all QE features have a length bias which results in lower BLEU scores and high METEOR scores. This bias is almost eliminated by retraining of weights in the model which used score feature, and we see how its scores become very close to the baseline.

Analogously to the tuning experiment, here we see that the score feature performs worse than the probability feature, although the difference in scores is not so large and holds only for the BLEU score.

7.4.4 German–English experiments

In order to provide a more objective assessment of our QE decoder feature we repeat the same experiments with the German–English dataset.

Decoding experiments

Our experiments with the decoder feature for the German–English data agree with the N-best list reranking experiments for this dataset in the fact that we are unable to improve the baseline with QE predictions. This time adding both **probability** and **score** feature resulted in a slight decrease of translation quality. Table 7.23 shows the result.

	BLEU	METEOR	Avg.sentence length	Feature weight
Baseline	**62.9**	**46.3**	26.2	—
Probability feature	62.3	46.0	26.2	0.048
Score feature	62.3	46.0	26.2	0.002

Table 7.23 Performance of the SMT model for the English–German language pair with additional `WordQEFeature` (**probability** and **score** variants) on the test set.

Unlike the the English–German experiments, the two types of word-level QE feature — probability and score — return the same result. We found out that the difference between the translations produced by these two types of the feature is not big — less than 10% of the sentences differ. In fact, the difference between the output of the baseline model and the models that use QE features is also quite small. Table 7.24 shows the percentage of sentences which are different in the translations of the test set. It is evident that the new features only have minor impact on the output.

This observation is corroborated by the small difference in scores between the baseline and extended models, and the absence of length bias in models with QE features which we

observed before. This weak effect of the QE features is explained by the low weights of these features (see table 7.23) which were defined via tuning. Here, analogously to our N-best list experiments with the German–English data, the QE features are assigned very low weights, which indicates their low reliability and ineffectiveness for the MT system.

	baseline	probability	score
baseline		13.11%	9.39%
probability			8.31%
score			

Table 7.24 Percentage of sentences that differ in translations of the test set generated by different models.

N-best reranking experiments

In another attempt to make our German–English MT system take into account the QE features we repeat another series of N-best reranking experiments with decoder features. Here, we decode the test set with pre-tuned baseline feature weights and the weight of `WordQEFeature` set to 0.1. Then we retrain the feature weights again this time taking into account the `WordQEFeature` weight. Table 7.25 contains the results of the baseline MT system, MT systems which used probability and score features, and their rescored versions.

	BLEU	METEOR	Avg.length
Baseline (no reranking)	**62.8**	**46.3**	26.2
Baseline (reranking without new features)	62.8	46.3	26.2
Probability feature	62.1	46.4*	26.8
Score feature	61.0	46.3*	26.9
Probability feature	62.8*	46.2	26.2
Score feature	62.6	46.2	26.3

Table 7.25 N-best list reranking with word-level QE features (experiments for the German–English pharmaceutical corpus). Asterisk (*) denotes results which are not significantly different from the baseline.

Here, analogously to the experiments with the English–German data, the incorporation of the QE feature failed to improve the performance: BLEU as well as METEOR go down for the MT systems which use `WordQEFeature`, although in the previous experiments the METEOR slightly improved. We also see the length bias that was characteristic of QE features in the decoding and N-best list reranking experiments, and the **probability** feature again outperforms the **score** feature. However, all the trends are much less pronounced

than in the experiments with the IT data. The length bias is not as strong and is completely eliminated after the retraining of weights. The difference between the performance of the **probability** and **score** features is also smaller.

7.5 Conclusions

In this chapter we presented ways of using QE as an additional feature in the MT system. We injected the QE feature at two different stages. First, we used it for selection of the best translation of a sentence from a list of translation variants. We showed that QE predictions can be useful for selecting the translation in combination with the baseline features of an MT model. Secondly, we used a word-level QE model to predict the quality of sentences during translation, so that better translations could be promoted and the worse ones penalised before the full translation is generated — this can allow decoder to keep more hypotheses which are considered good by the QE model. This method also yields an improvement of translation quality — the information on the probability of each word being correct helps selecting better translations. On the other hand, binary word-level quality predictions are too coarse-grained for this task.

However, the outcome all the described methods of incorporation of QE predictions highly depends on the performance of the baseline MT system. While in the experiments with MT systems trained on English–German data of IT domain the use of QE feature lead to moderate improvements in translation quality, the MT models trained on German–English pharmaceutical data did not benefit from it. We assume that the culprit of this result is the high quality of the baseline MT system. Therefore, our method does not seem suited for improving well-performing MT models, but can be applied to models with lower quality, trained on less repetitive data.

The conducted experiments were not exhaustive, thus there exist some other possibilities of future work. First of all, we could repeat the experiments with the use of better-performing QE models. For the decoding experiments this might require substantial changes in the MT model architecture and incorporation of new information that can improve the quality of predictions on unfinished sentences. In the next chapter we talk in more detail about further improvement of our methods and techniques and summarise the findings of the thesis.

Chapter 8

Conclusions and future work

The initial goal of this work was to find an effective way of improving the performance of Machine Translation systems using human feedback on automatically translated texts. We conducted the review of existing types of human feedback and techniques of incorporation of this feedback into MT systems as well as methods to improve translation quality via standalone systems that are trained on human feedback.

We found that manual post-edits of automatically translated texts are the best trade-off between the degree of informativeness and amount of effort to collect, so the majority of existing research in the field incorporates human feedback in the form of post-edits. The feedback is sometimes used in standalone applications that operate on already generated translations — one example of such applications are Automatic Post-Editing tools, which have lately achieved good results. However, the feedback is more often incorporated directly to an MT system where it is used to update its models in the online mode. This technique, although effective, has its limitations, which stem mainly from the insufficient amount of feedback available.

Therefore, in order to improve upon the existing methods, a considerably larger amount of human-labelled data than those that are available is needed. Since data labelling is a time-consuming task, its manual acquisition is not feasible. So we formulated two goals for our work:

- Generate high-quality pseudo-human feedback using Quality Estimation (QE) techniques.
- Devise methods to incorporate quality scores computed by QE models into SMT systems.

In the first goal, we simulate human feedback with QE models: they are trained with small amount of human feedback, and then can label translations for quality automatically,

without reference translations or human experts. This gives us a virtually unlimited amount of pseudo-human feedback, which is cheap and fast to obtain. However, in order to be able to use the predictions by QE models for the improvement of MT systems, we needed to make sure that these predictions were accurate enough. Therefore, our first goal revolve mainly around the performance of QE models. In what follows we summarise our findings and possible future work directions for each of the two goals.

8.1 Improvement of quality estimation models

Our work on the improvement of QE models consisted of two parts: generation of new artificial data for training of QE models at different levels of granularity and estimation of quality for a new level: multi-word phrases.

8.1.1 Artificial data

Our data manipulation techniques proved effective in certain settings. Word-level QE models can benefit from the elimination of error-free translations from the training data, provided that the overall translation quality for the training data is high. On the other hand, our artificially generated data failed to improve the performance of word-level QE models, although it was effective for the sentence-level ones.

The artificial data generation approach has its limitations. While it was shown to improve the scores of sentence-level QE models when the initial training data was small (up to 1,000 sentences), our further experiments showed that when the amount of training data is larger (6,000 sentences or more), the artificial sentences do not bring any improvements. This could have happened because of the fact that the artificial data is less useful when the amount of real data reaches some threshold. However, that could also indicate flaws in the artificial data generation techniques. As we have already pointed out, our current techniques operate at the level of individual words: errors are introduced by changing single words, although words in a sentence are interconnected, and a change in one word can influence its neighbours. A successful artificial data generation technique should be able to simulate these influences.

The work on artificial data could benefit from conceptually different approaches. While we considered only the types of errors that are based on the edit operations performed on them (substitution, deletion, insertion), there currently exist more elaborate error typologies which discriminate between errors of different types, origin and gravity. While these different types of errors are often difficult to detect in real automatic translations, they should be easier to simulate and induce into correct sentences. As MT engines and CAT-tools become more

popular, the need in QE models for new language pairs will grow. Since every new QE model needs human-labelled data which usually cannot be obtained in large quantities, the problem of data scarcity will still hold for the majority of language pairs. Therefore, future work on the improvement of QE should include research on the artificially generated data.

8.1.2 Phrase-level QE

Phrase-level QE is a new view on translation quality: for the first time we suggested evaluating the quality of phrases which form a sentence. The emergence of this new level of granularity evoked many questions: we needed to define a phrase itself, ways of acquiring its quality judgements, the most suitable representation of these judgements, the features to represent phrases and ways of evaluating phrase-level QE models.

In the thesis we suggested initial solutions for all these questions. We limited the definition of phrase to phrases as produced by decoder of a statistical MT system. We suggested to generate phrase-level labels from post-edits analogously to word-level ones. We experimented with two different feature sets: features that consider a phrase as a sequence, and embeddings for the source and the target phrases. The former set yielded better results. We extended it with context features, which further improved the performance. Finally, we trained our phrase-level models with Machine Learning algorithms which proved effective on word-level data and showed that CRF sequence labelling is a reasonable choice for phrase-level QE. However, each of those aspects could be further improved. It is infeasible to extensively research all these topics in one PhD thesis, so many directions were left for future work on phrase-level QE.

The notion of phrase itself needs more careful definition. Although the MT-defined phrases make sense for phrase-level SMT models, they might not be a good solution when estimating quality of MT models of a different architecture. For example, in this work we left out the linguistic aspect of phrases, although the representation of machine-translated sentences as a sequence of syntactically motivated phrases could be beneficial when estimating the quality of translations generated by rule-based or hybrid MT systems. Alternatively, the architecture of a phrase-level QE model could be reshaped completely: while we provided phrase segmentation as the input data, the segmentation of sentences into phrases and the labelling of these phrases could be predicted jointly.

Finally, we did not find an optimal way of evaluating phrase-level QE models. We compared a number of metrics for the evaluation of word-level QE models and found that the multiplication of F_1-scores for the "OK" and "BAD" classes was the least biased and the most effective in discriminating between QE models. However, the word-level metrics are not usable for the evaluation of phrase-level models, as the results of the shared task on

phrase-level QE showed. We suggest that a more adequate way of evaluating phrase-level QE is a metric which considers a phrase as a minimal unit, but this aspect also needs verification.

Overall, the phrase-level QE at its present state was not able to beat word-level models, however, we believe that careful feature engineering and probably combinations with other levels of granularity can lead to significantly better performance.

8.2 Improvement of machine translation

Another contribution of this thesis is the work on incorporating of quality predictions done by QE models into MT models. Our use of QE was twofold: we used quality predictions to select new data for MT system training, and introduced them into an MT system as a new feature.

We suggested an Active Learning technique for MT which is based on predicted data quality. We used the predictions made by QE models as a measure of model uncertainty: the worse the translation made by an MT system, the less certain it is about a particular text, and then the more new information (i.e. unseen words and ngrams) this text potentially contains. Translation quality did not however prove to be a good data selection criterion: our AL method failed to increase the rate of translation improvement compared to random data selection. However, our experiments were not exhaustive, and some interesting findings emerged. As it was shown, the QE models which were adapted to a particular MT model, were better informed about its needs. The use of better QE model adaptation techniques could result in more effective data selection.

Another direction of research dealt with tighter integration of QE and MT systems. We used QE predictions as additional features to improve the selection of the best translation variant from lists of N best translations produced by an MT model. We combined QE predictions with other features and re-ranked the sentences based the new combined scores. We performed experiments with a single weights update as well as multiple update rounds. The techniques proved useful to some extent, but could not lead to major improvements. This can be partially attributed to the fact that QE models that we used were not accurate enough. This to some extent is a characteristic of the state-of-the-art QE models in general: the task of QE has not reached a level of performance that makes it effective in this downstream task. That said, we could not use the best-performing possible QE models for our experiments: they were impossible to replicate because of lack of data or other resources. Therefore, one of the directions for future work on all the techniques of MT improvement is to use better QE models.

