Space-Aware Subword Tokenisation and Complex Word Processing in Language Models

Edward Gow-Smith

Supervisor: Aline Villavicencio

A thesis submitted for the degree of Doctor of Philosophy

in the

School of Computer Science

August 2025

Declaration

I, Edward Gow-Smith, hereby declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree. Except where stated otherwise by reference or acknowledgement, the work presented is entirely my own.

Acknowledgements

Firstly, I would like to thank my primary supervisor, Professor Aline Villavicencio, for all the support, input, opportunities, and care for my wellbeing that she has given to me during my PhD journey.

Secondly, I would like to thank the other members of my supervisory team, Dr Carolina Scarton and Dr Cass Zhao, for their input and guidance, along with Dr Harish Tayyar Madabushi, for passing on some of his wealth of knowledge in the formative months of my PhD. I also extend thanks to my external examiner Professor Hinrich Schütze, and my internal examiner Dr Nafise Sadat Moosavi. Thanks to Dr Mark Stevenson for providing important support during the first months of my time in Sheffield. My PhD wouldn't have existed without the the Centre for Doctoral Training (CDT) in Speech and Language Technologies at Sheffield, for which I'm grateful to Professor Thomas Hain and Professor Rob Gaizauskas, who brought it into existence and have continued to work to maintain its quality, and have with it taken on the task of marking all our yearly e-Portfolio submissions. Dr Stuart Wrigley and Elizabeth Pass, in their capacities as administrators for the CDT, have been invaluable at answering my many questions and pointing me in the right direction whenever needed.

The research herein wouldn't have been possible without my colleagues at Sheffield providing stimulating discussions, feedback on early versions of my work, and, of course, emotional support. For that, I'm indebted to Dylan Phelps, Tom Pickard, Maggie Mi, Atsuki Yamaguchi, Dr Wei He, Dr George Close, Tomas Goldsack, and Jasivan Sivakumar. Many of these have been collaborators on work that lies outside of this thesis, but nevertheless done whilst at Sheffield, and of importance to me. For that, I additionally want to mention Dr Marcos Garcia and Professor Marco Idiart for their work on the SemEval-2022 Shared Task on Multilingual Idiomaticity Detection and Sentence Embedding; Xuan-Rui Fan for his work on sample efficient approaches to idiomaticity detection; Dr Danae Sanchez Villegas for her work on the AmericasNLP 2023 Machine Translation Shared Task; Dr Xingyi Song for collaboration on work for EAMT 2025; Dr Mark McConville, Professor Roibeard Ó Maolalaigh, Professor William Gillies, and Doctor Jade Scott for work on the transliteration of the Book of Dean of Lismore; and Jacqueline Rowe for her work on the machine translation of Guinea-Bissau Creole under my supervision.

Finally, this work was supported by the UKRI AI Centre for Doctoral Training in Speech and Language Technologies (SLT) and their Applications funded by UK Research and Innovation (grant number EP/S023062/1).

Abstract

This work investigates the limitations of how subword tokenisers handle spaces, and how the processing of spaces impacts the performance of language models, with a particular focus on the processing of complex words. Motivated by previous work which shows tokeniser limitations, we propose a simple and effective modification to state-of-the-art subword tokenisers—treating spaces as individual tokens—that ameliorates known issues with the morphological validity of tokenisers such as BPE, Unigram, and WordPiece, especially regarding the splitting of prefixes. We extrinsically evaluate our space-aware tokeniser variants (BPE$'$, Unigram$'$, WordPiece$'$) through pretraining a number of encoder-only transformer language models, and finetuning them on a range of downstream tasks, in the general domain, and for the processing of complex words. For the latter, to extend the analysis of complex word processing beyond English for the first time, we introduce a new dataset (mCWIF) in English, German, Turkish, and Finnish. Across datasets, our space-aware tokenisers substantially improve performance on complex word classification for English, German, and Finnish, with inconsistent results in Turkish. We suggest that the Turkish results arise because topically-relevant subwords in Turkish occur either at the start of a word or not, but very rarely in both positions. In the general domain, we also find that we can remove all word boundary information from sequences and retain equivalent performance, and that such information doesn't boost performance when included either explicitly through the input or implicitly through the pretraining task. Our work contributes to understanding of the impact of subword tokenisation, the limitations of state-of-the-art subword tokenisers, and how they can be improved for complex word processing. The main research questions are: What is the impact of poor-quality tokenisation, and what causes issues with subword tokenisers? How can subword tokenisers be improved? Can we introduce space information in an alternative way to improve the performance of language models?

Contents

Chapter 1

Introduction

1.1 Overview

For text to be processed by computers, some initial conversion process must take place. In the field of Natural Language Processing (NLP), where data is processed by large neural network architectures such as GPT-4 (OpenAI, 2023) and Llama (Dubey et al., 2024), this involves a first step where the text is converted into subunits, which can then be vectorised and passed as input to the network. This process is known as **tokenisation**. Tokenisation can create units at any level of hierarchy. Work has been done to train transformer architectures (Vaswani et al., 2017) on units at the multi-word level (Tayyar Madabushi et al., 2021), the word level (Feng et al., 2024), the character level (El Boukkouri et al., 2020; Clark et al., 2022), or the byte level (Xue et al., 2022). But, by far the most common approach currently is to use a **subword tokeniser**, which splits text into subunits determined by frequency—more common words are represented as single units, whilst rarer words and strings may be broken down to the character level. Such subword tokenisers include BPE (Sennrich et al., 2016; Gage, 1994), Unigram (Kudo, 2018), and WordPiece (Schuster and Nakajima, 2012). Both GPT-4 and Llama use a tokeniser based on BPE.

With such tokenisers, many words will be composed of multiple subunits. Previous work has suggested that, for rare, complex words, language models compute the meaning from a composition of the meaning of these subunits (Hofmann et al., 2021). This suggests that we should care about how well tokenisers are able to align with the morphology of a language, since morphemes are the smallest meaningful constituents of words (Haspelmath and Sims, 2013); if an English speaker had never encountered the word "unbeatable", for example, they could nevertheless likely determine the meaning from its three morphemes "un", "beat", and "able", and it is expected that language models exhibit similar behaviour for complex words in general, which we define for this thesis as words comprised of multiple morphemes, in line with prior work (Anglin et al., 1993; Booij, 2012).

In Table 1.1, we show the tokenisations generated for GPT-4 and Llama-3 for the word

Input	Tokenisation
unbeatable	_un, **be, atable**
unicycle	**_un, icycle**
unequal	**_une, qual**
unsaturated	**_uns, aturated**
unaccessible	_un, accessible

Table 1.1: Tokenisations generated by GPT-4 and Llama-3 for five words with the prefix "un". Red, bold text indicates token boundaries which do not align with the morphemes of the word.

"unbeatable", along with four other words containing the prefix "un". We note that these tokenisers are slightly different, but generate the same tokenisations for these words. In the case of "unbeatable", we can see that, whilst the prefix "un" is correctly split, the rest of the word is tokenised incorrectly. In three of the other cases, the prefix is incorrectly split (e.g. "un" appears in the tokenisation of "unicycle", rather than the correct prefix "uni"). For "unaccessible", a less common synonym for "inaccessible", these models do tokenise it validly.

This simple analysis raises three questions:

Question 1: What is the impact of poor-quality tokenisations? The examples given in Table 1.1 are likely fairly high frequency, having been seen by the model many times during training. It is thus expected that the meaning of these words will be stored somehow in the weights of the model. But, what about for complex words in general, where many are rarer and might have been seen only a few times in training, or not at all? Will subpar tokenisation hinder the ability of language models to process them? In the course of our work (Chapters 3 to 5), we pretrain a large number of models to investigate this question and show that the answer is **yes**, with more morphologically valid tokenisation giving better performance on complex word classification across languages: an effect that doesn't diminish with increasing model scale in our experiments. In Chapters 3 and 4, our analysis is on English only, but we extend this to German, Turkish, and Finnish in Chapter 5 through the introduction of a new dataset.

Question 2: What causes these issues in tokenisation? In Chapter 3, we hypothesise that the treatment of whitespace by the current state-of-the-art subword tokenisation algorithms is a big contributor to the poor tokenisations seen in Table 1.1. In particular, the fact that space symbols (an encoding of whitespace in the form of ## or _) are prepended to tokens leads to issues, as during tokeniser training merges are computed with the inclusion of these symbols. This means substrings are handled differently depending on if they occur at the start of a word or not, leading to a particular issue with prefixes (such as "un").

Question 3: How can subword tokenisers be improved concerning poor-quality tokenisation? We introduce a modification to subword tokenisers in Chapter 3 which forces them to always treat spaces as individual tokens. Applying this to BPE, Unigram, and WordPiece to give their space-aware variants (BPE′, Unigram′, WordPiece′), we perform a quantitative and qualitative analysis of the generated tokenisations, finding consistently more valid segmentations, more aligned with morphology. Across Chapters 3 to 5, we pretrain and finetune a large number of encoder-only models with both tokeniser variants, finding that the space-aware tokenisers lead to improved performance on complex word processing across languages, except Turkish, which we investigate in Chapter 5.

As a result of our work in Chapter 3, we find similar performance on GLUE (Wang et al., 2018) when we modify our tokeniser to treat spaces as individual tokens, and then remove the spaces. The resulting model is thus trained on sequences with no word boundary information, which raises a further question:

Question 4: Can we introduce space information in an alternative way to improve the performance of language models? In Chapter 4, we investigate methods for modifying a RoBERTa architecture (Liu et al., 2019) to include space (word boundary) information, either directly through the input, or through a modification to the pretraining task. None of our modified models improve upon the version without space information, which leads us to hypothesise that such information is not useful for models, and that the morpheme is the subunit of most importance.

1.2 Contributions

- We investigate how current subword tokenisation algorithms handle spaces, and find that a simple modification, always treating whitespace as individual tokens, leads to much improved morphological validity.
- We explain that a reason for the low morphological validity of subword tokenisation algorithms discussed in previous work, in particular when handling prefixes, is due to the use of word-boundary characters prepended to subwords.
- We evaluate our modified tokenisers along with the defaults in the pretrain-finetune paradigm, looking at general domain tasks, along with complex word classification. We find a significant performance difference in the classification of complex words, which strengthens our hypothesis that more morphologically valid tokenisers im-

prove the processing of complex words by language models. We also find equivalent performance in the general domain when spaces are removed.