Finally, we tried incorporating QE directly into the translation generation process. This required substantial changes in the QE models we used because initially they were unsuitable for estimating quality of unfinished sentences. We had to drop features that use non-local contexts (e.g. syntactic features and other features that fetch information from the whole sentence) which further reduced the quality of the QE models. Despite these changes, the incorporation of QE into MT decoding was able to improve translation quality. We suggest that the future research should concentrate on improving the performance of QE models that operate at translation time. That could be achieved via extensions of the feature set: although the right context of a word is unavailable when estimating quality during decoding, some of the features that we dropped could be retrieved from partial translations: there are algorithms for producing POS-tagging and parsing for unfinished sentences and QE could benefit from them. Besides that, we suggest that vector representations of words and possibly phrases could be a good supplement to the set of features, as they allow to use large volumes of data which cannot be incorporated into QE otherwise. Our work includes some experiments on the use of vector representations in QE, which proved unsuccessful. Therefore, future research should be dedicated to finding better ways of integration of vector representations into QE and making them more suitable for the purpose.

The incorporation of QE into decoding opens a range of possibilities. It can be used not only as a feature, but also as a training objective. In other words, instead of optimising the translations to be close to the references in terms of an automatic evaluation metric Alternatively, the prediction by a QE model could be used as a measure of the translation quality instead of or together with the currently used model score which is a combination of feature values. We have made an attempt to replace model score with QE predictions in our N-best list reordering experiments (*naive reordering* experiments).

All our experiments on the incorporation of QE into MT considered only one MT system architecture, namely, phrase-based statistical MT. However, neural Machine Translation (NMT) has become a popular and promising direction of research, and in the latest competition of MT systems (Bojar et al., 2016) NMT models won in the majority of subtasks, which means that NMT has become a new state of the art in automatic translation. However, although we did not tested our methods and QE models in the task of improving NMT, we suppose that they could be useful for that. The NMT systems and QE models we used are based on different principles: while NMT extracts implicit regularities and correspondences from the data, our QE models use features from phrase-based SMT, for example, explicit alignments between words in the source and target. These alignments are often not presented in NMT, although the notion of correspondences between particular words or phrases has

also been transferred to NMT in the shape of attention mechanism (Bahdanau et al., 2014). Therefore, we suppose that NMT and QE models are complimentary.

The particular techniques of incorporation of QE scores into SMT systems that we described can also be used in NMT. First of all, the reordering of N-best list can still be used for the improvement of NMT systems output, because these systems have the same notion of N-best list (i.e. they can generate a list of top translation suggestions analogously to SMT systems). If the incorporation of QE predictions into decoding is concerned, it need substantial changes to be adapted to NMT. Nevertheless, analogously to SMT, decoding in NMT is also sequential and goes from left to right. Therefore, word-level QE predictions can be incorporated into it: at each step a QE model can evaluate the latest generated word, and the prediction can be fed back to the decoder.

Overall, the main contribution of this thesis was to propose a number of ways in which (pseudo-) human feedback can be helpful to improve MT performance. While the results were not positive in all cases, we found specific settings where the methods are more likely to work and suggested further work that could improve on our current results.

References

Abdi, H. (2007). The bonferroni and šidák corrections for multiple comparisons. *Encyclopedia of measurement and statistics*, 3:103–107.

Alabau, V., Leiva, L. A., Ortiz-Martínez, D., and Casacuberta, F. (2012). User Evaluation of Interactive Machine Translation Systems. In *EAMT-2012: 16th Annual Conference of the European Association for Machine Translation*, pages 28–30, Trento, Italy.

Alabau, V., Ortiz-Martínez, D., Sanchis, A., and Casacuberta, F. (2010). Multimodal interactive machine translation. In *ICMI-MLMI-2010: International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction*, pages 1–4, Beijing, China. ACM Press.

Albrecht, J. and Hwa, R. (2007). Regression for Sentence-Level MT Evaluation with Pseudo References. In *ACL-2007: 45th Annual Meeting of the Association for Computational Linguistics*, pages 296–303.

Ambati, V., Vogel, S., and Carbonell, J. (2010). Active Learning and Crowd-Sourcing for Machine Translation. In *LREC-2010: 7th International Conference on Language Resources and Evaluation*, pages 2169–2174, Valetta, Malta.

Ananthakrishnan, S., Prasad, R., Stallard, D., and Natarajan, P. (2010). Discriminative Sample Selection for Statistical Machine Translation. In *EMNLP-2010: Conference on Empirical Methods in Natural Language Processing*, pages 626–635, Massachusetts, USA.

Axelrod, A., He, X., and Gao, J. (2011). Domain Adaptation via Pseudo In-Domain Data Selection. In *EMNLP-2011: Conference on Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, UK.

Aziz, W., Castilho Monteiro de Sousa, S., and Specia, L. (2012). Pet: a tool for post-editing and assessing machine translation. In *LREC-2012: 8th international conference on Language Resources and Evaluation*, pages 3982–3987, Istanbul, Turkey.

Bach, N. (2011). Goodness: A Method for Measuring Machine Translation Confidence. In *ACL-2011: 49th Annual Meeting of the Association for Computational Linguistics*, pages 211–219, Portland, Oregon.

Bahdanau, D., Cho, K., and Bengio, B. (2014). Neural machine translation by jointly learning to align and translate.

Banerjee, P., Rubino, R., Roturier, J., and van Genabith, J. (2013). Quality Estimation-guided Data Selection for Domain Adaptation of SMT. In *MT Summit XIV: 14th Machine Translation Summit*, pages 101–108, Nice, France.

Banerjee, S. and Lavie, A. (2005). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *ACL-2005: 43rd Annual meeting of the Association for Computational Linguistics, Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and\or Summarization*, pages 65–72, Ann Arbor, USA.

Bechara, H., Ma, Y., and Genabith, J. v. (2011). Statistical post-editing for a statistical mt system. In *MTS-2011: Proceedings of the 13th Machine Translation Summit*, pages 308–315, Xiamen, China.

Beck, D., Shah, K., Cohn, T., and Specia, L. (2013). SHEF-Lite: When less is more for translation quality estimation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 337–342, Sofia, Bulgaria. Association for Computational Linguistics.

Bertoldi, N., Cettolo, M., and Federico, M. (2013). Cache-based Online Adaptation for Machine Translation Enhanced Computer Assisted Translation. In *MT Summit XIV: 14th Machine Translation Summit*, pages 35–42, Nice, France.

Bhanuprasad, K. and Svenson, M. (2008). Errgrams – A Way to Improving ASR for Highly Inflected Dravidian Languages. In *IJCNLP-2008: the Third International Joint Conference on Natural Language Processing*, pages 805–810, Hyderabad, India.

Biçici, E. (2013). Referential translation machines for quality estimation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 392–397, Sofia, Bulgaria. Association for Computational Linguistics.

Bicici, E. (2016). Referential translation machines for predicting translation performance. In *Proceedings of the First Conference on Machine Translation*, Berlin, Germany. Association for Computational Linguistics.

Bisazza, A., Ruiz, N., and Federico, M. (2011). Fill-up versus Interpolation Methods for Phrase-based SMT Adaptation. In *IWSLT-2011: 8th International Workshop on Spoken Language Translation*, pages 136–143, San Francisco, USA.

Blain, F., Logacheva, V., and Specia, L. (2016). Phrase level segmentation and labelling of machine translation errors. In *Tenth International Conference on Language Resources and Evaluation*, LREC, pages 2240–2245, Portoroz, Slovenia.

Blain, F., Schwenk, H., and Senellart, J. (2012). Incremental Adaptation Using Translation Information and Post-Editing Analysis. In *IWSLT-2012: 9th International Workshop on Spoken Language Translation*, pages 229–236, Hong Kong, China.

Blain, F., Senellart, J., Schwenk, H., Plitt, M., and Roturier, J. (2011). Qualitative Analysis of Post-Editing for High Quality Machine Translation. In *MT Summit XIII: 13th Machine Translation Summit*, pages 164–171, Xiamen, China.

Blatz, J., Fitzgerald, E., Foster, G., Gandrabur, S., Goutte, C., Kulesza, A., Sanchis, A., and Ueffing, N. (2004). Confidence Estimation for Machine Translation: workshop report. Technical report, Johns Hopkins University.

Bloodgood, M. and Callison-Burch, C. (2010). Bucking the Trend : Large-Scale Cost-Focused Active Learning for Statistical Machine Translation. In *ACL-2010: 48th Annual Meeting of the Association for Computational Linguistics*, pages 854–864, Uppsala, Sweden.

Bojar, O. (2011). Analyzing Error Types in English-Czech Machine Translation. *Prague Bulletin of Mathematical Linguistics*, 95:63–76.

Bojar, O., Buck, C., Callison-Burch, C., Federmann, C., Haddow, B., Koehn, P., Monz, C., Post, M., Soricut, R., and Specia, L. (2013). Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria. Association for Computational Linguistics.

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., Soricut, R., Specia, L., and Tamchyna, A. (2014). Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Jimeno Yepes, A., Koehn, P., Logacheva, V., Monz, C., Negri, M., Neveol, A., Neves, M., Popel, M., Post, M., Rubino, R., Scarton, C., Specia, L., Turchi, M., Verspoor, K., and Zampieri, M. (2016). Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Bojar, O., Chatterjee, R., Federmann, C., Haddow, B., Hokamp, C., Huck, M., Logacheva, V., Koehn, P., Monz, C., Negri, M., Pecina, P., Post, M., Scarton, C., Specia, L., and Turchi, M. (2015). Findings of the 2015 Workshop on Statistical Machine Translation. In *WMT-2015*, WMT, pages 12–58, Lisbon, Portugal.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Brockett, C., Dolan, W. B., and Gamon, M. (2006). Correcting esl errors using phrasal smt techniques. In *Coling-ACL-2006: 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, Sydney, Australia*, Sydney, Australia. Association for Computational Linguistics.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311.

Callison-Burch, C., Koehn, P., Monz, C., Post, M., Soricut, R., and Specia, L. (2012). Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada. Association for Computational Linguistics.

Camargo de Souza, J. G., Buck, C., Turchi, M., and Negri, M. (2013). FBK-UEdin participation to the WMT13 quality estimation shared task. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 352–358, Sofia, Bulgaria. Association for Computational Linguistics.

Camargo de Souza, J. G., González-Rubio, J., Buck, C., Turchi, M., and Negri, M. (2014). Fbk-upv-uedin participation in the wmt14 quality estimation shared-task. In *WMT-2014*, pages 322–328, Baltimore, Maryland, USA.

Cettolo, M., Bertoldi, N., Federico, M., Schwenk, H., Barrault, L., and Servan, C. (2014). Translation project adaptation for mt-enhanced computer assisted translation. *Machine Translation*, 28:127–150.

Cettolo, M., Federico, M., and Bertoldi, N. (2010). Mining Parallel Fragments from Comparable Texts. In *IWSLT-2010: 7th International Workshop on Spoken Language Translation*, pages 227–234, Paris, France.

Chang, M.-w., Srikumar, V., Goldwasser, D., and Roth, D. (2010). Structured Output Learning with Indirect Supervision. In *ICML-2010: 27th International Conference on Machine Learning*, Haifa, Israel.

Chatterjee, R., C. de Souza, J. G., Negri, M., and Turchi, M. (2016). The fbk participation in the wmt 2016 automatic post-editing shared task. In *Proceedings of the First Conference on Machine Translation*, pages 745–750, Berlin, Germany. Association for Computational Linguistics.

Cherry, C. and Foster, G. (2012). Batch Tuning Strategies for Statistical Machine Translation. In *HLT-NAACL-2012: Human Language Technologies: Conference of the North American Chapter of the Association for Computational Linguistics*, pages 427–436, Montréal, Canada.

Clark, J. H., Dyer, C., Lavie, A., and Smith, N. A. (2011). Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 176–181, Stroudsburg, PA, USA. Association for Computational Linguistics.

Collins, M. (2002). Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *EMNLP-2002: Conference on Empirical Methods in Natural Language Processing*, pages 1–8, Philadelphia, USA.

Collins, M., Roark, B., and Saraclar, M. (2005). Discriminative Syntactic Language Modeling for Speech Recognition. In *ACL-2005: 43rd Annual meeting of the Association for Computational Linguistics*, Ann Arbor, USA.

Crammer, K. and Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

Culotta, A. and McCallum, A. (2004). Confidence estimation for information extraction. In *Proceedings of HLT-NAACL 2004: Short Papers*, HLT-NAACL-Short '04, pages 109–112, Boston, Massachusetts. Association for Computational Linguistics.

de Souza, J. G. C., Negri, M., Ricci, E., and Turchi, M. (2015). Online multitask learning for machine translation quality estimation. In *ACL-IJCNLP-2015: the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 219–228, Beijing, China.