- Given these findings, we look at whether the space information can be included in an alternative way to improve performance. We modify the default RoBERTa architecture to pass space information through the input (by concatenating additional positional embeddings), or through the pretraining task (by adding an additional masked language modelling head).
- We introduce a novel multilingual dataset for complex word classification (mCWIF), allowing us to evaluate how the performance impact of tokenisation varies across languages. This dataset is in four languages: English, German, Turkish, and Finnish, chosen to span a range of morphological complexities. Internet forums are used as a basis for data extraction, allowing easy extension to other languages, which we hope will stimulate further research.
- We use this dataset to evaluate the performance of WordPiece′ vs WordPiece, finding the biggest performance improvement for German, then English, then Finnish, with an inconsistent improvement for Turkish.
- We investigate factors which may result in the different performance gaps across languages. We introduce the metrics of position diversity and consistency to understand how the handling of space prefixes may hinder WordPiece's performance, with effects that differ across languages. In particular, we see that topically relevant substrings in Turkish almost always occur either at the start of a word, or not, but never a mixture, which we suggest limits the positive effect that WordPiece′ will have on performance.

1.3 Thesis Overview

This thesis follows the publication format, consisting of three publications each taking up one chapter (Chapters 3 to 5), alongside a background chapter (Chapter 2) and a conclusion (Chapter 6). Here, we list the publications that comprise our work:

Improving Tokenisation by Alternative Treatment of Spaces In this paper, we introduce a novel modification to existing subword tokenisation algorithms which forces them to handle whitespace as individual tokens. We find that this ameliorates known issues with the linguistic validity of tokenisers, in particular their poor performance at splitting prefixes from complex words. We then evaluate this modification extrinsically by pretraining and finetuning encoder-only transformer models using the default tokenisers for BPE and Unigram, along with our modified space-aware versions (BPE′ and WordPiece ′). We find there to be no significant difference in general-domain tasks, but a substantial performance difference when processing complex words. This result bolsters the hypothesis that how well a model is able to segment a complex word correlates with its ability to process such a word.

My contributions to this work: conceptualisation, programming, experiments, evaluation, analysis, and writing.

This work was published in Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (Gow-Smith et al., 2022).

Word Boundary Information Isn't Useful for Encoder Language Models In this paper, we look at alternative methods of including whitespace (word boundary) information in transformer models, through either modifications to the input, or to the pretraining task. Given our previous work showing that BPE′ and Unigram′ perform equivalently to BPE and Unigram on general-domain tasks, despite having no knowledge of word boundaries, we are interested in whether this information is somehow useful to the model, when used in an alternative way. We introduce implicit and explicit space-variants of RoBERTa, either passing space information as an additional position embedding, or adding another masked language modelling head. Across English and Finnish, we find no performance improvement from these model variants, which indicates that word boundary information isn't useful for models, suggesting that morphemes are the most important subunit.

My contributions to this work: conceptualisation, programming, experiments, evaluation, analysis, and writing.

This work was published in Proceedings of the 9th Workshop on Representation Learning for NLP (RepL4NLP-2024) (Gow-Smith et al., 2024).

Multilingual Complex Word Classification From Internet Forums for Evaluating Subword Tokenisation In this paper, we introduce a dataset for the classification of complex words across four languages (English, German, Turkish, Finnish), allowing for the novel investigation of the multilingual impact of tokenisation on the processing of complex words. These datasets give an increased coverage of complex words compared to previous work, using only length as a filter. Using this dataset, we compare the performance of WordPiece′ to WordPiece, finding the biggest performance improvement for German, and the least for Turkish. We investigate this further by looking at potential linguistic and tokenisation factors across languages. In particular, we introduce two new metrics (position diversity and tokenisation consistency) to evaluate how the issue of space prefixes hinders WordPiece for a given language. In Turkish, we find that topically-relevant substrings almost always occupy a position at the start of a word, or a position not at the start, but very rarely a mixture. We suggest that WordPiece$'$'s alternative handling of space prefixes will thus have almost no impact for Turkish topicality classification, when it comes to the recurrence of semantically relevant substrings. On the other hand, German has the highest position diversity and lowest resulting consistency for WordPiece, which we suggest results in the largest performance improvement for WordPiece$'$.

My contributions to this work: conceptualisation, programming, data collection, dataset creation, experiments, evaluation, analysis, and writing.

This work is under review at CoNLL 2025.

Chapter 2

Background

2.1 Tokenisation

Tokenisation is the task of splitting input text into smaller units, known as **tokens**. These tokens correspond to elements in a **vocabulary**, each element having a unique numerical ID: these IDs then allow computation to be carried out as part of an NLP model. Tokenisation is the most fundamental and thus ubiquitous text processing step, and almost all modern NLP systems will perform tokenisation of some form (Mielke et al., 2021). Here, we give a brief history of tokenisation in NLP.

Early discussions of tokenisation as a step for processing text include Webster and Kit (1992) and Grefenstette and Tapanainen (1994), with the tokens in these cases being taken as words. More broadly, the use of "word tokenization" in relation to the compression of textual data was introduced by Long et al. (1985). Although these are the first works that explicitly discuss tokenisation, approaches to parsing text and handling word-tokens stretch further back in the information retrieval literature (Van Rijsbergen, 1975). In languages with no spaces between words, the problem of tokenisation is closely tied to that of word segmentation, and the importance of this was recognised and discussed in many early works (Palmer, 2000; Chen and Liu, 1992; Chiang et al., 1992).

For languages with spaces between words, such as English, simple string-searching techniques using regular expressions (Thompson, 1968) can be used, and were used in early NLP systems. One example is the tokeniser used in the Penn Treebank (Taylor et al., 2003), now implemented as part of the Natural Language Toolkit (NLTK) along with a range of other regular expression based tokenisers (Bird et al., 2009). The Stanford Tokenizer (Manning et al., 2014) is another tokeniser that builds on the Penn Treebank standard, this time implemented in Java.

Non-contextualised word embedding approaches used such regular expression based methods of tokenisation. In word2vec (Mikolov et al., 2013) they do not explicitly mention tokenisation, although in their code[1] they simply take spaces, tabs and new-line characters as word boundaries. They also perform some work tokenising commonly occurring phrases as single tokens, extending the simple word-based approach to a coarser granularity. In GloVe (Pennington et al., 2014), they use the Stanford Tokenizer, mentioned previously. ELMo (Peters et al., 2018) builds on GloVe, and thus the same tokenisation method is implemented.

Tokenisers built using rule-based pattern matching (including regular expressions) have drawbacks:

- They require large vocabulary sizes to cover all of the textual data they are trained on: problematic when using neural models, as large input layer sizes are needed which increases model size and computation.
- They cannot handle out-of-vocabulary words encountered at inference time. Such words are commonly replaced with [UNK] tokens.
- They do not work for languages without spaces between words (such as Chinese, Japanese, Korean).

Due to these issues, **subword tokenisation** was introduced, initially in the field of neural machine translation (NMT), but now used in virtually all NLP models. The first explicit mention (and popularisation) of this approach is by Sennrich et al. (2016), although it was introduced indirectly by Schuster and Nakajima (2012). This is an **unsupervised method**, learning from training data to build a vocabulary, and then tokenising text at inference time using this learned vocabulary and a tokenisation algorithm. In this approach, more frequent words will be represented as single tokens, with rarer words being broken down into multiple subword tokens, possibly down to the character level, thus handling the out-of-vocabulary problem and reducing the required vocabulary size. It should be noted that the algorithm for building the vocabulary and the algorithm for tokenising text using the vocabulary are distinct, and a set vocabulary could be defined for such models. These models do not need spaces for segmentation, which allows them to handle languages without them.

There are various subword tokenisation algorithms that are used in contemporary NLP systems, although most are variants on the two most popular: **Byte Pair Encoding (BPE)** (Sennrich et al., 2016; Gage, 1994) and **Unigram** (Kudo, 2018). Both of these algorithms are implemented in the SentencePiece library (Kudo and Richardson, 2018), which also implements extensions including subword regularisation (Kudo, 2018) for Unigram and BPE-dropout (Provilkov et al., 2020) for BPE.

BPE was initially introduced for data compression (Gage, 1994), and then later adapted for tokenisation (Sennrich et al., 2016). The algorithm is trained by starting

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

with a vocabulary consisting of individual characters occurring in the training data, and then iteratively merging the most frequently occurring bigram of vocabulary items until the desired vocabulary size is reached. The generated merge rules are then applied in order at inference time to tokenise text. For example, the word "the" may be tokenised by a succession of merges: "t" "h" "e" $\rightarrow$ "th" "e" $\rightarrow$ "the". Unigram was introduced by Kudo (2018). The algorithm works by starting with a vocabulary consisting of all substrings occurring more than once in the training data, and then iteratively removing the vocabulary items with the lowest Unigram language model loss until the desired vocabulary size is reached. At inference, a Viterbi algorithm is used along with the learned language modelling parameters to tokenise text. It should be noted that a similar algorithm was introduced much earlier (Creutz and Lagus, 2005), with the difference being a penalty for token length. BERT (Devlin et al., 2019) uses **WordPiece** tokenisation, which was introduced by Schuster and Nakajima (2012) but only named as such when used as part of Google's NMT system (Wu et al., 2016). WordPiece is similar to BPE, but merges are chosen based on the resultant increase in likelihood of an n-gram language model, rather than their frequency of occurrence.

2.2 Morphology

Morphology is variously defined as "the study of the internal structure of words" (Haspelmath and Sims, 2013), the study of "forms of words" (Matthews, 1991), or the "study of words and their structure" (Bauer, 2003). For the purpose of this thesis, our interest in morphology is an interest in how meaningful subunits are combined to form words, and how subword tokenisation aligns with this. Morphology splits morphemes into categories of **roots** and **affixes**, which are further broken down into **prefixes**, added to the start of a base, **suffixes**, added to the end of a base, and **infixes**, added to the middle of a base (which are rare in English). The combination of these morphemes gives **complex words**, i.e. those words comprised of multiple morphemes (Anglin et al., 1993; Booij, 2012). In the case of "unbeatable", for example, it can be analysed as a combination of the prefix "un", the root "beat", and the suffix "able". In general, complex words can either have a concatenative or non-concatenative morphological parse, although a concatenative parse is the most common across languages (Sproat, 1992). There is also work which suggests that non-concatenative effects are described wholly by phonology (Svenonius and Bye, 2011). In our work, we focus on concatenative morphology, and there is evidence that non-concatenative morphology poses greater problems for language models (Amrhein and Sennrich, 2021).

2.2.1 Morphological Classification

It is generally accepted that there are two types of morphology, **inflectional** and **derivational** (Carstairs-McCarthy, 2017; Bauer, 2003). Given a base form, inflectional morphology refers to the different word-forms of a lexeme from that base, whereas derivational morphology refers to the different lexemes formed from that base (Bauer, 2003). Starting from the root "beat", inflected forms include "beats" and "beaten", whilst derivational forms include "unbeatable" or "beater". Nevertheless, there are also morphemes, such as "ing" which do not fit easily into one of these two categories, and there are thus suggestions that inflection and derivation exist on a spectrum (Bochner, 1993). Inflectional morphemes do not typically add meaning to the base, whereas derivational morphemes do (Haspelmath and Sims, 2013). As such, one could expect that it would be more important to correctly tokenise derivationally complex words than inflectional ones.