Déchelotte, D. (2010). Analysis of translation suggestions on Reverso translation engines: initial findings.

Denkowski, M., Dyer, C., and Lavie, A. (2014a). Learning from post-editing: Online model adaptation for statistical machine translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 395–404, Gothenburg, Sweden.

Denkowski, M. and Lavie, A. (2010). Meteor-next and the meteor paraphrase tables: Improved evaluation support for five target languages. In *WMT-2010: Joint 5th Workshop on Statistical Machine Translation and MetricsMATR*, WMT '10, pages 339–342, Uppsala, Sweden. Association for Computational Linguistics.

Denkowski, M. and Lavie, A. (2012). Challenges in Predicting Machine Translation Utility for Human. In *AMTA-2012: 10th Conference of the Association for Machine Translation in the Americas*, San Diego, CA.

Denkowski, M., Lavie, A., Lacruz, I., and Dyer, C. (2014b). Real time adaptive machine translation for post-editing with cdec and transcenter. In *Proceedings of the EACL 2014 Workshop on Humans and Computer-assisted Translation*, pages 72–77, Gothenburg, Sweden.

Du, J., Srivastava, A., Way, A., Maldonado-guerra, A., and Lewis, D. (2015). An Empirical Study of Segment Prioritization for Incrementally Retrained Post-Editing-Based SMT. In *MT Summit XV: 15th Machine Translation Summit*, pages 172–185, Miami, USA.

Dugast, L., Senellart, J., and Koehn, P. (2007). Statistical Post-Editing on SYSTRAN ' s Rule-Based Translation System. In *WMT-2007: 2nd Workshop on Statistical Machine Translation*, pages 220–223, Prague, Czech Republic.

Durrani, N., Fraser, A., Schmid, H., Hoang, H., and Koehn, P. (2013). Can Markov Models Over Minimal Translation Units Help Phrase-Based SMT ? In *ACL-2013: 51st Annual Meeting of the Association for Computational Linguistics, Short papers*, pages 399–405, Sofia, Bulgaria.

Dyer, C., Chahuneau, V., and Smith, N. (2013). A simple, fast, and effective parameterization of IBM model 2. In *NAACL-HLT-2013: Human Language Technologies: Conference of the North American Chapter of the Association for Computational Linguistics*, pages 644–648, Atlanta, Georgia, USA.

Dyer, C., Lopez, A., Ganitkevitch, J., Weese, J., Ture, F., Blunsom, P., Setiawan, H., Eidelman, V., and Resnik, P. (2010). cdec: A Decoder, Alignment, and Learning Framework for Finite-State and Context-Free Translation Models. In *ACL-2010: 48th Annual Meeting of the Association for Computational Linguistics, System Demonstrations*, pages 7–12, Uppsala, Sweden.

Eck, M., Vogel, S., and Waibel, A. (2005). Low Cost Portability for Statistical Machine Translation based on N-gram Frequency and TF-IDF. In *IWSLT-2005: International Workshop on Spoken Language Translation: Evaluation Campaign on Spoken Language Translation*, Pittsburgh, PA.

Esplà-Gomis, M., Sánchez-Martínez, F., and Forcada, M. (2015). Ualacant word-level machine translation quality estimation system at wmt 2015. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 309–315, Lisbon, Portugal. Association for Computational Linguistics.

Esplà-Gomis, M., Sánchez-Martínez, F., and Forcada, M. (2016). Ualacant word-level and phrase-level machine translation quality estimation systems at wmt 2016. In *Proceedings of the First Conference on Machine Translation*, Berlin, Germany. Association for Computational Linguistics.

Esplà-Gomis, M., Sánchez-Martínez, F., and Forcada, M. L. (2012). A Simple Approach to Use Bilingual Information Sources for Word Alignment. *Procesamiento del Lenguaje Natural*, 49:93–100.

Esteban, J., Lorenzo, J., Valderrábanos, A. S., and Lapalme, G. (2004). TransType2 – An Innovative Computer-Assisted Translation System. In *ACL-2004: 42nd Annual Meeting of the Association for Computational Linguistics*, pages 1–4, Barcelona, Spain.

Farajian, M. A., Bertoldi, N., and Federico, M. (2014). Online Word Alignment for Online Adaptive Machine Translation. In *EACL-2014: 14th Conference of the European Chapter of the Association for Computational Linguistics, Workshop on Humans and Computer-assisted Translation*, pages 84–92, Gothenburg, Sweden.

Federmann, C. (2012). Appraise: An open-source toolkit for manual evaluation of machine translation output. *The Prague Bulletin of Mathematical Linguistics*, 98:25–35.

Felice, M. and Yuan, Z. (2014). Generating artificial errors for grammatical error correction. In *EACL-2014: 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 116–126.

Fishel, M., Bojar, O., and Popović, M. (2012). Terra: a Collection of Translation Error-Annotated Corpora. In *LREC-2012: 8th international conference on Language Resources and Evaluation*, pages 7–14, Istanbul, Turkey.

Formiga, L., Gonzàlez, M., Barrón-Cedeño, A., Fonollosa, J. A. R., and Màrquez, L. (2013). The TALP-UPC Approach to System Selection: Asiya Features and Pairwise Classification Using Random Forests. In *WMT-2013: 8th Workshop on Statistical Machine Translation*, pages 359–364.

Foster, G., Isabelle, P., and Plamondon, P. (1997). Target-Text Mediated interactive machine translation.pdf. *Machine Translation*, 12(1/2):175 – 194.

Foster, G. and Kuhn, R. (2007). Mixture-Model Adaptation for SMT. In *WMT-2007: 2nd Workshop on Statistical Machine Translation*, pages 128–135, Prague, Czech Republic.

Gamon, M., Aue, A., and Smets, M. (2005a). Sentence-level mt evaluation without reference translations: Beyond language modeling. In *EAMT-2005: 10th Annual Conference of the European Association for Machine Translation "Practical applications of machine translation"*, Budapest, Hungary.

Gamon, M., Aue, A., and Smets, M. (2005b). Sentence-Level MT evaluation without reference translations: beyond language modeling. In *EAMT-2005: 10th Conference of the European Association for Machine Translation*, Budapest, Hungary.

Gandrabur, S. and Foster, G. (2003). Confidence estimation for translation prediction. In *HLT-NAACL-2003: proceedings of Seventh Conference on Natural Language Learning*, pages 95–102, Edmonton, Canada. Association for Computational Linguistics.

González-Rubio, J., Ortiz-Martínez, D., and Casacuberta, F. (2010). Balancing User Effort and Translation Error in Interactive Machine translation via confidence measures. In *ACL-2010: 48th Annual Meeting of the Association for Computational Linguistics*, pages 173–177, Uppsala, Sweden.

González-Rubio, J., Ortiz-Martínez, D., and Casacuberta, F. (2012). Active learning for interactive machine translation. In *EACL-2012: 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 245–254, Avignon, France.

GPy (since 2012). GPy: A gaussian process framework in python. http://github.com/SheffieldML/GPy.

Graham, Y. (2015). Improving Evaluation of Machine Translation Quality Estimation. In *ACL-2015: 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1804–1813, Beijing, China.

Green, S., Cer, D., and Manning, C. D. (2014a). Phrasal: A toolkit for new directions in statistical machine translation. In *In Proceddings of the Ninth Workshop on Statistical Machine Translation*.

Green, S., Wang, S., Chuang, J., Heer, J., Schuster, S., and Manning, C. D. (2014b). Human Effort and Machine Learnability in Computer Aided Translation. In *EMNLP-2014: Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar.

Groves, D. and Schmidtke, D. (2009). Identification and Analysis of Post-Editing Patterns for MT. In *MT Summit XII: 12th Machine Translation Summit*, Ottawa, Ontario, Canada.

Guzmán, R. (2007). Manual MT Post-editing: "If It's not Broken, Don't Fix It!". *Translation Journal*, 11(4).

Haffari, G., Roy, M., and Sarkar, A. (2009). Active learning for statistical phrase-based machine translation. In *NAACL-HLT-2009: Human Language Technologies: the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Boulder, Colorado, USA. Association for Computational Linguistics.

Hardmeier, C. (2011). Improving Machine Translation Quality Prediction with Syntactic Tree Kernels. In *EAMT-2011: 15th Conference of the European Association for Machine Translation*, pages 233–240.

Hardmeier, C., Nivre, J., and Tiedemann, J. (2012). Tree Kernels for Machine Translation Quality Estimation. In *WMT-2012: 7th Workshop on Statistical Machine Translation*, pages 109–113.

Hardt, D. and Elming, J. (2010). Incremental Re-training for Post-editing SMT. In *AMTA-2010: 9th Conference of the Association for Machine Translation in the Americas*, Denver, Colorado.

Hasler, E., Haddow, B., and Koehn, P. (2011). Margin infused relaxed algorithm for moses. *The Prague Bulletin of Mathematical Linguistics*, 96:69–78.

Hopkins, M. and May, J. (2011). Tuning as ranking. In *EMNLP-2011: Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1352–1362, Edinburgh, United Kingdom.

Hu, C., Resnik, P., Kronrod, Y., Eidelman, V., Buzek, O., and Bederson, B. B. (2011). The Value of Monolingual Crowdsourcing in a Real-World Translation Scenario: Simulation using Haitian Creole Emergency SMS Messages. In *WMT-2011: 6th Workshop on Statistical Machine Translation*, pages 399–404, Edinburgh, Scotland, UK.

Izumi, E., Uchimoto, K., Saiga, T., Supnithi, T., and Isahara, H. (2003). Automatic error detection in the japanese learners' english spoken data. In *ACL-2003: 41st Annual Meeting of the Association for Computational Linguistics*, pages 145–148, Sapporo, Japan.

Junczys-Dowmunt, M. and Grundkiewicz, R. (2016). Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing. In *Proceedings of the First Conference on Machine Translation*, pages 751–758, Berlin, Germany. Association for Computational Linguistics.

Kim, H. and Lee, J.-H. (2016). Recurrent neural network based translation quality estimation. In *Proceedings of the First Conference on Machine Translation*, pages 787–792, Berlin, Germany. Association for Computational Linguistics.

Knight, K. and Chander, I. (1994). Automated Postediting of Documents. In *AAAI-1994: 12th National conference of the American Association for Artificial Intelligence*, Seattle, Washington, USA.

Koehn, P. (2005). Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

Koehn, P. (2009). A Web-Based Interactive Computer Aided Translation Tool. In *ACL-IJCNLP-2009: 47th Annual Meeting of the Association for Computational Linguistics and the 4th IJCNLP, Software Demonstrations*, pages 17–20, Suntec, Singapore.

Koehn, P. (2010). Enabling Monolingual Translators: Post-Editing vs. Options. In *NAACL-HLT-2010: Human Language Technologies: the annual conference of the North American Chapter of the Association for Computational Linguistics*, pages 537–545, Los Angeles, California.

Koehn, P. and Haddow, B. (2009). Interactive Assistance to Human Translators using Statistical Machine Translation Methods. In *MT Summit XII: 12th Machine Translation Summit*, pages 73–80, Ottawa, Ontario, Canada.

Koehn, P., Hoang, H., Birch, A., Callison-burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open Source Toolkit for Statistical Machine Translation. In *ACL-2007: 45th Annual Meeting of the Association for Computational Linguistics, Demo and Poster sessions*, pages 177–180, Prague, Czech Republic.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 48–54, Edmonton, Canada.

Koehn, P. and Schroeder, J. (2007). Experiments in Domain Adaptation for Statistical Machine Translation. In *WMT-2007: 2nd Workshop on Statistical Machine Translation*, pages 224–227, Prague, Czech Republic.

Kozlova, A., Shmatova, M., and Frolov, A. (2016). Ysda participation in the wmt'16 quality estimation shared task. In *Proceedings of the First Conference on Machine Translation*, pages 793–799, Berlin, Germany. Association for Computational Linguistics.

Kreutzer, J., Schamoni, S., and Riezler, S. (2015). Quality estimation from scratch (quetch): Deep learning for word-level translation quality estimation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 316–322, Lisbon, Portugal. Association for Computational Linguistics.

Krings, H. P. (2001). *Repairing Texts*. Kent State University Press.

Kunchukuttan, A., Roy, S., Patel, P., Ladha, K., Gupta, S., Khapra, M., and Bhattacharyya, P. (2012). Experiences in Resource Generation for Machine Translation through Crowdsourcing. In *LREC-2012: 8th international conference on Language Resources and Evaluation*, pages 384–391, Istanbul, Turkey.

Lafferty, J., Mcallum, A., and Pereira, F. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML-2001*, pages 282–289.

Langlais, P., Foster, G., and Lapalme, G. (2000). TransType: a Computer-Aided Translation Typing System. In *NAACL-ANLP-EMTS-2000: NAACL-ANLP Workshop on Embedded machine translation systems*, volume 5, pages 46–51.

Langlois, D. (2015). Loria system for the wmt15 quality estimation shared task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 323–329, Lisbon, Portugal. Association for Computational Linguistics.

Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1188–1196.

Levenberg, A., Callison-burch, C., and Osborne, M. (2010). Stream-based Translation Models for Statistical Machine Translation. In *NAACL-HLT-2010: Human Language Technologies: the annual conference of the North American Chapter of the Association for Computational Linguistics*, pages 394–402, Los Angeles, California.

Lewis, W. D. and Lansing, E. (2013). Dramatically Reducing Training Data Size Through Vocabulary Saturation. In *WMT-2013: 8th Workshop on Statistical Machine Translation*, Sofia, Bulgaria.

Li, Z. and Khudanpur, S. (2008). Large-scale Discriminative n -gram Language Models for Statistical Machine Translation. In *AMTA-2008: 8th Conference of the Association for Machine Translation in the Americas, "MT at work"*, pages 21–25, Waikiki, Hawai'i.

Li, Z., Wang, Z., Khudanpur, S., and Eisner, J. (2010). Unsupervised Discriminative Language Model Training for Machine Translation using Simulated Confusion Sets. In *Coling-2010: 23rd International Conference on Computational Linguistics*, Beijing, China.

Liao, S., Wu, C., and Huerta, J. (2011). Evaluating human correction quality for machine translation from crowdsourcing. In *RANLP-2011: Recent Advances in Natural Language Processing*, pages 598–603, Hissar, Bulgaria.

Libovický, J., Helcl, J., Tlustý, M., Bojar, O., and Pecina, P. (2016). Cuni system for wmt16 automatic post-editing and multimodal translation tasks. In *Proceedings of the First Conference on Machine Translation*, pages 646–654, Berlin, Germany. Association for Computational Linguistics.

Logacheva, V., Blain, F., and Specia, L. (2016a). Usfd's phrase-level quality estimation systems. In *Proceedings of the First Conference on Machine Translation*, Berlin, Germany. Association for Computational Linguistics.

Logacheva, V., Hokamp, C., and Specia, L. (2015). Data enhancement and selection strategies for the word-level Quality Estimation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 330–335, Lisbon, Portugal. Association for Computational Linguistics.

Logacheva, V., Hokamp, C., and Specia, L. (2016b). Marmot: A toolkit for translation quality estimation at the word level. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC-2016)*, Portorož, Slovenia.

Logacheva, V., Lukasik, M., and Specia, L. (2016c). Metrics for evaluation of word-level machine translation quality estimation. In *ACL-2016: 54th Annual Meeting of the Association for Computational Linguistics*, ACL, pages 585–590, Berlin, Germany.

Logacheva, V. and Specia, L. (2014). Confidence-based Active Learning Methods for Machine Translation. In *EACL-2014: 14th Conference of the European Chapter of the Association for Computational Linguistics, Workshop on Humans and Computer-assisted Translation (HaCaT)*, pages 78–83, Gothenburg, Sweden.

Logacheva, V. and Specia, L. (2015a). Phrase-level quality estimation for machine translation. In *Proceedings of the 2015 International Workshop on Spoken Language Translation (IWSLT-2015)*, Da Nang, Vietnam.

Logacheva, V. and Specia, L. (2015b). the role of artificially generated negative data for quality estimation of machine translation. In *Proceedings of the 18th Annual Conference of the European Association for Machine Translation (EAMT-2015)*, Antalya, Turkey.

Lommel, A., Burchardt, A., Popović, M., Harris, K., Avramidis, E., and Uszkoreit, H. (2014). Using a New Analytic Measure for the Annotation and Analysis of MT Errors on Real Data. In *EAMT-2014: 18th Annual Conference of the European Association for Machine Translation*, pages 165–172.

Lopez, A. (2008). *Machine translation by pattern matching*. PhD dissertation, University of Maryland.

López-Salcedo, F.-J., Sanchis-Trilles, G., and Casacuberta, F. (2012). Online learning of log-linear weights in interactive machine translation. In *IberSPEECH-2012: Advances in Speech and Language Technologies for Iberian Languages*, Madrid, Spain.

Luong, N.-Q., Besacier, L., and Lecouteux, B. (2014a). An Efficient Two-Pass Decoder for SMT Using Word Confidence Estimation. In *EAMT-2014: Proceedings of the 18th Annual Conference of the European Association for Machine Translation*, Dubrovnik, Croatia.

Luong, N. Q., Besacier, L., and Lecouteux, B. (2014b). Lig system for word level qe task at wmt14. In *WMT-2014*, pages 335–341, Baltimore, USA.

Luong, N. Q., Besacier, L., and Lecouteux, B. (2014c). Lig system for word level qe task at wmt14. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 335–341, Baltimore, Maryland, USA. Association for Computational Linguistics.

Luong, N.-Q., Besacier, L., and Lecouteux, B. (2014d). Word Confidence Estimation for SMT N-best List Re-ranking. In *EACL-2014: 14th Conference of the European Chapter of the Association for Computational Linguistics, Workshop on Humans and Computer-assisted Translation*, pages 1–9, Gothenburg, Sweden.

Luong, N. Q., Lecouteux, B., and Besacier, L. (2013). LIG system for WMT13 QE task: Investigating the usefulness of features in word confidence estimation for MT. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 386–391, Sofia, Bulgaria. Association for Computational Linguistics.

Macklovitch, E. (2006). TransType2: The Last Word. In *LREC-2006: 5th International Conference on Language Resources and Evaluation*, pages 167–172, Genoa, Italy.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014a). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014b). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Marie, B. and Max, A. (2015). Touch-based pre-post-editing of machine translation output. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1040–1045, Lisbon, Portugal. Association for Computational Linguistics.

Martínez-Gómez, P., Sanchis-Trilles, G., and Casacuberta, F. (2012). Online adaptation strategies for statistical machine translation in post-editing scenarios. *Pattern Recognition*, 45(9).

Martins, A. F. T., Astudillo, R., Hokamp, C., and Kepler, F. (2016). Unbabel's participation in the wmt16 word-level translation quality estimation shared task. In *Proceedings of the First Conference on Machine Translation*, pages 806–811, Berlin, Germany. Association for Computational Linguistics.

Mathur, P., Cettolo, M., and Federico, M. (2013). Online Learning Approaches in Computer Assisted Translation. In *MT Summit XIV: 14th Machine Translation Summit, Workshop on Post-editing Technology and Practice*, pages 301–308, Nice, France.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Moore, R. C. and Lewis, W. D. (2010). Intelligent Selection of Language Model Training Data. In *ACL-2010: 48th Annual Meeting of the Association for Computational Linguistics*, pages 220–224, Uppsala, Sweden.

Moreau, E. and Vogel, C. (2012). Quality Estimation: an experimental study using unsupervised similarity measures. In *WMT-2012: 7th Workshop on Statistical Machine Translation*, pages 120–126.

Neal, R. M. and Hinton, G. E. (1999). A View of the EM Algorithm that Justifies Incremental, Sparse, and Other Variants. *Learning in graphical models*, pages 355–368.

Nepveu, L., Lapalme, G., and Foster, G. (2004). Adaptive Language and Translation Models for Interactive Machine Translation. In *EMNLP-2004: Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain.

Och, F. J. (2003). Minimum Error Rate Training in Statistical Machine Translation. In *ACL-2003: 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Okanohara, D. (2007). A Discriminative Language Model with Pseudo-Negative Samples. In *ACL-2007: 45th Annual Meeting of the Association for Computational Linguistics*, pages 73–80.

Ortiz-Martínez, D. and Casacuberta, F. (2014). The New Thot Toolkit for Fully-Automatic and Interactive Statistical Machine Translation. In *EACL-2014: 14th Conference of the European Chapter of the Association for Computational Linguistics, Demonstrations*, pages 45–48, Gothenburg, Sweden.

Ortiz-Martínez, D., García-Varea, I., and Casacuberta, F. (2010). Online Learning for Interactive Statistical Machine Translation. In *HLT-2010: Conference of the North American Chapter of the Association for Computational Linguistics*, pages 546–554, Los Angeles, California.

Pal, S., Zampieri, M., and van Genabith, J. (2016). Usaar: An operation sequential model for automatic statistical post-editing. In *Proceedings of the First Conference on Machine Translation*, pages 759–763, Berlin, Germany. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-j. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. In *ACL-2002: 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, USA.

Patel, R. N. and M, S. (2016). Translation quality estimation using recurrent neural network. In *Proceedings of the First Conference on Machine Translation*, Berlin, Germany. Association for Computational Linguistics.

Paul, M. (2009). Overview of the IWSLT 2009 Evaluation Campaign. In *IWSLT-2009: International Workshop on Spoken Language Translation*, pages 3–27, Tokio, Japan.

Pighin, D., Màrquez, L., and Formiga, L. (2012a). The FAUST Corpus of Adequacy Assessments for Real-World Machine Translation Output. In *LREC-2012: 8th international conference on Language Resources and Evaluation*, pages 29–35, Istanbul, Turkey.

Pighin, D., Màrquez, L., and May, J. (2012b). An Analysis (and an Annotated Corpus) of User Responses to Machine Translation Output. In *LREC-2012: 8th international conference on Language Resources and Evaluation*, volume 1, pages 1131–1136, Istanbul, Turkey.

Popović, M. (2011). Hjerson : An Open Source Tool for Automatic Error Classification of Machine Translation Output Maja Popović. *Prague Bulletin of Mathematical Linguistics*, 96:59–67.

Potet, M., Besacier, L., Blanchon, H., and Azouzi, M. (2012a). Towards a Better Understanding of Statistical Post-Edition Usefulness. In *IWSLT-2012: 9th International Workshop on Spoken Language Translation*, pages 284–291, Hong Kong, China.

Potet, M., Esperança-Rodier, E., Besacier, L., and Blanchon, H. (2012b). Collection of a Large Database of French-English SMT Output Corrections. In *LREC-2012: 8th International Conference on Language Resources and Evaluation*, pages 4043–4048, Istanbul, Turkey.

Potet, M., Esperança-Rodier, E., Blanchon, H., and Besacier, L. (2011). Preliminary Experiments on Using Users' Post-Editions to Enhance a SMT System. In *EAMT-2011: 15th Conference of the European Association for Machine Translation*, pages 161–168, Leuven, Belgium.

Povlsen, C. and Bech, A. (2001). Ape: Reducing the Monkey Business in Post-Editing by Automating the Task Intelligently. In *MT Summit VIII: Machine Translation in the Information Age*, pages 283–286, Santiago de Compostela, Spain.

Powers, D. M. (2011). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1):37–63.

Quirk, C. (2004). Training a Sentence-Level Machine Translation Confidence Measure. In *LREC-2004: 4th International Conference on Language Resources and Evaluation*, volume 4, pages 825–828.

Raybaud, S., Langlois, D., and Smaïli, K. (2011). "This sentence is wrong." Detecting errors in machine-translated sentences. *Machine Translation*, 25(1):1–34.

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *LREC-2010: 7th International Conference on Language Resources and Evaluation*, pages 45–50, Valletta, Malta.

Reichart, R. and Rappoport, A. (2007). An Ensemble Method for Selection of High Quality Parses. In *ACL-2007: 45th Annual Meeting of the Association for Computational Linguistics*, pages 408–415, Prague, Czech Republic.

Ringger, E., Mcclanahan, P., Haertel, R., Busby, G., Carmen, M., Carroll, J., Seppi, K., and Lonsdale, D. (2007). Active Learning for Part-of-Speech Tagging: Accelerating Corpus Annotation. In *LAW-2007: Linguistic Annotation Workshop*, pages 101–108, Prague, Czech Republic.

Rousseau, A. (2013). XenC : An Open-Source Tool for Data Selection in Natural Language Processing. *Prague Bulletin of Mathematical Linguistics*, 100(Oct 2013):73–82.

Rozovskaya, A. and Roth, D. (2010). Generating confusion sets for context-sensitive error correction. In *EMNLP-2010: Conference on Empirical Methods in Natural Language Processing*, pages 961–970, Cambridge, Massachusetts.