2.2.2 Morphological Typology

Different languages have different morphological structures, and can be classified based on them. Languages can be grouped into isolating, agglutinating, fusional, or polysynthetic (Comrie, 1989). An isolating language is one where every word is made of a single morpheme (e.g. Yoruba and Vietnamese). An agglutinating language is one where words can consist of multiple morphemes, but the boundaries between morphemes are always clear, with each morpheme encoding a single feature (e.g. Turkish). A fusional language is one where the boundary between morphemes isn't always clear, with each "morpheme" encoding multiple morphological features (e.g. Russian). Polysynthetic languages are those where many morphemes may be combined into a single word, and these may be either fusional or agglutinative (e.g. Western Greenlandic). In Chapter 5, we introduce a dataset for evaluating subword tokenisation across languages with different morphological complexities, and that form complex words in different ways. Specifically, we look at the impact of tokenisation on Turkish and Finnish (morphologically complex, agglutinative languages) as well as German and English (morphologically more simple, non-agglutinative languages).

Chapter 3

Publication I: Improving Tokenisation by Alternative Treatment of Spaces

Edward Gow-Smith[1], Harish Tayyar Madabushi[2], Carolina Scarton[1], Aline Villavicencio[1,3]

[1]Department of Computer Science, University of Sheffield
[2]Department of Computer Science, University of Bath
[3]Institute of Data Science and Artificial Intelligence, University of Exeter

Abstract

Tokenisation is the first step in almost all NLP tasks, and state-of-the-art transformer-based language models all use subword tokenisation algorithms to process input text. Existing algorithms have problems, often producing tokenisations of limited linguistic validity, and representing equivalent strings differently depending on their position within a word. We hypothesise that these problems hinder the ability of transformer-based models to handle complex words, and suggest that these problems are a result of allowing tokens to include spaces. We thus experiment with an alternative tokenisation approach where spaces are always treated as individual tokens. Specifically, we apply this modification to the BPE and Unigram algorithms. We find that our modified algorithms lead to improved performance on downstream NLP tasks that involve handling complex words, whilst having no detrimental effect on performance in general natural language understanding tasks. Intrinsically, we find our modified algorithms give more morphologically correct tokenisations, in particular when handling prefixes. Given the results of our experiments, we advocate for always treating spaces as individual tokens as an improved tokenisation method.

3.1 Introduction

Tokenisation is a key initial step in processing natural language, as it identifies the linguistic units to be processed, converting them to numerical IDs which can then be vectorised and manipulated by mathematical operations.

Earlier NLP approaches used simple string-searching techniques with regular expressions to tokenise text; however, these pattern-matching tokenisation methods have drawbacks: they require large vocabulary sizes to cover the training data, they cannot handle out-of-vocabulary words, and they do not work for languages without spaces as word boundaries. To address these issues, subword tokenisation was introduced. The first explicit mention (and popularisation) of this approach was by Sennrich et al. (2016), though it was indirectly introduced earlier by Schuster and Nakajima (2012). This method works by learning from training data to build a vocabulary (of a fixed size) and then tokenising text at inference time using this vocabulary (and possibly other learnt parameters). More frequent words are represented as single tokens, with rare words being broken down into multiple subword tokens, possibly down to the character level.

State-of-the art transformer-based language models all use subword tokenisation algorithms based on either byte-pair encoding (BPE) (Sennrich et al., 2016; Gage, 1994) or Unigram (Kudo, 2018). The original transformer model (Vaswani et al., 2017) uses BPE, whilst BERT (Devlin et al., 2019), which consists of a transformer encoder pretrained with a masked language modelling objective, uses WordPiece tokenisation (Schuster and Nakajima, 2012), which is a variant of BPE with a language model loss function. WordPiece is also used by ERNIE (Sun et al., 2019), DistilBERT (Sanh et al., 2019), ELECTRA (Clark et al., 2020), StructBERT (Wang et al., 2020) and NEZHA (Wei et al., 2019). GPT-2 (Radford et al., 2019) introduced byte-level BPE, operating on byte sequences rather than Unicode code points, which allows all sequences to be encoded using a base vocabulary of 256, avoiding the issue of unknown characters. The same approach is used in RoBERTa (Liu et al., 2019), DeBERTa (He et al., 2021), and BART (Lewis et al., 2020).

The BPE and Unigram algorithms are implemented in the SentencePiece library (Kudo and Richardson, 2018). There is a lack of clarity regarding SentencePiece in the literature, with it being erroneously considered as its own algorithm rather than an implementation of other algorithms. For example, in the paper introducing T5 (Raffel et al., 2020) they state that they "use SentencePiece to encode text as WordPiece tokens", which is not in fact implemented in SentencePiece. Looking at their code, we find that they use the default SentencePiece implementation, which is Unigram. XLNET (Yang et al., 2019) say they tokenise with SentencePiece, but do not say which algorithm they use—again, looking at their code, we find they use the default of Unigram. Equivalently, ALBERT (Lan et al., 2020) say that they tokenise with SentencePiece as for XLNET, meaning they again use Unigram.

Despite their ubiquity, existing tokenisation algorithms have problems, which we hypothesise hinders the ability of language models to handle complex words (Section 3.2). We suggest that these problems are pervasive across all existing subword tokenisation algorithms due to a shared fundamental design choice of allowing tokens to include spaces, and thus experiment with an alternative treatment of spaces where they are always taken as individual tokens. We implement this approach by making simple modifications to the existing WordPiece, BPE, and Unigram algorithms (Section 3.3). We first evaluate our modified algorithms intrinsically (Section 3.4), quantitatively finding that they improve morphological correctness, in particular when handling prefixes. Qualitatively, we take examples from previous papers critiquing existing tokenisation algorithms, and show how our modified algorithms are able to alleviate the discussed issues. We then evaluate our modified algorithms extrinsically by pretraining and finetuning transformer-based models (Section 3.5), showing that they give improved performance on NLP tasks that require handling complex words with no detrimental effect on performance in the general domain.

3.2 Problems with Existing Tokenisation Algorithms

Existing tokenisation algorithms often produce unintuitive tokenisations for complex words, incorrectly splitting prefixes, and producing unmeaningful subword tokens, which are problems that have been discussed in previous works. Church (2020) looks at the BERT (WordPiece) tokenisations for complex words, highlighting the many unnatural tokenisations that arise, with tokens often splitting up morphemes and digraphs. Nayak et al. (2020) also discuss the issues with BERT's tokeniser, specifically highlighting problems with the splitting of prefixes, and they show that poor tokenisation leads to weak semantic representations. Hofmann et al. (2021) find that BERT performs poorly on classifying complex words containing prefixes, performing much better on suffixes. They suggest that a reason is that BERT's tokeniser is seldom accurate for splitting prefixes, but is much more often correct for splitting suffixes. Schick and Schütze (2020) argue that a reason BERT struggles to understand rare words is due to suboptimal tokenisation of these words. Here we give a few of our own examples of BERT tokenisations that illustrate the problems[1]:

joint → _joint
jointed → _joint, ed
disjointed → _di, s, jo, int, ed
unisex → _un, ise, x

[1]BERT's tokeniser actually prepends the space symbol to subword units not occurring at the start of words, and the space symbol they use is "##" rather than "_", but these are inconsequential differences and we standardise the output here for clarity.

true → _true
untrue → _un, tr, ue
estimate → _estimate
overestimate → _over, est, imate

We see here that the prefixed words are tokenised poorly: the prefix is either incorrectly split, as in "disjointed" and "unisex", or the prefix is correctly split, but the rest of the word is tokenised differently from the standalone case, as in "untrue" and "overestimate". We note that suffixes are handled better than prefixes, which is due to spaces being prepended rather than appended to words (see Section 3.3).

For these latter examples, there is a second problem: even if the base were tokenised as a single token, the addition of the space symbol means that there would be no explicit link between the prefixed word and the standalone base. As an example, we cherry-pick a rare example of a morphologically correct tokenisation by BERT of a word containing a prefix, showing both strings and token IDs:

beatable → _beat, able (3786, 3085)
unbeatable → _un, beat, able (4895, 19442, 3085)

We can see that, even though these tokenisations are reasonable, the subword "beat" is assigned different IDs in the two cases due to the prepending of the special space symbol.

We hypothesise that both of these problems hinder the ability of existing language models (such as BERT) to deal with complex words. Regarding the first problem, we argue that the morphological correctness of a tokeniser is a metric which will correlate with the ability of language models to deal with complex words: correctly splitting affixes means morphologically related words (those sharing a common base) are given related tokenisations. The splitting of prefixes is particularly important in English, as English prefixes always have a semantic function, unlike suffixes which can have both syntactic and semantic functions (Giraudo and Grainger, 2003). Also, tokenisations made up of meaningful subword tokens (morphemes or groups of morphemes) will allow language models to build stronger representations with less data, since the representations of complex words can be computed from the representations of the subwords. Regarding the second problem, the fact that base forms are represented differently depending on their position within a word means a reduction in relevant training instances and hence a further weakening of representations for complex words.

3.3 Our Modified Algorithms

We suggest that the problems discussed in Section 3.2 arise as a result of how spaces are handled by existing algorithms: All subword tokenisation algorithms currently used by

(a) Default BPE

Training

```
input  : training data $T$, vocabulary size $s$
output: vocabulary $V$
1 Replace whitespace in $T$ with the space symbol
2 **Prepend the space symbol to the first word of every sentence in $T$[4]**
3 Vocabulary $V$ initialised as all characters
4 while $|V| < s$ do
5     Find the most frequently occurring bigram in $T$**, only allowing spaces as the first character**
6     Apply merge operation on the bigram to make a new token
7     Add merge operation to $V$
8 end
```

Tokenisation

```
input  : text $T$, vocabulary $V$
output: tokens $\tau$
1 Replace whitespace in $T$ with the space symbol
2 **Prepend the space symbol to the first word of every sentence in $T$**
3 Apply the merge operations from $V$ in order to $T$.
```

(b) Modified BPE (BPE′)

Training

```
input  : training data $T$, vocabulary size $s$
output: vocabulary $V$
1 Replace whitespace in $T$ with the space symbol
2 Vocabulary $V$ initialised as all characters
3 while $|V| < s$ do
4     Find most frequently occurring bigram in $T$ *that does not include spaces*
5     Apply merge operation on the bigram to make a new token
6     Add merge operation to $V$
7 end
```

Tokenisation

```
input  : text $T$, vocabulary $V$
output: tokens $\tau$
1 Replace whitespace in $T$ with the space symbol
2 Apply the merge operations from $V$ in order to $T$.
```

Figure 3.1: Default and modified BPE algorithms. Red, bold text is removed from the default algorithm, whilst green, italic text is added.

transformer-based models allow tokens to include space symbols as the first character[2]. This means equivalent strings are treated differently depending on whether they appear at the start of a word or not. This difference occurs when training these tokenisers, which leads to suboptimal tokenisations of prefixed words. It also occurs when using these tokenisers in NLP models, leading to equivalent strings being assigned different tokens depending on whether they occur at the start of a word or not.