Rubino, R., Guilherme Camargo de Souza, J., Foster, J., and Specia, L. (2013a). Topic Models for Translation Quality Estimation for Gisting Purposes. In *MT Summit XIV: 14th Machine Translation Summit*, Nice, France.

Rubino, R., Huet, S., Lefèvre, F., and Linarès, G. (2012). Statistical Post-Editing of Machine Translation for Domain Adaptation. In *EAMT-2012: 16th Annual Conference of the European Association for Machine Translation*, pages 221–228, Trento, Italy.

Rubino, R., Wagner, J., Foster, J., Roturier, J., Samad Zadeh Kaljahi, R., and Hollowood, F. (2013b). DCU-Symantec at the WMT 2013 quality estimation shared task. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 392–397, Sofia, Bulgaria. Association for Computational Linguistics.

Saluja, A., Lane, I., and Zhang, Y. (2012). Machine Translation with Binary Feedback: a Large-Margin Approach. In *AMTA-2012: 10th Conference of the Association for Machine Translation in the Americas*, San Diego, CA.

Sanchis-Trilles, G., Ortiz-Martínez, D., Civera, J., Casacuberta, F., Vidal, E., and Hoang, H. (2008). Improving Interactive Machine Translation via Mouse Actions. In *EMNLP-2008: Conference on Empirical Methods in Natural Language Processing*, pages 485–494, Edinburgh, Scotland, UK.

Sarawagi, S. and Cohen, W. W. (2004). Semi-markov conditional random fields for information extraction. In *NIPS-2004: Advances in Neural Information Processing Systems 17*, pages 1185–1192.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, UK.

Schnabel, T., Labutov, I., Mimno, D., and Joachims, T. (2015). Evaluation methods for unsupervised word embeddings. In *EMNLP-2015: Conference on Empirical Methods in Natural Language Processing*, pages 298–307.

Schwenk, H. (2013). Cslm - a modular open-source continuous space language modeling toolkit. In *INTERSPEECH-2013: 14th Annual Conference of the International Speech Communication Association*, pages 1198–1202, Lyon, France.

Seigel, M. S. and Woodland, P. C. (2011). Combining information sources for confidence estimation with crf models. In *INTERSPEECH-2011: 12th Annual Conference of the International Speech Communication Association*, pages 905–908.

Sennrich, R., Schneider, G., Volk, M., and Warin, M. (2009a). A new hybrid dependency parser for German. In Chiarcos, C., de Castilho, R. E., and Stede, M., editors, *Von der Form zur Bedeutung: Texte automatisch verarbeiten / From Form to Meaning: Processing Texts Automatically. Proceedings of the Biennial GSCL Conference 2009*, pages 115–124, Tübingen.

Sennrich, R., Schneider, G., Volk, M., and Warin, M. (2009b). A new hybrid dependency parser for german. In *GSCL-2013: Gesellschaft für Sprachtechnologie Computerlinguistik*, Germany, Potsdam.

Servan, C., Le, N.-T., Luong, N. Q., Lecouteux, B., and Besacier, L. (2015). An Open Source Toolkit for Word-level Confidence Estimation in Machine Translation. In *The 12th International Workshop on Spoken Language Translation (IWSLT'15)*, Da Nang, Vietnam.

Settles, B. (2012). *Active Learning*. Morgan Claypool.

Shah, K., Bougares, F., Barrault, L., and Specia, L. (2016). Shef-lium-nn: Sentence level quality estimation with neural network features. In *Proceedings of the First Conference on Machine Translation*, pages 838–842, Berlin, Germany. Association for Computational Linguistics.

Shah, K., Cohn, T., and Specia, L. (2013). An investigation on the effectiveness of features for translation quality estimation. In *MT Summit XIV*, pages 167–174, Nice, France.

Shah, K., Logacheva, V., Paetzold, G., Blain, F., Beck, D., Bougares, F., and Specia, L. (2015). Shef-nn: Translation quality estimation with neural networks. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 342–347, Lisbon, Portugal. Association for Computational Linguistics.

Shang, L., Cai, D., and Ji, D. (2015). Strategy-based technology for estimating mt quality. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 348–352, Lisbon, Portugal. Association for Computational Linguistics.

Simard, M. and Foster, G. (2013). PEPr: Post-Edit Propagation Using Phrase-based Statistical Machine Translation. In *MT Summit XIV: 14th Machine Translation Summit*, pages 191–198, Nice, France.

Simard, M., Goutte, C., and Isabelle, P. (2007). Statistical Phrase-based Post-editing. In *NAACL-HLT-2007: Human Language Technology: the conference of the North American Chapter of the Association for Computational Linguistics*, pages 508–515, Rochester, NY.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A Study of Translation Edit Rate with Targeted Human Annotation. In *AMTA-2006: 7th Conference of the Association for Machine Translation in the Americas, "Visions for the Future of Machine Translation"*, pages 223–231, Cambridge, Massachusetts, USA.

Snover, M., Madnani, N., Dorr, B., and Schwartz, R. (2008). TERp System Description. In *AMTA-2008: 8th Conference of the Association for Machine Translation in the Americas, MetricsMATR workshop*, Waikiki, Hawai'i.

Snover, M., Madnani, N., Dorr, B. J., and Schwartz, R. (2009). Fluency, adequacy, or hter?: Exploring different human judgments with a tunable mt metric. In *WMT-2009: 4th Workshop on Statistical Machine Translation*, pages 259–268, Athens, Greece.

Soricut, R., Bach, N., and Wang, Z. (2012). The SDL Language Weaver Systems in the WMT12 Quality Estimation Shared Task. In *WMT-2012: 7th Workshop on Statistical Machine Translation*, pages 145–151.

Specia, L. (2011). Exploiting objective annotations for measuring translation post-editing effort. In *15th Conference of the European Association for Machine Translation*, EAMT, pages 73–80, Leuven, Belgium.

Specia, L., Cancedda, N., Dymetman, M., Turchi, M., and Cristianini, N. (2009). Estimating the Sentence-Level Quality of Machine Translation Systems. In *EAMT-2009: 13th Annual Conference of the European Association for Machine Translation*, pages 28–35.

Specia, L., Paetzold, G., and Scarton, C. (2015). Multi-level translation quality prediction with quest++. In *ACL-IJCNLP 2015 System Demonstrations*, pages 115–120, Beijing, China.

Specia, L., Shah, K., de Souza, J. G. C., and Cohn, T. (2013). QuEst - A translation quality estimation framework. In *ACL-2013: 51st Annual Meeting of the Association for Computational Linguistics, System demonstrations*, Sofia, Bulgaria.

Stolcke, A. (2002). Srilm — an extensible language modeling toolkit. In *ICSLP-2002 - INTERSPEECH-2002: 7th International Conference on Spoken Language Processing*, pages 901–904, Denver, Colorado, USA.

Tezcan, A., Hoste, V., Desmet, B., and Macken, L. (2015). Ugent-lt3 scate system for machine translation quality estimation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 353–360, Lisbon, Portugal. Association for Computational Linguistics.

Tezcan, A., Hoste, V., and Macken, L. (2016). Ugent-lt3 scate submission for wmt16 shared task on quality estimation. In *Proceedings of the First Conference on Machine Translation*, pages 843–850, Berlin, Germany.

Tiedemann, J. (2010). Context Adaptation in Statistical Machine Translation Using Models with Exponentially Decaying Cache. In *ACL-2010: 48th Annual Meeting of the Association for Computational Linguistics, Workshop on Domain Adaptation for Natural Language Processing*, pages 8–15, Uppsala, Sweden.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*, pages 142–147.

Turchi, M. and Negri, M. (2014). Automatic Annotation of Machine Translation Datasets with Binary Quality Judgements. In *LREC-2014: 9th International Conference on Language Resources and Evaluation*, pages 1788–1792, Reykjavik, Iceland.

Ueffing, N., Macherey, K., and Ney, H. (2003). Confidence Measures for Statistical Machine Translation. In *MT Summit IX: 9th Machine Translation Summit*, pages 394–401.

Ueffing, N. and Ney, H. (2005a). Application of Word-Level Confidence Measures in Interactive Statistical Machine Translation. In *EAMT-2005: 10th Annual Conference of the European Association for Machine Translation "Practical applications of machine translation"*, pages 262–270, Budapest, Hungary.

Ueffing, N. and Ney, H. (2005b). Word-level confidence estimation for machine translation using phrase-based translation models. In *HLT-EMNLP-2005: Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 763–770, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Vakil, Z. and Khadivi, S. (2012). A new search approach for interactive - predictive computer-assisted translation. In *Coling-2012: 20th International Conference on Computational Linguistics*, pages 1261–1270, Mumbai, India.

Vilar, D., Xu, J., D'Haro, L. F., and Ney, H. (2006). Error Analysis of Statistical Machine Translation Output. In *LREC-2006: 5th International Conference on Language Resources and Evaluation*, pages 697–702, Genoa, Italy.

Wäschle, K., Simianer, P., Bertoldi, N., Riezler, S., and Federico, M. (2013). Generative and Discriminative Methods for Online Adaptation in SMT. In *MT Summit XIV: 14th Machine Translation Summit*, pages 11–18, Nice, France.

Wisniewski, G. (2013). Design and Analysis of a Large Corpus of Post-Edited Translations: Quality Estimation, Failure Analysis and the Variability of Post-Edition. In *MT Summit XIV: 14th Machine Translation Summit*, pages 117–124, Nice, France.

Xiao, M. and Guo, Y. (2013). Online Active Learning for Cost Sensitive Domain Adaptation. In *CoNLL-2013: 17th Conference on Computational Natural Language Learning*, Sofia, Bulgaria.

Ye, N., Lee, W. S., Chieu, H. L., and Wu, D. (2004). Conditional Random Fields with High-Order Features for Sequence Labeling. *Journal of Web Semantics*, 2:2.

Yeh, A. (2000). More accurate tests for the statistical significance of result differences. In *Coling-2000: the 18th Conference on Computational Linguistics*, pages 947–953, Saarbrücken, Germany.

Yu, C.-N. J. and Joachims, T. (2009). Learning structural SVMs with latent variables. In *ICML-2009: 26th Annual International Conference on Machine Learning*, pages 1–8, Montr{é}al, Canada. ACM Press.

Zaretskaya, A., Corpas Pastor, G., and Sighiri, M. (2015). Translators' requirements for translation technologies: Results of a user survey. In *AIETI7 Conference: New Horizons in Translation and Interpreting Studies*, Malaga, Spain.

Zeman, D., Fishel, M., Berka, J., and Bojar, O. (2011). Addicter: What Is Wrong with My Translations? *Prague Bulletin of Mathematical Linguistics*, 96:79–88.

Zens, R. and Ney, H. (2006). N-Gram Posterior Probabilities for Statistical Machine Translation. In *WMT-2006: 1st Workshop on Statistical Machine Translation*, pages 72–77.

Appendix A

Quality Estimation models used for the experiments

In this chapter we list the full feature sets of all QE models used for the experiments.

A.1 Sentence-level QE models

A.1.1 sentence-17

This is a baseline sentence-level QE model which was used in Active learning experiments in chapter 6 and N-best list reranking and MT model re-tuning experiments in chapter 7. This model uses 17 features:

1. number of tokens in the source sentence,
2. number of tokens in the target sentence,
3. average source token length,
4. LM probability of the source sentence,
5. LM probability of the target sentence,
6. number of occurrences of the target word within the target hypothesis (averaged for all words in the hypothesis - type/token ratio),
7. average number of translations per source word in the sentence (as given by IBM 1 table thresholded such that prob(t|s) > 0.2),
8. average number of translations per source word in the sentence (as given by IBM 1 table thresholded such that prob(t|s) > 0.01) weighted by the inverse frequency of each word in the source corpus,

9. percentage of unigrams in quartile 1 of frequency (lower frequency words) in a corpus of the source language (SMT training corpus),
10. percentage of unigrams in quartile 4 of frequency (higher frequency words) in a corpus of the source language,
11. percentage of bigrams in quartile 1 of frequency of source words in a corpus of the source language,
12. percentage of bigrams in quartile 4 of frequency of source words in a corpus of the source language,
13. percentage of trigrams in quartile 1 of frequency of source words in a corpus of the source language,
14. percentage of trigrams in quartile 4 of frequency of source words in a corpus of the source language,
15. percentage of unigrams in the source sentence seen in a corpus (SMT training corpus)
16. number of punctuation marks in the source sentence,
17. number of punctuation marks in the target sentence.