Thus, to attempt to alleviate these issues, and hence improve the handling of complex words by language models, we propose an alternative treatment of spaces where they are always assigned individual tokens. This simple modification can be made to any existing subword tokenisation algorithm, though for brevity we focus our attention on BPE and Unigram; this modification can also be made to the WordPiece algorithm, and we see similar (intrinsic) performance improvements from doing so. In Section 3.4, we perform a qualitative analysis of our modified WordPiece algorithm and also include the default WordPiece algorithm in our quantitative evaluation for comparison. Our modified algorithms and the defaults are shown in Figure 3.1 and Figure 3.2 for BPE and Unigram, respectively[3].

In the following sections, we compare our modified tokenisation algorithms to the defaults by evaluating them intrinsically (Section 3.4) and extrinsically (Section 3.5).

[2]Splitting on spaces occurs as a first step, so space symbols cannot occur in the middle of tokens. The default implementation splits before spaces, meaning space symbols occur only at the start of words.

[3]We release code for training our modified algorithms, as well as running both our intrinsic and extrinsic experiments at `https://github.com/edwardgowsmith/improved-tokenisation-methods`

(a) Default Unigram

Training

```
input  : training data $T$, vocabulary size $s$
output: vocabulary $V$, language model parameters $\Theta$
1 Replace whitespace in $T$ with the space symbol
2 **Prepend the space symbol to the first word of every sentence in $T$**
3 Vocabulary $V$ initialised as all substrings occurring in $T$**, only allowing spaces as the first character**[5]
4 while $|V| > s$ do
5     Optimise a Unigram language model with parameters $\Theta$ to fit the data using the EM algorithm
6     For each substring in $V$, compute the loss from removing this from the vocabulary
7     Remove the substring with the smallest loss from $V$
8 end
```

Tokenisation

```
input  : text $T$, vocabulary $V$, language model parameters $\Theta$
output: tokens $\tau$
1 Replace whitespace in $T$ with the space symbol
2 **Prepend the space symbol to the first word of every sentence in $T$**
3 Use the Viterbi algorithm with the learned language modelling parameters and the vocabulary to tokenise $T$
```

(b) Modified Unigram (Unigram′)

Training

```
input  : training data $T$, vocabulary size $s$
output: vocabulary $V$, language model parameters $\Theta$
1 Replace whitespace in $T$ with the space symbol
2 Vocabulary $V$ initialised as all substrings occurring in $T$ *that do not include spaces, plus the space symbol*
3 while $|V| > s$ do
4     Optimise a Unigram language model with parameters $\Theta$ to fit the data using the EM algorithm
5     For each substring in $V$, compute the loss from removing this from the vocabulary
6     Remove the substring with the smallest loss from $V$
7 end
```

Tokenisation

```
input  : text $T$, vocabulary $V$, language model parameters $\Theta$
output: tokens $\tau$
1 Replace whitespace in $T$ with the space symbol
2 Use the Viterbi algorithm with the learned language modelling parameters and the vocabulary to tokenise $T$ *with spaces being given an arbitrarily high score so they are always selected as individual tokens*
```

Figure 3.2: Default and modified Unigram algorithms. Red, bold text is removed from the default algorithm, whilst green, italic text is added.

3.4 Intrinsic Evaluation: Morphological Correctness

Given our hypothesis that the morphological correctness of a tokeniser, especially when handling prefixes, correlates with the performance of language models in dealing with complex words (Section 3.2), we perform a controlled intrinsic evaluation of our tokenisers using this metric. We train our modified algorithms and the defaults on 1 million sentences from English Wikipedia for BPE and Unigram, with a fixed vocabulary size of 16 000, and then run evaluation on four morphological datasets: LADEC, MorphoLex, MorphyNet and DagoBERT.

The LADEC dataset (Gagné et al., 2019) consists of 7 804 noun compounds with a unique morphological parse (we exclude those with multiple parses). MorphoLex (Sánchez-Gutiérrez et al., 2018) provides derivational morphology for 68 624 entries from the English Lexicon Project (Balota et al., 2007). Here we only consider those with a concatenative parse (i.e. no overlapping tokens), resulting in 12 028 entries. MorphyNet (Batsuren et al., 2021) provides derivational and inflectional morphology for words across 15 languages, expanding the UniMorph dataset (McCarthy et al., 2020). Taking only those derivational morphology entries in English with a concatenative parse gives 193 945 entries. The DagoBERT dataset (Hofmann et al., 2020a) comprises 279 443 words containing low-frequency derivatives, taken from Reddit posts. Again, we take those with a concatenative parse, giving 268 513 entries.

We evaluate a tokeniser on these datasets using the evaluation method introduced by Creutz et al. (2004), which produces metrics by comparing the boundaries of a generated tokenisation with a gold standard reference: false negatives are boundaries appearing in the reference but not in the generated tokenisation, whilst false positives are boundaries appearing in the generated tokenisation but not in the reference. Because it makes sense to store common words as single tokens in the vocabulary, even if they can be decomposed into morphemes, we report precision along with F1 as a potentially more meaningful metric, since this allows undersegmentation whilst penalising oversegmentation. We also compute the mean sequence length (number of tokens) for each tokeniser across each dataset. Results are shown in Table 3.1. Here, and throughout, the prime symbol ($'$) denotes the given algorithm modified to always treat spaces as individual tokens.

The general trend is that Unigram outperforms BPE (consistent with findings by Bostrom and Durrett 2020, Hofmann et al. 2022), with the modified algorithms performing better than their default counterparts — the average F1 scores across the four datasets are 43.0, 50.9, 59.7, and 62.4 for the four algorithms BPE, BPE$'$, Unigram, and Unigram$'$, respectively. On the MorphoLex dataset, however, the default Unigram algorithm performs the best. This is also the only dataset where default Unigram gives a shorter mean sequence length than Unigram$'$. To further investigate this, we evaluate on the subsets of the data containing only prefixed and only suffixed entries, shown in Table 3.2. We can see that Unigram$'$ performs best on prefixed entries, but worse than default Unigram on suffixed entries. Since the dataset consists of many more entries containing suffixes than those containing prefixes (7 422 vs 2 692), this could explain the performance difference. Because the correct tokenisation of prefixed words is particularly important (Section 3.2), we believe that this performance trade-off is beneficial. In Section 3.5, we confirm this through evaluation on downstream, showing BPE$'$ and Unigram$'$ to outperform their default counterparts on complex word classification.

Interestingly, BPE$'$ gives the shortest sequence length on three of the four datasets, but not the most morphologically correct tokenisations. Since BPE was developed as a compression algorithm, the short sequence lengths are perhaps expected, but here we only see a weak correlation between sequence length and morphological correctness[6].

For a qualitative analysis, we take examples from papers that highlight problems with existing tokenisers (Section 3.2) and generate the output from the default and modified algorithms for BPE and Unigram, shown in Table 3.3. These examples illustrate how

[4]In the standard implementation, space symbols are added at the start of sentences so that words are equivalent whether they appear at the start of a sentence or not.

[5]This is only tractable for languages that include spaces. For languages without them, other initialisation methods must be used.

[6]Pearson's correlation of between -0.262 and -0.857 depending on the dataset.

	LADEC			MorphoLex			MorphyNet			DagoBERT		
	Seq. Length	Precision	F1	Seq. Length	Precision	F1	Seq. Length	Precision	F1	Seq. Length	Precision	F1
WordPiece	2.97	44.5	59.1	2.64	45.1	51.0	3.15	21.8	31.6	3.20	32.5	44.1
BPE	2.98	41.2	54.8	2.67	43.4	49.5	3.17	19.9	29.0	3.22	28.4	38.6
BPE′	2.60	53.8	66.2	2.47	50.8	54.7	2.93	24.6	34.8	2.86	37.4	48.0
Unigram	2.80	51.9	66.8	2.56	**58.1**	**64.3**	3.09	32.3	46.6	3.16	45.3	61.1
Unigram′	2.67	**56.7**	**70.9**	2.65	53.9	61.2	3.03	**33.6**	**48.1**	2.81	**54.5**	**69.2**

Table 3.1: Performance of the tokenisation algorithms across four morphological datasets, showing the average sequence length, precision and F1 score generated following the standard introduced by Creutz et al. (2004). Best results are shown in bold.

	Only Prefixes			Only Suffixes		
	Seq. Length	Precision	F1	Seq. Length	Precision	F1
WordPiece	2.56	26.7	32.2	2.56	48.4	53.4
BPE	2.54	33.5	40.2	2.62	44.0	49.4
BPE′	2.26	50.4	55.4	2.50	45.9	49.8
Unigram	2.51	53.4	63.6	2.53	**55.2**	**60.5**
Unigram′	2.48	**57.2**	**67.4**	2.75	46.4	53.9

Table 3.2: Performance of the tokenisation algorithms on subsets of the MorphoLex dataset with entries containing only prefixes and only suffixes. Best results are shown in bold.