A.1.2 sentence-79 (black-box features)

This QE model uses the full set of *black-box* sentence-level features — i.e. features that consider the MT system that generated the translations as a black box and do not use any information from it. This model with the full set of features was used in the N-best list reranking experiments with German–English pharmaceutical dataset (see the section 7.2.3). In addition to that, it was used as a basis for other sentence-level models (except the **sentence-17**) whose sets of features were selected from this 79-feature set with a feature selection method. The **sentence-79** model uses the following features:

1. number of tokens in the source sentence,
2. number of tokens in the target sentence,
3. ratio of number of tokens in the source and the target,
4. ratio of number of tokens in the target and the source,
5. absolute difference between the number tokens in the source and target normalised by the source length,
6. average source token length,
7. number of mismatched brackets in the target sentence,
8. number of mismatched quotation marks in the target sentence,
9. source sentence log probability,
10. source sentence perplexity,

11. source sentence perplexity without end of sentence marker,
12. target sentence log probability,
13. target sentence perplexity,
14. target sentence perplexity without end of sentence marker,
15. number of occurrences of the target word within the target hypothesis (averaged for all words in the hypothesis — type/token ratio),
16. average number of translations per source word in the sentence (probability > 0.01),
17. average number of translations per source word in the sentence (probability > 0.05),
18. average number of translations per source word in the sentence (probability > 0.1),
19. average number of translations per source word in the sentence (probability > 0.2),
20. average number of translations per source word in the sentence (probability > 0.5),
21. average number of translations per source word in the sentence (probability > 0.01) weighted by the frequency of each word in the source corpus,
22. average number of translations per source word in the sentence (probability > 0.05) weighted by the frequency of each word in the source corpus,
23. average number of translations per source word in the sentence (probability > 0.1) weighted by the frequency of each word in the source corpus,
24. average number of translations per source word in the sentence (probability > 0.2) weighted by the frequency of each word in the source corpus,
25. average number of translations per source word in the sentence (probability > 0.5) weighted by the frequency of each word in the source corpus,
26. average number of translations per source word in the sentence (probability > 0.01) weighted by the inverse frequency of each word in the source corpus,
27. average number of translations per source word in the sentence (probability > 0.05) weighted by the inverse frequency of each word in the source corpus,
28. average number of translations per source word in the sentence (probability > 0.1) weighted by the inverse frequency of each word in the source corpus,
29. average number of translations per source word in the sentence (probability > 0.2) weighted by the inverse frequency of each word in the source corpus,
30. average number of translations per source word in the sentence (probability > 0.5) weighted by the inverse frequency of each word in the source corpus,
31. average unigram frequency in quartile 1 of frequency (lower frequency words) in the corpus of the source sentence,
32. average unigram frequency in quartile 2 of frequency (lower frequency words) in the corpus of the source sentence,

33. average unigram frequency in quartile 3 of frequency (higher frequency words) in the corpus of the source sentence,
34. average unigram frequency in quartile 4 of frequency (higher frequency words) in the corpus of the source sentence,
35. average bigram frequency in quartile 1 of frequency (lower frequency words) in the corpus of the source sentence,
36. average bigram frequency in quartile 2 of frequency (lower frequency words) in the corpus of the source sentence,
37. average bigram frequency in quartile 3 of frequency (higher frequency words) in the corpus of the source sentence,
38. average bigram frequency in quartile 4 of frequency (higher frequency words) in the corpus of the source sentence,
39. average trigram frequency in quartile 1 of frequency (lower frequency words) in the corpus of the source sentence,
40. average trigram frequency in quartile 2 of frequency (lower frequency words) in the corpus of the source sentence,
41. average trigram frequency in quartile 3 of frequency (higher frequency words) in the corpus of the source sentence,
42. average trigram frequency in quartile 4 of frequency (higher frequency words) in the corpus of the source sentence,
43. percentage of distinct unigrams seen in the corpus (in all quartiles),
44. percentage of distinct bigrams seen in the corpus (in all quartiles),
45. percentage of distinct trigrams seen in the corpus (in all quartiles),
46. average frequency of the source words,
47. absolute difference between the number of periods in the source and the target,
48. absolute difference between the number of periods in the source and the target normalised by target length,
49. absolute difference between the number of commas in the source and the target,
50. absolute difference between the number of commas in the source and the target normalised by target length,
51. absolute difference between the number of colons in the source and the target,
52. absolute difference between the number of colons in the source and the target normalised by target length,
53. absolute difference between the number of semicolons in the source and the target,
54. absolute difference between the number of semicolons in the source and the target normalized by target length,

55. absolute difference between the number of question marks in the source and the target,
56. absolute difference between the number of question marks in the source and the target normalized by target length,
57. absolute difference between the number of exclamation marks in the source and the target,
58. absolute difference between the number of exclamation marks in the source and the target normalized by target length,
59. percentage of punctuation marks in the source,
60. percentage of punctuation marks in the target,
61. absolute difference between the number of punctuation marks in the source and the target normalised by target length,
62. percentage of numbers in the source sentence,
63. percentage of numbers in the target sentence,
64. absolute difference between number of numbers in the source and target sentence normalised by source sentence length,
65. number of source tokens that contain non-alphabetic symbols,
66. number of target tokens that contain non-alphabetic symbols,
67. ratio of percentages of tokens with non-alphabetic symbols in the source and the target,
68. percentage of content words in the source,
69. percentage of content words in the target,
70. ratio of percentages of content words in the source and target,
71. part-of-speech LM log-probability of target,
72. part-of-speech LM perplexity of target,
73. percentage of nouns in the source,
74. percentage of verbs in the source,
75. percentage of nouns in the target,
76. percentage of verbs in the target,
77. ratio of percentages of nouns in the source and target,
78. ratio of percentages of verbs in the source and target,
79. ratio of percentages of pronouns in the source and target.

A.1.3 syntactic features

This is a list of sentence-level syntactic features that was partially used in **sentence-62-syntax** and **sentence-50-syntax** models:

1. maximum depth of source dependency tree,

2. average depth of source dependency tree,
3. maximum width of source dependency tree (number of dependants of the root),
4. proportion of internal nodes of the source dependency tree (nodes with dependants, not root),
5. number of subjects in the source,
6. number of verbs with dependant subject in the source,
7. number of dependant clauses in the source,
8. number of verbs in the source,
9. number of nouns in the source,
10. number of conjunctions in the source,
11. source sentence starts with a verb (0/1),
12. maximum depth of target dependency tree,
13. average depth of target dependency tree,
14. maximum width of target dependency tree (number of dependants of the root),
15. proportion of internal nodes of the target dependency tree (nodes with dependants, not root),
16. number of subjects in the target,
17. number of verbs with dependant subject in the target,
18. number of dependant clauses in the target,
19. number of verbs in the target,
20. number of nouns in the target,
21. number of conjunctions in the target,
22. target sentence starts with a verb (0/1).

A.1.4 sentence-60

This model was used in AL (chapter 6), N-best list reranking (section 7.2.1) and retuning (section 7.3) experiments with English–German data. It uses a 60-feature subset of 79 black-box features (for the full list see section A.1.2) selected via feature selection experiments described in the section 6.2.2:

1. number of tokens in the source sentence,
2. number of tokens in the target sentence,
3. ratio of number of tokens in the source and the target,
4. ratio of number of tokens in the target and the source,
5. absolute difference between the number tokens in the source and target normalised by the source length,

6. average source token length,
7. source sentence log probability,
8. source sentence perplexity,
9. source sentence perplexity without end of sentence marker,
10. target sentence log probability,
11. target sentence perplexity,
12. target sentence perplexity without end of sentence marker,
13. number of occurrences of the target word within the target hypothesis (averaged for all words in the hypothesis — type/token ratio),
14. average number of translations per source word in the sentence (probability > 0.01),
15. average number of translations per source word in the sentence (probability > 0.05),
16. average number of translations per source word in the sentence (probability > 0.1),
17. average number of translations per source word in the sentence (probability > 0.5),
18. average number of translations per source word in the sentence (probability > 0.01) weighted by the frequency of each word in the source corpus,
19. average number of translations per source word in the sentence (probability > 0.05) weighted by the frequency of each word in the source corpus,
20. average number of translations per source word in the sentence (probability > 0.1) weighted by the frequency of each word in the source corpus,
21. average number of translations per source word in the sentence (probability > 0.2) weighted by the frequency of each word in the source corpus,
22. average number of translations per source word in the sentence (probability > 0.5) weighted by the frequency of each word in the source corpus,
23. average number of translations per source word in the sentence (probability > 0.01) weighted by the inverse frequency of each word in the source corpus,
24. average number of translations per source word in the sentence (probability > 0.05) weighted by the inverse frequency of each word in the source corpus,
25. average number of translations per source word in the sentence (probability > 0.1) weighted by the inverse frequency of each word in the source corpus,
26. average number of translations per source word in the sentence (probability > 0.2) weighted by the inverse frequency of each word in the source corpus,
27. average number of translations per source word in the sentence (probability > 0.5) weighted by the inverse frequency of each word in the source corpus,
28. average unigram frequency in quartile 1 of frequency (lower frequency words) in the corpus of the source sentence,

29. average unigram frequency in quartile 2 of frequency (lower frequency words) in the corpus of the source sentence,
30. average unigram frequency in quartile 3 of frequency (higher frequency words) in the corpus of the source sentence,
31. average bigram frequency in quartile 3 of frequency (higher frequency words) in the corpus of the source sentence,
32. average bigram frequency in quartile 4 of frequency (higher frequency words) in the corpus of the source sentence,
33. average trigram frequency in quartile 1 of frequency (lower frequency words) in the corpus of the source sentence,
34. average trigram frequency in quartile 2 of frequency (lower frequency words) in the corpus of the source sentence,
35. average trigram frequency in quartile 3 of frequency (higher frequency words) in the corpus of the source sentence,
36. average trigram frequency in quartile 4 of frequency (higher frequency words) in the corpus of the source sentence,
37. percentage of distinct unigrams seen in the corpus (in all quartiles),
38. percentage of distinct trigrams seen in the corpus (in all quartiles),
39. average frequency of the source words,
40. absolute difference between the number of commas in the source and the target,
41. absolute difference between the number of commas in the source and the target normalised by target length,
42. absolute difference between the number of colons in the source and the target,
43. absolute difference between the number of colons in the source and the target normalised by target length,
44. absolute difference between the number of semicolons in the source and the target,
45. absolute difference between the number of semicolons in the source and the target normalized by target length,
46. percentage of punctuation marks in the source,
47. percentage of punctuation marks in the target,
48. absolute difference between the number of punctuation marks in the source and the target normalised by target length,
49. percentage of numbers in the source sentence,
50. absolute difference between number of numbers in the source and target sentence normalised by source sentence length,
51. number of source tokens that contain non-alphabetic symbols,

52. number of target tokens that contain non-alphabetic symbols,
53. ratio of percentages of tokens with non-alphabetic symbols in the source and the target,
54. percentage of content words in the target,
55. ratio of percentages of content words in the source and target,
56. part-of-speech LM log-probability of target,
57. percentage of nouns in the source,
58. percentage of verbs in the source,
59. percentage of nouns in the target,
60. percentage of verbs in the target.

A group of features was removed because all these features had the same values for all sentences in the training data, therefore they were useless for the training. These features are:

1. number of mismatched brackets in the target sentence,
2. number of mismatched quotation marks in the target sentence,
3. absolute difference between the number of periods in the source and the target,
4. absolute difference between the number of periods in the source and the target normalised by target length,
5. absolute difference between the number of question marks in the source and the target,
6. absolute difference between the number of question marks in the source and the target normalized by target length,
7. absolute difference between the number of exclamation marks in the source and the target,
8. absolute difference between the number of exclamation marks in the source and the target normalized by target length,
9. ratio of percentages of pronouns in the source and target.

Other features were removed via feature selection.