Input	BPE	BPE′	Unigram	Unigram′
directional	_direction, al	direction, al	_direction, al	direction, al
unidirectional	_un, id, ire, ction, al	un, id, ire, ction, al	_un, i, direct, ional	uni, direction, al
electroneutral	_elect, r, one, ut, ral	electr, one, utr, al	_electron, eu, tral	electro, neutral
neurotransmitter	_neuro, trans, mit, ter	neuro, transmitter	_neuro, trans, mitt, er	neuro, transmitter
responsiveness	_respons, iveness	respons, iveness	_re, s, pon, s, ive, ness	r, e, sp, on, s, ive, ness
hyporesponsiveness	_hyp, ores, p, ons, iveness	hypo, respons, iveness	_hypo, res, pon, s, ive, ness	hypo, r, e, sp, on, s, ive, ness
hyperresponsiveness	_hyper, resp, ons, iveness	hyper, respons, iveness	_hyper, res, pon, s, ive, ness	hyper, r, e, sp, on, s, ive, ness
saturated	_sat, urated	sat, urated	_sat, ur, ated	saturated
unsaturated	_uns, atur, ated	un, sat, urated	_un, sa, tur, ated	un, saturated
equal	_equal	equal	_equal	equal
unequal	_un, equ, al	une, qual	_un, e, qual	un, equal
multiplayer	_multip, layer	multi, player	_multi, play, er	multi, player
nonmultiplayer	_non, m, ult, ip, layer	non, multi, player	_non, mul, ti, play, er	non, multi, player
overpriced	_over, p, ric, ed	over, pr, iced	_over, p, ric, ed	over, price, d
accessible	_accessible	accessible	_accessible	accessible
unaccessible	_un, ac, cess, ible	un, accessible	_un, ac, ces, s, ible	un, accessible
unicycle	_un, icy, cle	un, icy, cle	_un, i, cycle	uni, cycle

Table 3.3: Example tokenisations of the default and modified BPE and Unigram algorithms, with inputs taken from the following papers: Church (2020), Nayak et al. (2020), Hofmann et al. (2020a) and Schick and Schütze (2020).

our modified algorithms are able to generate improved tokenisations for complex words. For example, whereas the default Unigram algorithm tokenises "unicycle" into "_un" "i" "cycle", which is misleading as the string "un" does not have its typical semantic role, our modified Unigram algorithm tokenises it more meaningfully into "uni" "cycle". Also, the modified algorithms explicitly create links between words containing prefixes and their bases. For the words "accessible" and "unaccessible", the modified algorithms tokenise the subword "accessible" identically in both cases. The default Unigram and BPE algorithms

Input	WordPiece	WordPiece′
directional	_direction, al	direction, al
unidirectional	_un, idi, rect, ional	uni, direction, al
electroneutral	_electron, e, ut, ral	electron, eu, tra, l
neurotransmitter	_ne, uro, tr, ans, mit, ter	neuro, transmit, ter
responsiveness	_respons, iveness	respons, iveness
hyporesponsiveness	_hyp, ores, po, n, s, iveness	hypo, respons, iveness
hyperresponsiveness	_hyp, er, resp, ons, iveness	hyper, respons, iveness
saturated	_sat, ura, ted	sat, urated
unsaturated	_uns, at, ura, ted	un, sat, urated
equal	_equal	equal
unequal	_un, equ, al	une, qual
multiplayer	_multip, la, yer	multi, player
nonmultiplayer	_non, m, ult, ipl, ayer	non, multi, player
overpriced	_over, pr, iced	over, price, d
accessible	_access, ible	accessible
unaccessible	_un, acc, ess, ible	una, c, cess, ible
unicycle	_un, icy, cle	uni, cycle

Table 3.4: Example tokenisations of the default and modified WordPiece algorithms, with inputs taken from the following papers: Church (2020), Nayak et al. (2020), Hofmann et al. (2020a) and Schick and Schütze (2020).

do correctly split the prefix "un", but the rest of the word is tokenised differently, which is problematic, and even if the tokenisation was equivalent, the inclusion of the space symbol means there would be no link between these forms (Section 3.2). We note that our modified algorithms are not immune to oversegmentation, with Unigram′ tokenising "responsiveness" into seven tokens, although this is arguably inevitable with a limited vocabulary size. In Table 3.4, we show the same qualitative analysis between the default and modified WordPiece algorithms, finding parallels with default and modified BPE.

We investigate the vocabularies of the default and modified algorithms, shown in Table 3.5. We remove the tokens "[CLS]", "[SEP]", and "[UNK]" from the vocabularies. For the default algorithms, we also remove tokens that are duplicates apart from prepended space symbols, and we find that there is significant vocabulary degeneracy (8.7% and 9.1% for BPE and Unigram, respectively). We also find that a large percentage of the vocabulary is transferred over from the default to the modified algorithm (90.0% and 90.1% for BPE and Unigram, respectively). Additionally, we see that all of the algorithms have a similar number of prefixes in their vocabularies, which suggests the tokenisation algorithm plays an important role, as performance differences on handling prefixes are large (Table 3.2) despite similar vocabularies. This is supported by work by Hofmann et al. (2021), who find that employing a fixed vocabulary in a morphologically correct way leads to performance improvements. We also see, however, that Unigram′ has fewer suffixes in its vocabulary than default Unigram, which reflects the performance difference seen in Table 3.2.

	Vocab Size	Unique Elements	#Prefixes	#Suffixes
WordPiece	15 123	-	107	184
BPE	14 613	1 459	114	182
BPE′	15 997	2 843	123	192
Unigram	14 544	1 443	123	201
Unigram′	15 997	2 896	116	147

Table 3.5: Vocabularies of the models, showing size, number of unique elements, and numbers of prefixes and suffixes.

We note that an interesting result of our modifications is an improvement at word segmentation. As an example, the outputs of the default and modified Unigram algorithms when passed the concatenated sentence "thisisasentencethatneedstobesegmented" are:

Unigram _this, isa, s, ent, ence, that, ne, ed, s, to, be, s, eg, ment, ed

Unigram′ this, is, a, sentence, that, needs, to, be, segment, e, d

3.5 Extrinsic Evaluation: Pretrain-Finetune

Given the improved intrinsic performance of our algorithms, we wish to evaluate how this impacts the extrinsic performance of NLP models, both in general, and in particular on tasks involving complex words. As in Section 3.4, we train the default and modified BPE and Unigram algorithms on 1 million sentences from English Wikipedia, with a fixed vocabulary size of 16 000, but we also implement a variant of our modified algorithm that removes spaces as a post-processing step. The reasoning behind this is that it reduces the sequence length significantly with minimal information loss, and more closely mirrors existing models which have no explicit space information. Example tokenisations for the Unigram algorithms given the input "This is an input sentence." are:

Unigram _This, _is, _an, _input, _sentence, .

Unigram′ This, _, is, _, an, _, input, _, sentence, .

Unigram′ no spaces This, is, an, input, sentence, .

For each of the tokenisers, we pretrain RoBERTa (base) on the full text of English Wikipedia, and then finetune on downstream tasks, keeping all hyperparameters fixed, changing only the tokenisation algorithm used. For evaluation of the models in a general domain, we use the GLUE benchmark (Wang et al., 2018), excluding WNLI. For evaluation in specifically handling complex words, we use the two Superbizarre topicality tasks

(Hofmann et al., 2021), which require the binary classification of derivationally complex English words[7].

Over the whole of the English Wikipedia data, the sequence lengths for each of the tokenisation approaches are:

BPE 3.72e+09

BPE′ 5.88e+09

BPE′ no spaces 3.61e+09

Unigram 3.68e+09

Unigram′ 5.94e+09

Unigram′ no spaces 3.67e+09

As in the evaluation in Table 3.1, the modified models without spaces give shorter sequences than their default counterparts, with BPE′ without spaces giving the shortest mean sequence length. The difference in sequence lengths of the models means a difference in number of updates per epoch during pretraining. Hence, fixing the number of updates (and thus training time) will advantage models with shorter sequence lengths, especially disadvantaging the models that include spaces. Because of this, we perform two evaluations: one fixing the number of pretraining updates, and one fixing the number of pretraining epochs[8].

3.5.1 Results

Due to computational constraints, we only ran pretraining once for each model. For finetuning, we ran each experiment with 10 different seeds, reporting the mean development result and standard deviation. Results are shown in Table 3.6 and Table 3.7 for fixed updates and fixed epochs, respectively. Full training procedure is given in Appendix A.1.

On the Superbizarre datasets, we can see that Unigram outperforms BPE, with Unigram′ no spaces performing significantly better than all other models using a Welch's t-test ($p < 0.05$), see Appendix A.3. Note that DelBERT (Hofmann et al., 2021), a model which is passed the input segmented by a morphological algorithm, achieves 73.1 on the Arxiv dev set and 72.3 on the Arxiv test set, both worse than our (unsupervised) model, although DelBERT outperforms our best models on the Reddit task, achieving 69.6 and 70.1 on the dev and test sets, respectively.

On the mean GLUE benchmark, the modified models without spaces perform as well or better than their default counterparts, with Unigram′ performing the best when both

[7]We do not consider the Superbizarre sentiment task due to a higher proportion of uninformative words.

[8]In finetuning, the number of updates and epochs is equivalent for all models as one example is processed at a time. In pretraining, we follow the standard implementation of RoBERTa by taking contiguous sentences from the training data up to the max sequence length.

	Epochs	GLUE	SB Reddit		SB Arxiv	
			Dev	Test	Dev	Test
DelBERT (supervised)	-	-	69.6	70.1	73.1	72.3
BPE	27	81.6	66.8	66.6	71.1	70.2
BPE′	16	79.2	66.6	66.2	70.3	69.3
BPE′ no spaces	28	81.7	67.2	66.9	70.9	70.0
Unigram	27	81.5	68.0	67.8	72.2	71.4
Unigram′	16	78.4	68.2	68.2	72.5	71.6
Unigram′ no spaces	27	81.9	**68.8**	**68.8**	**73.0**	**72.3**

Table 3.6: Finetuning results after pretraining for 100,000 updates. Shown are mean results across 10 seeds. Results that are significantly better than all others using a Welch's t-test ($p < 0.05$) are shown in bold. More detailed results are given in Appendix A.2. We include DelBERT (Hofmann et al., 2021) as a supervised baseline, where the models are passed a morphological parse of the input.

	Updates	GLUE	SB Reddit		SB Arxiv	
			Dev	Test	Dev	Test
DelBERT (supervised)	-	-	69.6	70.1	73.1	72.3
BPE	109,761	81.5	67.1	66.8	71.0	70.1
BPE′	177,845	79.5	66.8	66.5	70.5	69.8
BPE′ no spaces	106,485	81.5	67.1	67.1	70.8	70.1
Unigram	108,606	81.6	67.9	67.9	72.2	71.6
Unigram′	179,909	79.1	68.3	68.3	72.5	71.8
Unigram′ no spaces	108,441	81.8	**68.8**	**69.0**	**73.2**	**72.5**

Table 3.7: Finetuning results after pretraining for 30 epochs. Shown are mean results across 10 seeds. Results that are significantly better than all others using a Welch's t-test ($p < 0.05$) are shown in bold. More detailed results are given in Appendix A.2. We include DelBERT (Hofmann et al., 2021) as a supervised baseline, where the models are passed a morphological parse of the input.

updates and epochs are fixed. However, this result is not statistically significant (see Appendix A.3), and over the individual GLUE tasks the best performing models vary, with high variances across seeds on some tasks due to the small dataset sizes (see Appendix A.2). Since the GLUE tasks do not rely on handling complex words, a significant performance difference is probably not expected, but we see no drop in performance with the modified algorithms.