A.1.5 sentence-62-syntax

This model was used in AL (chapter 6), N-best list reranking (section 7.2.1) and retuning (section 7.3) experiments with English–German data. This model contains 62 features which were selected from the set of 79 black-box and 22 syntactic features:

1. number of tokens in the source sentence,
2. number of tokens in the target sentence,
3. ratio of number of tokens in the source and the target,

4. ratio of number of tokens in the target and the source,
5. absolute difference between the number tokens in the source and target normalised by the source length,
6. average source token length,
7. source sentence log probability,
8. source sentence perplexity,
9. source sentence perplexity without end of sentence marker,
10. target sentence log probability,
11. target sentence perplexity,
12. target sentence perplexity without end of sentence marker,
13. number of occurrences of the target word within the target hypothesis (averaged for all words in the hypothesis — type/token ratio),
14. average number of translations per source word in the sentence (probability > 0.01),
15. average number of translations per source word in the sentence (probability > 0.05),
16. average number of translations per source word in the sentence (probability > 0.1),
17. average number of translations per source word in the sentence (probability > 0.5),
18. average number of translations per source word in the sentence (probability > 0.01) weighted by the frequency of each word in the source corpus,
19. average number of translations per source word in the sentence (probability > 0.05) weighted by the frequency of each word in the source corpus,
20. average number of translations per source word in the sentence (probability > 0.1) weighted by the frequency of each word in the source corpus,
21. average number of translations per source word in the sentence (probability > 0.2) weighted by the frequency of each word in the source corpus,
22. average number of translations per source word in the sentence (probability > 0.5) weighted by the frequency of each word in the source corpus,
23. average number of translations per source word in the sentence (probability > 0.01) weighted by the inverse frequency of each word in the source corpus,
24. average number of translations per source word in the sentence (probability > 0.1) weighted by the inverse frequency of each word in the source corpus,
25. average number of translations per source word in the sentence (probability > 0.2) weighted by the inverse frequency of each word in the source corpus,
26. average unigram frequency in quartile 1 of frequency (lower frequency words) in the corpus of the source sentence,
27. average unigram frequency in quartile 2 of frequency (lower frequency words) in the corpus of the source sentence,

28. average unigram frequency in quartile 3 of frequency (higher frequency words) in the corpus of the source sentence,
29. average unigram frequency in quartile 4 of frequency (higher frequency words) in the corpus of the source sentence,
30. average bigram frequency in quartile 1 of frequency (lower frequency words) in the corpus of the source sentence,
31. average bigram frequency in quartile 3 of frequency (higher frequency words) in the corpus of the source sentence,
32. average trigram frequency in quartile 2 of frequency (lower frequency words) in the corpus of the source sentence,
33. average trigram frequency in quartile 3 of frequency (higher frequency words) in the corpus of the source sentence,
34. percentage of distinct unigrams seen in the corpus (in all quartiles),
35. percentage of distinct bigrams seen in the corpus (in all quartiles),
36. average frequency of the source words,
37. absolute difference between the number of commas in the source and the target normalised by target length,
38. absolute difference between the number of semicolons in the source and the target,
39. absolute difference between the number of semicolons in the source and the target normalized by target length,
40. percentage of punctuation marks in the source,
41. absolute difference between the number of punctuation marks in the source and the target normalised by target length,
42. number of target tokens that contain non-alphabetic symbols,
43. ratio of percentages of tokens with non-alphabetic symbols in the source and the target,
44. percentage of content words in the source,
45. ratio of percentages of content words in the source and target,
46. part-of-speech LM log-probability of target,
47. part-of-speech LM perplexity of target,
48. percentage of verbs in the source,
49. percentage of verbs in the target,
50. ratio of percentages of verbs in the source and target,
51. average depth of source dependency tree,
52. proportion of internal nodes of the source dependency tree (nodes with dependants, not root),
53. number of subjects in the source,

54. number of dependant clauses in the source,
55. number of verbs in the source,
56. number of nouns in the source,
57. number of conjunctions in the source,
58. maximum width of target dependency tree (number of dependants of the root),
59. number of verbs with dependant subject in the target,
60. number of verbs in the target,
61. number of conjunctions in the target,
62. target sentence starts with a verb (0/1).

Other features were removed via the feature selection algorithm or because they had the same values for all instances of the training set (see section A.1.4)).

A.1.6 sentence-50-syntax

This model was used in the N-best list reranking experiments with German–English data (section 7.2.3). It contains a 50-feature subset of 79 black-box + 22 syntactic features:

1. number of tokens in the source sentence,
2. number of tokens in the target sentence,
3. ratio of number of tokens in the source and the target,
4. ratio of number of tokens in the target and the source,
5. absolute difference between the number tokens in the source and target normalised by the source length,
6. average source token length,
7. source sentence log probability,
8. source sentence perplexity without end of sentence marker,
9. average number of translations per source word in the sentence (probability > 0.1),
10. average number of translations per source word in the sentence (probability > 0.5),
11. average number of translations per source word in the sentence (probability > 0.01) weighted by the frequency of each word in the source corpus,
12. average number of translations per source word in the sentence (probability > 0.05) weighted by the frequency of each word in the source corpus,
13. average number of translations per source word in the sentence (probability > 0.2) weighted by the frequency of each word in the source corpus,
14. average number of translations per source word in the sentence (probability > 0.5) weighted by the frequency of each word in the source corpus,

15. average number of translations per source word in the sentence (probability > 0.01) weighted by the inverse frequency of each word in the source corpus,
16. average number of translations per source word in the sentence (probability > 0.2) weighted by the inverse frequency of each word in the source corpus,
17. average number of translations per source word in the sentence (probability > 0.5) weighted by the inverse frequency of each word in the source corpus,
18. average unigram frequency in quartile 1 of frequency (lower frequency words) in the corpus of the source sentence,
19. average unigram frequency in quartile 2 of frequency (lower frequency words) in the corpus of the source sentence,
20. average unigram frequency in quartile 3 of frequency (higher frequency words) in the corpus of the source sentence,
21. average unigram frequency in quartile 4 of frequency (higher frequency words) in the corpus of the source sentence,
22. average bigram frequency in quartile 2 of frequency (lower frequency words) in the corpus of the source sentence,
23. average bigram frequency in quartile 3 of frequency (higher frequency words) in the corpus of the source sentence,
24. average bigram frequency in quartile 4 of frequency (higher frequency words) in the corpus of the source sentence,
25. average trigram frequency in quartile 1 of frequency (lower frequency words) in the corpus of the source sentence,
26. average trigram frequency in quartile 3 of frequency (higher frequency words) in the corpus of the source sentence,
27. average trigram frequency in quartile 4 of frequency (higher frequency words) in the corpus of the source sentence,
28. absolute difference between the number of commas in the source and the target normalised by target length,
29. absolute difference between the number of colons in the source and the target,
30. absolute difference between the number of colons in the source and the target normalised by target length,
31. percentage of numbers in the source sentence,
32. number of source tokens that contain non-alphabetic symbols,
33. number of target tokens that contain non-alphabetic symbols,
34. percentage of content words in the source,
35. ratio of percentages of content words in the source and target,

36. part-of-speech LM log-probability of target,
37. part-of-speech LM perplexity of target,
38. ratio of percentages of verbs in the source and target,
39. maximum width of source dependency tree (number of dependants of the root),
40. proportion of internal nodes of the source dependency tree (nodes with dependants, not root),
41. number of subjects in the source,
42. number of verbs with dependant subject in the source,
43. number of verbs in the source,
44. number of nouns in the source,
45. source sentence starts with a verb (0/1),
46. maximum depth of target dependency tree,
47. average depth of target dependency tree,
48. number of verbs in the target,
49. number of nouns in the target,
50. target sentence starts with a verb (0/1).

A.2 Word-level QE models

A.2.1 word-BL

This is a baseline word-level QE model. It was used in the N-best list reranking (section 7.2) and MT system retuning (section 7.3) experiments. It uses the following set of features:

1. number of tokens in the source,
2. number of tokens in the target,
3. ratio of numbers of tokens in the source and target,
4. target token,
5. left target context,
6. right target context,
7. source token (source token aligned to the target token),
8. left source context,
9. right source context,
10. target token is a stopword (0/1),
11. target token is a punctuation mark (0/1),
12. target token is a proper noun,
13. target token is a number,

14. highest order of ngram that includes target token and its left context,
15. highest order of ngram that includes target token and its right context,
16. backoff behavior of the trigram $t_{i-2}t_{i-1}t_i$ where t_i is target token,
17. backoff behavior of the trigram $t_{i-1}t_it_{i+1}$,
18. backoff behavior of the trigram $t_it_{i+1}t_{i+2}$,
19. highest order of ngram that includes source token and its left context,
20. highest order of ngram that includes source token and its right context,
21. POS of the target token,
22. POS of the source token.

A.2.2 word-extended

This is an extended word-level QE model. It was used in the N-best list reranking (section 7.2) and MT system retuning (section 7.3) experiments. It uses the baseline feature set (section A.2.1) extended with the following features:

1. target token + source token,
2. target token + left context,
3. target token + right context,
4. target token + left source context,
5. target token + right source context,
6. target POS + source POS.
7. type of dependency between the target token and its head word,
8. target token + type of dependency,
9. target token + its head word,
10. POS of the target token + POS of its head,
11. target token + token's closest sibling to the left,
12. POS of the target token + POS of its closest sibling to the left,
13. target token + token's closest sibling to the right,
14. POS of the target token + POS of its closest sibling to the right,
15. target token + its head word + head word of the head (grandhead),
16. target token's POS + its head's POS + its grandhead's POS,
17. type of dependency between the target token and its head word,
18. source token + type of dependency,
19. source token + its head word,
20. POS of the source token + POS of its head,
21. source token + token's closest sibling to the left,

22. POS of the source token + POS of its closest sibling to the left,
23. token + source token's closest sibling to the right,
24. POS of the source token + POS of its closest sibling to the right,
25. source token + its head word + head word of the head (grandhead),
26. source token's POS + its head's POS + its grandhead's POS.

A.2.3 word-decoding

This word-level model was used in our experiments on incorporation of QE feature into MT decoder (section 7.4). It uses the following feature set:

1. target token,
2. left context of the target token,
3. source token,
4. left context of the source token,
5. right context of the source token,
6. target token is a stopword
7. target token is a punctuation mark,
8. target token is a proper noun
9. target token is a digit,
10. target token POS-tag,
11. source token POS-tag,
12. target token + left context,
13. target token + source token,
14. target POS-tag + source POS-tag,
15. target token + left token + source token.

A.3 Phrase-level QE models

A.3.1 phrase-BL

This phrase-level QE model was used as a baseline in WMT-16 QE shared task. We also used it in our N-best list reordering experiments in chapter 7 (see section 7.2). The model contains the following list of features:

1. percentage of numbers in the source,
2. percentage of numbers in the target,

3. absolute difference between number of numbers in the source and target phrase normalised by the source phrase length,
4. percentage of source words that contain non-alphabetic symbols,
5. percentage of target words that contain non-alphabetic symbols,
6. ratio of percentage of alphabetical tokens in the source and alphabetical tokens in the target,
7. average unigram frequency in quartile 1 of frequency in the corpus of the source language,
8. average unigram frequency in quartile 2 of frequency in the corpus of the source language,
9. average unigram frequency in quartile 3 of frequency in the corpus of the source language,
10. average unigram frequency in quartile 4 of frequency in the corpus of the source language,
11. average bigram frequency in quartile 1 of frequency in the corpus of the source language,
12. average bigram frequency in quartile 2 of frequency in the corpus of the source language,
13. average bigram frequency in quartile 3 of frequency in the corpus of the source language,
14. average bigram frequency in quartile 4 of frequency in the corpus of the source language,
15. average trigram frequency in quartile 1 of frequency in the corpus of the source language,
16. average trigram frequency in quartile 2 of frequency in the corpus of the source language,
17. average trigram frequency in quartile 3 of frequency in the corpus of the source language,
18. average trigram frequency in quartile 4 of frequency in the corpus of the source language,
19. percentage of distinct source unigrams seen in a corpus of the source language (in all quartiles),
20. percentage of distinct source bigrams seen in a corpus of the source language (in all quartiles),
21. percentage of distinct source trigrams seen in a corpus of the source language (in all quartiles),