The modified models that include spaces perform poorly on the GLUE benchmark, even when the number of epochs is fixed rather than updates, meaning they are trained for ∼65% more updates than the modified models without spaces. This suggests that this method of including spaces as additional tokens is suboptimal for general language tasks, though interestingly Unigram′ with spaces is the second best performing model across all

Superbizarre datasets. The tokenisers themselves perform splitting on spaces as a first step, so additionally including spaces may be simply passing noise to the model for the masked language modelling task, especially due to the high frequency of spaces. This means the pretraining loss decreases rapidly due to space prediction, but plateaus earlier (see Figure A.1). Due to the much greater sequence lengths, the models that include spaces would potentially also discard more examples that are too long during finetuning, which could lead to worse results, though not in the case of the GLUE benchmark.

3.6 Related Work

There are previous works that have performed controlled extrinsic comparisons of existing subword tokenisation algorithms (BPE, Unigram, and WordPiece), and have provided results which we relate here to our own findings. Gallé (2019) investigates various compression algorithms for tokenisation, including BPE, and finds an inverse link between mean tokens per sentence and translation quality, hypothesising that the compression capability of BPE leads to its effectiveness in NLP tasks. In our experiments we find that Unigram$'$ outperforms BPE$'$ on the complex words tasks, and there to be no significant difference between them on the general language understanding (GLUE) tasks. This is despite Unigram$'$ having a longer sequence length, suggesting this factor is not wholly indicative of model performance. However, if we look at the results for fixed pretraining updates, we do see a slight negative correlation between sequence length and performance on the Superbizarre datasets, and a very strong negative correlation on the GLUE benchmark[9], though this is skewed by the models including spaces performing very poorly. Intrinsically, we see a correlation (albeit weak) between sequence length and morphological correctness (Section 3.4). Bostrom and Durrett (2020) compare Unigram and BPE, finding that Unigram generates more morphologically correct tokenisations and gives improved downstream task performance. Whilst we saw similar improvements in intrinsic performance, we were unable to replicate the performance difference on MNLI that they found, finding no significant difference in performance (see Appendix A.2). We did not perform evaluation on the other two English datasets they used. Hofmann et al. (2022) corroborate these intrinsic results, additionally finding the morphological quality of WordPiece to lie in between that of BPE and Unigram, reflecting our own findings (Section 3.4). Wei et al. (2021) perform comparison between byte-level BPE and byte-level Unigram, finding BPE to perform better than Unigram across seven languages on the XNLI dataset, which is contrary to our findings and those of Bostrom and Durrett (2020) and Hofmann et al. (2022).

There have also been some recent attempts to develop improved subword tokenisation methods. Hofmann et al. (2021) introduce DelBERT, which takes input words tokenised

[9]Pearson correlations between -0.157 and -0.224 for the Superbizarre datasets, and -0.985 for GLUE.

according to gold standard morphological references, with an unchanged vocabulary. They find that this improves performance on their Superbizarre datasets (Section 3.5). Hofmann et al. (2022) also introduce FLOTA (Few Longest Token Approximation), which improves the performance of BERT, GPT-2, and XLNET at classifying ArXiv papers into their subareas from the title. Yehezkel and Pinter (2023) introduce a context-aware tokeniser, SaGe, which they find improves performance over BPE on GLUE tasks, the Turkish subset of XLNI, and NER in both Turkish and English. There are also alternative subword tokenisation algorithms which have a history of use in machine translation tasks, including Morfessor (Creutz and Lagus, 2002) and its successors (Virpioja et al. 2013, Grönroos et al. 2020), and Dynamic Programming Encoding (DPE) (He et al., 2020). (See Mielke et al. 2021 for a more extensive review.)

For all of these approaches, spaces still occur as the first character of start-of-word tokens, and we believe this hinders performance: our alternative treatment of spaces could be combined with these algorithms, and the impact on performance investigated.

Finally, we note that Wei et al. (2021) experiment with different methods of handling spaces within their byte-level BPE algorithm which appear similar to those implemented here, although they find these alternatives perform worse than the default on XNLI. They do not release code for their experiments so unfortunately we are unable to make a controlled comparison.

3.7 Conclusion and Future Work

We hypothesise that problems with current tokenisation algorithms arise from allowing tokens to include spaces, and thus experiment with an alternative tokenisation approach where spaces are always treated as individual tokens. We find that this leads to improved performance on NLP tasks involving complex words, whilst having no detrimental effect on performance in general natural language understanding tasks. Whilst our work focuses on BPE and Unigram, our modifications can be applied to any existing subword tokenisation algorithm, including WordPiece, and hence to any transformer-based model. Also, although our experiments have only been in English, the algorithms used are unsupervised and language-independent and our results should extend to other languages.

Our best-performing models use lossy tokenisation, removing the space tokens as a post-processing step. For our experiments, we find this to give a performance improvement due to much reduced sequence lengths and removal of redundant information. Nevertheless, this may not be ideal for all tasks. We did not perform evaluation on sequence-to-sequence tasks, and indeed the subword tokenisation algorithms discussed here were introduced in the field of NMT, where space information needs to be generated in the output. Future work could thus look at alternative methods for including space information that maintain the performance gains seen here whilst keeping tokenisation lossless.

Chapter 4

Publication II: Word Boundary Information isn't Useful for Encoder Language Models

Edward Gow-Smith[1], Dylan Phelps[1], Harish Tayyar Madabushi[2], Carolina Scarton[1], Aline Villavicencio[1,3]

[1]Department of Computer Science, University of Sheffield

[2]Department of Computer Science, University of Bath

[3]Institute of Data Science and Artificial Intelligence, University of Exeter

Abstract

All existing transformer-based approaches to NLP using subword tokenisation algorithms encode whitespace (word boundary information) through the use of special space symbols (such as ## or _) forming part of tokens. These symbols have been shown to a) lead to reduced morphological validity of tokenisations, and b) give substantial vocabulary redundancy. As such, removing these symbols has been shown to have a beneficial effect on the processing of morphologically complex words for transformer encoders in the pretrain-finetune paradigm. In this work, we explore whether word boundary information is at all useful to such models. In particular, we train transformer encoders across four different training scales, and investigate several alternative approaches to including word boundary information, evaluating on two languages (English and Finnish) with a range of tasks across different domains and problem set-ups: sentence classification datasets, NER (for token-level classification), and two classification datasets involving complex words (Superbizarre and

FLOTA). Overall, through an extensive experimental setup that includes the pre-training of 35 models, we find no substantial improvements from our alternative approaches, suggesting that modifying tokenisers to remove word boundary information isn't leading to a loss of useful information for decoder language models.

4.1 Introduction

Transformer (Vaswani et al., 2017) pretrained language models for NLP, such as BERT (Devlin et al., 2019) and the GPT family (Brown et al., 2020; OpenAI, 2023), typically use subword tokenisation algorithms, such as WordPiece (Schuster and Nakajima, 2012), to process text. Previous work (Church, 2020; Park et al., 2021) has shown that such methods have limited alignment with word morphology, resulting in worsened downstream performance for various tasks (Klein and Tsarfaty, 2020; Bostrom and Durrett, 2020; Pinter et al., 2020). In fact, it has been shown that the morphological validity of tokenisation can be improved by treating whitespace as individual tokens, and that downstream performance is improved by removing all whitespace markers (and hence word boundary (WB) information) from the tokenisers (Gow-Smith et al., 2022). However, the full impact of this modification on downstream performance is unknown, and the question of whether WB information is at all useful to models is as yet unanswered. In this work, we first perform a morphological evaluation of WordPiece and WordPiece′, a version which has been modified to have no WB information. We find that WordPiece′ significantly improves the alignment with morphological gold standard references. Then, we evaluate WordPiece and WordPiece′ as tokenisers on downstream tasks. We also introduce models which modify WordPiece′ by including WB information in various ways—either explicitly through the input or implicitly through the pretraining objective. Much interest recently has been in the scaling laws of language models (Kaplan et al., 2020; Hoffmann et al., 2022), and a direction towards training larger models. On the other hand, there has been recent work investigating sample-efficient pretraining on datasets of a developmentally plausible size (Warstadt et al., 2023). In companion to such work, we train our models across four training scales, from approximately 6M params and 250M tokens at the lowest scale to approximately 370M params and 23B tokens at the highest scale.

Across these scales we pretrain all of our models and evaluate in English on four

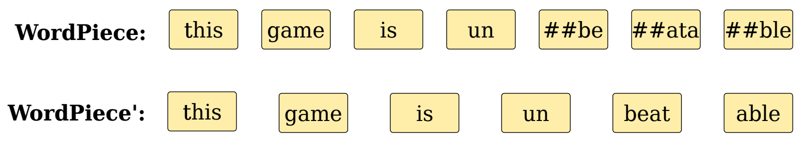


Figure 4.1: Tokenisations generated by WordPiece and WordPiece′ for the input sequence "this game is unbeatable".

downstream tasks (comprising 16 datasets): Named Entity Recognition (NER), GLUE, and two tasks involving classifying complex words. We additionally train and evaluate in Finnish across two tasks: NER and Sequence Classification.

The findings of our work are as follows: (1) we show that modifying WordPiece to remove WB information (giving WordPiece′) substantially improves the morphological validity of the resulting tokenisations across English and Finnish; (2) across four training scales, we find that WordPiece′ outperforms WordPiece on downstream tasks involving complex words, and gives better performance across most datasets at the lower training scales; (3) we find that none of our methods for including WB information into models, whether implicit or explicit, or through finetuning alone, significantly affects the performance across four downstream tasks and three training scales. Our results indicate that word boundary information isn't providing additional useful information to encoder language models, with morphemes being the most important subunit.

	WordPiece	WordPiece′
hyporesponsiveness	hyp ##ores ##po ##n ##s ##iveness	hypo respons iveness
nonmultiplayer	non ##m ##ult ##ipl ##ayer	non multi player
overpriced	over ##pr ##iced	over price d
unicycle	un ##icy ##cle	uni cycle
undesirable	und ##es ##ira ##ble	und es ira ble

Table 4.1: Some examples of the tokenisations from WordPiece and WordPiece′.

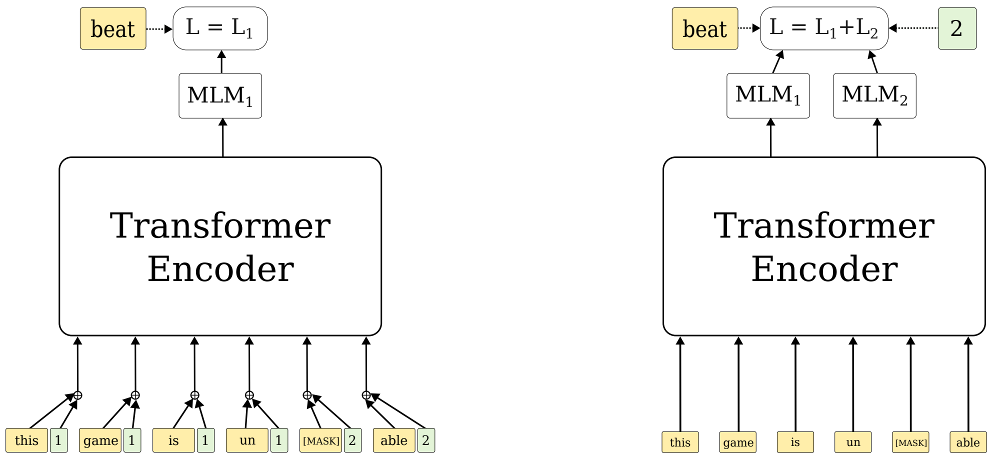
4.2 Tokenisers

One particular design choice of subword tokenisers used by transformer models is the addition of prefixes such as "_" and "##" in order to encode space information, hence representing word boundaries in languages with spaces between words. Previous work (Gow-Smith et al., 2022) has investigated the impact of these prefixes, showing they lead to less morphologically valid tokenisations, and also to a reduced efficiency, since the dual representation of subwords (e.g. "beat" and "_beat") gives a vocabulary redundancy (of approximately 9%). As such, removing these tokens for Unigram (Kudo, 2018) and BPE (Sennrich et al., 2016; Gage, 1994) has been shown to have a beneficial effect on downstream performance for complex word tasks, whilst retaining equivalent performance in general natural language understanding tasks. We refer readers to Gow-Smith et al. (2022) for a full analysis, but here we focus on WordPiece′ – WordPiece modified such that WB information is removed. We train this model and the default on 1 million sentences from Wikipedia for two languages (English and Finnish). We show an example of the tokenisations generated by this compared to the default for English in Figure 4.1. We perform a morphological evaluation of WordPiece′ compared to WordPiece across the two

	English			Finnish		
	Len	Precis.	F1	Len	Precis.	F1
WordPiece	3.29	24.8	33.8	3.21	28.3	38.9
WordPiece′	2.75	**42.6**	**52.7**	2.86	**34.7**	**45.0**

Table 4.2: Performance of WordPiece and WordPiece′ across English and Finnish, showing the average sequence length, precision and F1 score generated following the standard introduced by Creutz et al. (2004).

languages, shown in Table 4.2. For English, we use four datasets (LADEC (Gagné et al., 2019), MorphoLex (Sánchez-Gutiérrez et al., 2018), MorphyNet (Batsuren et al., 2021), DagoBERT (Hofmann et al., 2020a)), and we average across all four (full breakdown in Appendix B.1). For Finnish, we use the subset of MorphyNet. Here, we follow the evaluation standard from Creutz et al. (2004), reporting precision and F1. Averaging across English and Finnish, we see that WordPiece′ gives 14% shorter sequences, 46% higher precision, and 34% higher F1 compared to WordPiece. We also show examples of English tokenisations for WordPiece and WordPiece′ in Table 4.1. In general, we can see that WordPiece generates more meaningful tokenisations, but sometimes they are still of limited morphological validity, as for "undesirable" where the prefix is incorrectly split and the base form of the word is lost: we note that WordPiece (like BPE) is a greedy algorithm, meaning it has a tendency to overlengthen the initial token of a word.



(a) Explicit model, where word boundary embeddings are passed in the input.

(b) Implicit model, with an additional MLM head for predicting word boundaries.

Figure 4.2: Network diagrams for the modified transformer architectures trained in this work.

4.3 Models

The sequences generated by WordPiece′ have *no word boundary information*, which means some information is lost when using it to encode sequences. We aim to answer the question of whether such information is at all useful to transformer encoders—-i.e. can it be incorporated in an alternative way to improve performance? We investigate transformer encoders pretrained using the masked language modelling (MLM) task, and then finetuned on downstream tasks (pretrain-finetune paradigm). We then look to include WB information in two ways, either directly as input (both in pretraining and finetuning), or through a modification of the pretraining task.

4.3.1 Explicit Model

One approach is to include WB information *explicitly* through the input. Naively, we could add WB tokens in the input sequence, shown in Figure 4.3. However, this is rather inefficient as it leads to much longer sequences and has been shown to lead to reduced downstream task performance, even when the number of epochs (rather than steps) is matched (Gow-Smith et al., 2022). Nevertheless, we implement this as a baseline. An alternative, and significantly more efficient, way to include this information is to add "word boundary embeddings" to the input, added element-wise with the token embeddings and standard position embeddings, shown in Figure 4.2a. These embeddings are equivalent to the standard position embeddings in being randomly-initialised and then learned through training.

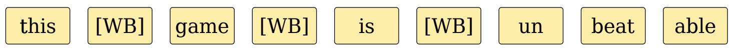


Figure 4.3: WordPiece′ with word boundary tokens.

We experiment with three methods for indexing the WB embeddings: *binary index*, *word index*, and *subword index*, shown in Figure 4.4. The *word index* is the position of the word the corresponding token belongs to, whereas the *subword index* is the position within the word. These are chosen to align with how the standard position indices work within transformer architectures, but the *binary index* aligns with how standard WordPiece processes word-initial and word-internal tokens, having a value of 1 if a token appears at the start of the word, and a value of 2 otherwise. The *binary index* is also more parameter-efficient, since it only requires an embedding dimension of 2. In fact, for our experiments the *subword index* gives the most new parameters, since even in our English pretraining corpora (Wikipedia and C4) we encounter large chunks of (e.g. Chinese) text with no whitespace, requiring a high embedding dimension.[1]

[1]We set the embedding dimension at 512, which covers all text encountered for all scales. For the word index, the embedding dimension is set at the max sequence length (256).

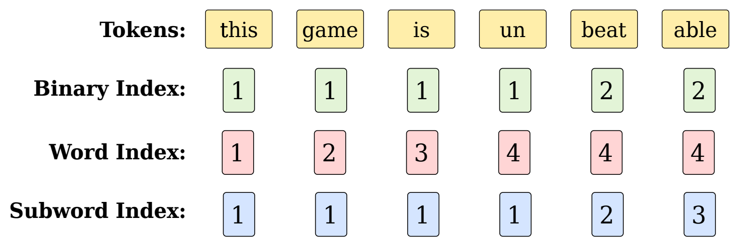


Figure 4.4: Three alternative indexing methods for the word boundary embeddings.

	# Articles (M)		Params (M)	Batch Size	# GPUs	Steps (k)
	Eng.	Fin.				
V Low	0.1	0.1	5.8	1024	1	25
Low	0.5	2	21.2	512	1	50
High	6.5	10	98.2	256	1	400
V High	40	-	370.4	128	4	400

Table 4.3: The four training scales we use to evaluate our models.

Finetuning

Alongside including WB information at pretraining, we also experiment with pretraining using the default MLM task and architecture, and then passing the WB information during finetuning only, either with binary index WB embeddings, or WB tokens.

4.3.2 Implicit Model

One possible drawback of the explicit approach is the reduced difficulty of the MLM task: passing WB information in the input allows the model to utilise this directly for predicting the masked token, rather than inferring it from context alone. Thus, as an alternative, we modify the architecture with an additional MLM head such that the model has to predict the word boundaries from the input, which we state as *implicitly* using WB information through backpropagation. We show the architecture in Figure 4.2b. In this set-up, we simply sum the losses from the two MLM heads to give the overall loss.[2]

	GLUE				NER				Superbizarre				FLOTA			
	V Low	Low	High	V High	V Low	Low	High	V High	V Low	Low	High	V High	V Low	Low	High	V High
WordPiece	54.7	67.7	77.9	83.1	54.3	**68.9**	**76.9**	81.5	65.7	66.2	67.3	68.6	19.5	31.2	50.4	55.0
WordPiece′	**56.2**	**69.8**	78.0	83.7	53.6	68.0	75.7	81.5	**66.9**	**67.6**	**68.4**	**69.5**	**23.6**	**43.1**	**52.3**	55.2

Table 4.4: English results across the four tasks and training scales for WordPiece and WordPiece′, with standard deviations in parentheses. Results in bold are those better by more than the combined standard deviation ranges.

[2]In preliminary experiments we tried weighting the two losses, but no increase in performance was observed.

	GLUE			NER			Superbizarre			FLOTA		
	V Low	Low	High	V Low	Low	High	V Low	Low	High	V Low	Low	High
WordPiece′	56.2	69.8	78.0	53.6	68.0	75.7	66.9	67.6	68.4	23.6	43.1	52.3
WordPiece′ implicit	56.2	69.0	77.8	55.3	69.2	75.6	66.9	67.6	68.3	23.5	45.1	51.8
WordPiece′ explicit binary	55.7	70.1	78.4	54.4	68.2	75.3	66.9	67.6	68.2	24.5	44.5	51.8
WordPiece′ explicit word	57.2	69.2	78.8	54.9	68.4	75.4	66.8	67.6	68.4	22.3	43.2	51.0
WordPiece′ explicit subword	55.6	70.3	78.1	55.0	68.1	75.4	67.0	67.7	68.2	24.3	38.2	51.8
WordPiece′ explicit WB tokens	55.3	68.7	77.5	52.4	67.6	74.1	66.6	67.5	68.3	23.3	43.5	52.3
WordPiece′ explicit f/t WB tokens	55.1	69.8	76.7	53.6	68.3	75.7	-	-	-	23.4	43.7	52.5
WordPiece′ explicit f/t binary	56.2	69.9	77.8	53.6	68.6	75.4	66.9	67.5	68.1	23.4	43.6	52.6

Table 4.5: English results across the four tasks and three training scales for WordPiece′ and the modified architectures which include word boundary information, with standard deviations in parentheses.

4.4 Experiments

	NER			SeqClass		
	V Low	Low	High	V Low	Low	High
WordPiece	72.2	84.2	89.9	73.1	78.7	83.6
WordPiece′	73.0	85.0	89.8	73.0	79.0	**84.1**

Table 4.6: Finnish results across the three tasks and training scales for WordPiece and WordPiece′, with standard deviations in parentheses. Results in bold are those better by more than the combined standard deviation ranges.

We evaluate the two tokenisers (WordPiece and WordPiece′) and our seven explicit and implicit models in the pretrain-finetune paradigm for English and Finnish across three training scales (V Low, Low, High), with an additional scale (V High) for English WordPiece and WordPiece′ (unmodified)—due to the high computational cost of training, we don't train the other models at this scale. Across these scales we vary the number of parameters, batch size, and training steps, shown in Table 4.3, with further detail in Appendix B.2. The first three set-ups for English, and the first two for Finnish, take the training data from Wikipedia, whilst the remaining take data from C4 (Raffel et al., 2020). The number of parameters is altered by adjusting the layers, attention heads, and embedding dimension, with full information in Appendix B.2. We train our models in the manner of RoBERTa (Liu et al., 2019) (in comparison to BERT, this involves no next sentence prediction, and dynamic masking is performed), and we mask 15% of tokens. Across all set-ups, we linearly warmup the learning rate to a maximum value of 1e-4, and then linearly decay to 0. We use a sequence length of 256. All training is performed on A100 or H100 GPUs. Training and validation losses for these models are given in Appendix B.4.