22. target phrase LM probability,
23. target phrase LM perplexity,
24. average number of translations per source word in the phrase (probability > 0.01),
25. average number of translations per source word in the phrase (probability > 0.05),
26. average number of translations per source word in the phrase (probability > 0.1),
27. average number of translations per source word in the phrase (probability > 0.2),
28. average number of translations per source word in the phrase (probability > 0.5),
29. average number of translations per source word in the phrase (probability > 0.01) weighted by the frequency of each word in the source corpus,
30. average number of translations per source word in the phrase (probability > 0.5) weighted by the frequency of each word in the source corpus,
31. average number of translations per source word in the phrase (probability > 0.1) weighted by the frequency of each word in the source corpus,
32. average number of translations per source word in the phrase (probability > 0.2) weighted by the frequency of each word in the source corpus,
33. average number of translations per source word in the phrase (probability > 0.5) weighted by the frequency of each word in the source corpus,
34. absolute difference between number of periods in the source and target phrases,
35. absolute difference between number of commas in the source and target phrases,
36. absolute difference between number of colons in the source and target phrases,
37. absolute difference between number of semicolons in the source and target phrases,
38. absolute difference between number of question marks in the source and target phrases,
39. absolute difference between number of exclamation marks in the source and target phrases,
40. absolute difference between number of periods in the source and target phrases normalised by target length,
41. absolute difference between number of commas in the source and target phrases normalised by target length,
42. absolute difference between number of colons in the source and target phrases normalised by target length,
43. absolute difference between number of semicolons in the source and target phrases normalised by target length,
44. absolute difference between number of question marks in the source and target phrases normalised by target length,
45. absolute difference between number of exclamation marks in the source and target phrases normalised by target length,

46. percentage of punctuation marks in the source phrase,
47. percentage of punctuation marks in the target phrase,
48. absolute difference between number of punctuation marks between the source and target phrases normalised by target length,
49. source phrase LM probability,
50. source phrase LM perplexity,
51. number of tokens in the target phrase,
52. number of tokens in the source phrase,
53. ratio of number of tokens in the source and number of tokens in the target,
54. ratio of number of tokens in the target and number of tokens in the source,
55. average target token length,
56. average source token length,
57. average number of occurrences of the target word within the target phrase,
58. percentage of unaligned words,
59. percentage of words aligned with more than one word,
60. average number of aligned words per word,
61. percentage of content words in the source phrase,
62. percentage of content words in the target phrase,
63. percentage of verbs in the source phrase,
64. percentage of verbs in the target phrase,
65. percentage of nouns in the source phrase,
66. percentage of nouns in the target phrase,
67. percentage of pronouns in the source phrase,
68. percentage of pronouns in the target phrase,
69. ratio of percentage of content words in the source and target,
70. ratio of percentage of nouns in the source and target phrases,
71. ratio of percentage of verbs in the source and target phrases,
72. ratio of percentage of pronouns in the source and target phrases.

A.3.2 phrase-extended

This model was also used in our N-best list reranking experiments (section 7.2). It uses the set of baseline phrase-level features (section A.3.1) and additional features:

1. out-of-vocabulary words (0/1),
2. source left context,
3. target left context,

4. source right context,
5. target right context,
6. highest order of ngram that includes the first target word,
7. highest order of ngram that includes the last target word,
8. backoff behavior of first target trigram,
9. backoff behavior of last target trigram,
10. named entities in the source (0/1),
11. named entities in the target (0/1),
12. part of speech of the source left context,
13. part of speech of the source right context,
14. part of speech of the target left context,
15. part of speech of the target right context.

Appendix B

Details of the experiments

This chapter contains the details of the experiments.

B.1 Results of the metric comparison experiments

The table B.1 contains the results of the experiments on metrics comparison on the real data (section 5.2.2): all the systems submitted to the WMT-15 shared task on word-level QE are evaluated with all the six tested metrics and ranked with respect to these metrics. The figure B.1 contains the statistic significance of differences between each pair of systems for each of the metrics. Here, every row and column stand for a system, with systems sorted according to their rank under a particular metric, every cell denotes the statistic significance of the pair of systems corresponding to the cell's row and column number. Each cell is painted in a shade of gray denoting the p-value: from 0 (white cell, significant difference) to 1 (black cell, difference is not significant).

B.2 Results of the retuning experiments

Here we report the full results of the retuning experiments described in the section 7.3 and shown in the figures 7.16 and 7.17. The table B.2 contains the BLEU and METEOR scores for all the combinations of tested QE models (sentence-level, phrase-level, word-level) and experimental settings (tuning started with previously tuned or untuned feature weights).

F_1-BAD	
UAlacant+Baseline	0.431
QUETCHPLUS	0.430
UAlacant	0.415
SAU CRF	0.391
SAU SLG-CRF	0.389
SHEF W2V	0.384
SHEF W2V-SIM	0.384
QuEst++ arow	0.383
UGENT HYBRID	0.367
DCU-SHEF 2K	0.366
QUETCH	0.352
DCU-SHEF 5K	0.345
QuEst++ pa	0.343
all-bad	0.317
pessimistic	0.312
UGENT MBL	0.305
DCU s5-RTM	0.239
DCU s4-RTM	0.226
random	0.191
optimistic	0.172
BASELINE	0.167
all-good	0.0

F_1-multiplied	
QUETCHPLUS	0.341
SAU KERC-CRF	0.337
UAlacant+Baseline	0.336
SAU SLG-CRF	0.336
UAlacant	0.315
UGENT HYBRID	0.304
QUETCH	0.298
DCU-SHEF 5K	0.291
SHEF W2V	0.275
SHEF W2V-SIM	0.274
DCU-SHEF 2K	0.272
QuEst++ arow	0.259
UGENT MBL	0.257
DCU s5-RTM	0.21
DCU s4-RTM	0.2
optimistic	0.156
random	0.155
BASELINE	0.149
QuEst++ pa	0.083
pessimistic	0.055
all-bad	0.0
all-good	0.0

MCC	
UAlacant+Baseline	0.268
QUETCHPLUS	0.267
optimistic	0.263
SAU CRF	0.255
SAU SLG-CRF	0.253
UAlacant	0.243
UGENT HYBRID	0.201
QUETCH	0.198
SHEF W2V	0.194
SHEF W2V-SIM	0.193
QuEst++ arow	0.192
DCU-SHEF 5K	0.19
DCU-SHEF 2K	0.169
DCU s5-RTM	0.158
DCU s4-RTM	0.153
UGENT MBL	0.149
QuEst++ pa	0.135
BASELINE	0.134
random	0.003
pessimistic	0.001
all-bad	0.0
all-good	0.0

SeqCor	
SAU CRF	0.35
UAlacant+Baseline	0.349
SAU SLG-CRF	0.349
UAlacant	0.337
QUETCH	0.334
QUETCHPLUS	0.327
DCU-SHEF 5K	0.324
UGENT HYBRID	0.314
DCU-SHEF 2K	0.314
UGENT MBL	0.298
SHEF W2V	0.287
SHEF W2V-SIM	0.287
QuEst++ arow	0.284
QuEst++ pa	0.263
DCU s5-RTM	0.241
random	0.238
DCU s4-RTM	0.229
pessimistic	0.216
optimistic	0.191
BASELINE	0.16
all-good	0.086
all-bad	0.0005

Phrase F_1-BAD	
QuEst++ pa	0.572
all-bad	0.536
pessimistic	0.499
UAlacant+Baseline	0.422
UAlacant	0.415
QuEst++ arow	0.387
QUETCHPLUS	0.385
SHEF W2V	0.38
SHEF W2V-SIM	0.378
DCU-SHEF 2K	0.374
SAU CRF	0.336
SAU SLG-CRF	0.334
DCU-SHEF 5K	0.308
QUETCH	0.301
UGENT HYBRID	0.299
UGENT MBL	0.236
DCU s5-RTM	0.223
DCU s4-RTM	0.21
optimistic	0.158
BASELINE	0.145
random	0.139
all-good	0.0

Phrase F_1-mult	
SAU CRF	0.195
SAU SLG-CRF	0.193
UAlacant+Baseline	0.19
UAlacant	0.183
DCU-SHEF 5K	0.17
QUETCHPLUS	0.168
QUETCH	0.163
DCU-SHEF 2K	0.16
UGENT HYBRID	0.147
SHEF W2V	0.139
SHEF W2V-SIM	0.137
DCU s5-RTM	0.134
QuEst++ arow	0.129
DCU s4-RTM	0.124
UGENT MBL	0.123
QuEst++ pa	0.119
optimistic	0.098
BASELINE	0.081
random	0.064
pessimistic	0.05
all-bad	0.0
all-good	0.0

Table B.1 Rankings of systems from WMT15 shared task produced by different metrics.

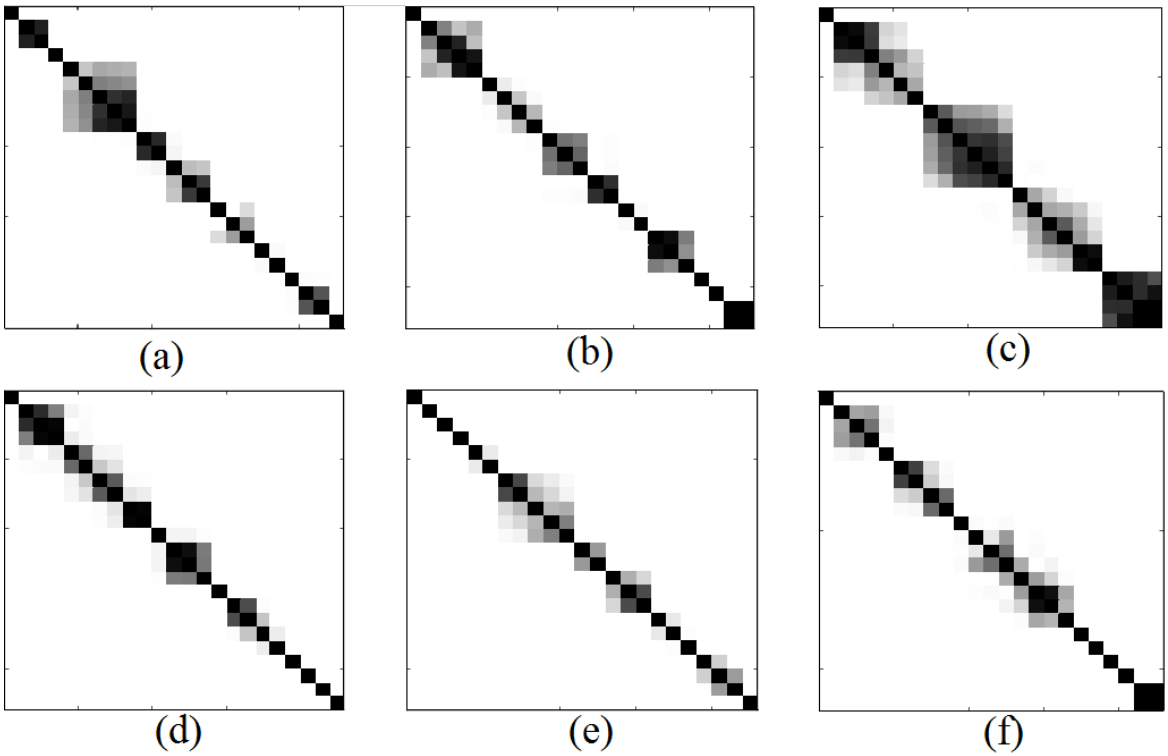

Fig. B.1 P-values between metrics for F_1-BAD (a), F_1-mult (b), MCC (c), SeqCor (d), Phrase F_1-BAD (e), Phrase F_1-mult (f). Every cell is a pair of compared systems. Higher p-values (= no significant difference between the two systems) denoted with darker squares

QE model		Initial weights	BLEU	METEOR
Baseline			**25.4**	44.1
sentence-17		untuned	15.8	35.4
		tuned	25.2	44.7
sentence-60		untuned	10.8	26.3
		tuned	25.5*	44.5
sentence-62-syntax		untuned	11.4	31.5
		tuned	25.0	**45.2**
word-BL	probability	untuned	16.1	33.0
		tuned	25.3*	44.4*
	score	untuned	5.6	20.8
		tuned	25.4*	44.4
word-extended	probability	untuned	11.3	26.8
		tuned	25.5*	44.6
	score	untuned	20.5	39.1
		tuned	25.5*	44.8
phrase-BL	probability	untuned	18.3	37.7
		tuned	25.2	44.3*
	score	untuned	19.9	38.5
		tuned	25.4*	44.4
phrase-extended	probability	untuned	9.5	24.0
		tuned	25.5*	44.8
	score	untuned	16.2	34.5
		tuned	25.4*	44.5

Table B.2 Results of the retuning experiment on the English–German data. Asterisk (*) denotes systems which are not significantly different from the baseline.