For these models, we run an evaluation on four downstream tasks. The first two tasks focus on natural language understanding across a broad range of domains:

GLUE We evaluate on 8 GLUE (Wang et al., 2018) tasks (excluding the 9th task of WNLI (Levesque et al., 2012), following previous work, due to its adversarial nature). These tasks all involve sequence classification, and cover a wide range of domains and setups: two single-sentence tasks, three similarity and paraphrase tasks, and three inference tasks. We report the average metric across all tasks.

NER We evaluate on three NER datasets from different domains: the English portion of the CoNLL-2003 NER dataset (Tjong Kim Sang and De Meulder, 2003), consisting of sentences taken from the Reuters news corpus (Rose et al., 2002); the NCBI Disease corpus (Dogan et al., 2014), consisting of PubMed abstracts; and the WNUT2017 Shared Task (Derczynski et al., 2017), with training data taken from Twitter, and test data from YouTube.

The final two tasks specifically involve morphologically complex words, where we expect more morphologically valid tokenisations to result in improved performance:

Superbizarre The Superbizarre datasets (Hofmann et al., 2020b) involve the binary classification of standalone complex words. We take the two topicality datasets: Arxiv, which involves predicting whether a word comes from the Physics or Computer Science subject areas; Reddit, which involves predicting whether a word comes from an entertainment or discussion subreddit. We report the average macro F1 across the two datasets.

FLOTA The datasets introduced alongside the FLOTA tokenisation method (Hofmann et al., 2022) involve classifying the title of an Arxiv paper into one of 20 subareas for three subject areas (Computer Science, Maths, Physics). We take the small version of the dataset, with a train set of 2 000 titles per subject area. We report the average macro F1 across the three datasets.

4.4.1 Finnish

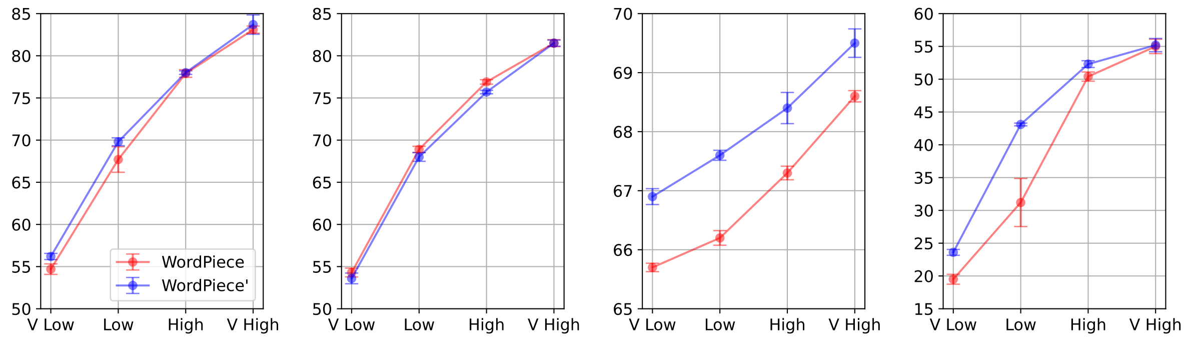
In addition to our experiments on English, we train models on Finnish, to see whether our results are transferable to a morphologically complex language—one could hypothesise that with greater morphological complexity, word boundary information would be more helpful in disambiguation. We run our experiments on Finnish for WordPiece and WordPiece′ across three training scales, and evaluate on two downstream tasks:

NER We evaluate on the FiNER dataset (Ruokolainen et al., 2020), consisting of news articles annotated with six entity classes, reporting macro F1.

Sequence Classification We look at two sequence classification datasets: the Eduskunta dataset,[3] consisting of ministers' answers to questions from MPs, labelled with the relevant ministry; the FinnSentiment dataset (Lindén et al., 2023), consisting of sentences from social media labelled with their polarity. We report the accuracy over these two datasets.

4.4.2 Finetuning Procedure

An overview of all datasets is given in Appendix B.3. We finetune on each dataset by updating all parameters, with the following hyperparameters: batch size 32, max sequence length 128, learning rate of 2e-5, warm-up for 5% of steps. We evaluate every epoch on the dev set, taking the best-performing epoch. We train five seeds for every model and report the average metric across these. We also remove outliers which lie more than two standard deviations from the mean, or when very low scores suggest the model failed to train.[4] For the English NER and Complex Words Datasets, and all Finnish datasets, we train for 20 epochs, but for GLUE we limit it to 10 epochs per dataset due to the relatively high training time.



(a) GLUE (b) NER (c) Superbizarre (d) FLOTA

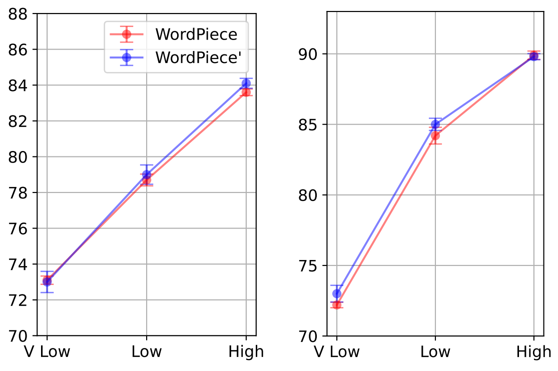
Figure 4.5: English results for WordPiece and WordPiece′ across four training scales and four tasks.

4.5 Results

We report our full results across all individual datasets for all models in Appendix B.5. Here, we look at the overall metrics from the four tasks across the training scales, and

[3] https://github.com/aajanki/eduskunta-vkk

[4] This occurs for the following. High: one seed of WordPiece′ FLOTA CS (score of 7), one seed of WordPiece′ FLOTA Maths (score of 11), one seed of WordPiece′ f/t WB tokens (score of 3); V High: one seed of WordPiece′ WB tokens CoLA (score of 0), two seeds of WordPiece CoLA (scores of 0 and 8), one seed of WordPiece′ STS-B (score of 2), one seed of WordPiece FLOTA CS (score of 4), one seed of WordPiece FLOTA Maths (score of 3).



(a) SEQ (b) NER

Figure 4.6: Finnish results for WordPiece and WordPiece′ across three training scales and two tasks.

present our main findings. We note that the plots produced (Figures 4.5, 4.6 and B.4 to B.7) are approximately logarithmic in training scale, and we reproduce them using a scale factor on the x-axis in Appendix B.7.

Firstly, we compare WordPiece and WordPiece′ in Table 4.4 and Figure 4.5 (standard deviations in Appendix B.6). On GLUE, WordPiece′ performs better than WordPiece across all scales, with a bigger performance difference at the lower scales (+1.5 and +2.1 for the V Low and Low training scales, respectively). We note that at the higher scales, the differences are within two standard deviations of the baseline, so these results are consistent with those by Gow-Smith et al. (2022). For NER, on the other hand, we find that WordPiece′ performs worse than WordPiece across all training scales except V High, where they perform equivalently. Looking at the individual dataset performances (Table B.6) we see that the worse performance on WNUT2017 (-2.5 average decrease across scales) accounts for the worse overall NER performance, with the other two datasets giving similar results (apart from at the V Low scale, where WordPiece′ performs substantially better on them). The WNUT2017 dataset involves tagging "unusual, previously-unseen entities", which means morphological composition cannot be leveraged—we hypothesise that the improved ability of WordPiece′ to do this is the cause of the performance drop, due to the futility of composing the meaning of novel surface forms from subunits. Overall, further work is needed to determine the affect of our tokeniser modifications on NER performance. Our results on Finnish (Table 4.6 and Figure 4.6) show no substantial performance difference between WordPiece and WordPiece′ across the sequence classification and NER tasks, apart from for the High training scale on sequence classification, where WordPiece′ outperforms WordPiece.

For the complex word tasks, WordPiece′ substantially outperforms WordPiece: averaging across the training scales, we get 1.1 average increase for Superbizarre, and 4.5 average increase for FLOTA. The relative performance difference is biggest for Superbizarre: at the V Low scale, we would require approximately 20 times the training scale for WordPiece to match WordPiece′ (Figure B.8c). In general, we find the performance differences to decrease as the training scale increases, as expected,[5] however this effect seems much smaller for Superbizarre, which still has a large performance difference at the V High training scale (+0.9).

Next, we look at the models that attempt to use WB information, with results in Table 4.5.

Comparing WordPiece′ and the *implicit* variant (shown also in Figure B.4), we find that adding the extra loss term gives mixed results across the four tasks and training scales. We do however see that at the V Low and Low training scales, the implicit model improves performance for NER (+1.7 and +1.2, respectively). Since this prediction task is very similar to the finetuning task of token classification, this may explain the effect on performance. The additional MLM head increases the total loss, but when we look at the evaluation accuracies for the two MLM heads, we see that the default MLM head has very similar accuracies to the WordPiece′ baseline (Appendix B.4). We also note that for the Very Low training scale, there is a 3.5% (relative) improvement in default MLM accuracy, which could be contributing to the performance improvement—in a low resource scenario (both compute and data), the extra prediction task may help to leverage additional information, and linguistic information is likely to provide more benefit than a purely data-driven approach.

Next, we look at the *explicit* variants. Naively including the WB information through additional tokens leads to decreased performance across all tasks except for FLOTA, where there is no substantial performance difference (Figure B.5). Overall, these differences are small: around 1 for GLUE, 0.5-2 for NER, 0.1-0.3 for Superbizarre. This is despite a significantly lower MLM loss (approximately 60%: Figure B.3a) due to the high probability of WB tokens, and the fact that this model trains for around 40% fewer epochs (Appendix B.2). We next look at the three variants for WB embeddings (Figure B.6). Overall, none of these models consistently improve over WordPiece′, and the relative performance of the three indexing methods varies with training scale and task. The subword index model has the greatest number of additional parameters, which might explain why this model performs the best overall at the V Low scale. In this setting this model has 2.3% more parameters than the baseline, compared to 1.1% for the word index model, and 0.01% for the binary index model. The model achieves an average performance across the four tasks of 50.5, compared to 50.3 for the other two variants, and 50.1 for the baseline. However, at the Low training scale, this model actually performs worse than the other two variants (61.1 average compared to 62.6 and 62.1 for binary and word, respectively). At the High training scale they all perform equivalently (68.4 average). Since all three

[5]Improved morphological validity should matter less when the model capacity is greater, and when morphologically complex and rare words have been encountered more times during pretraining.

indexing methods are encoding equivalent information through trivial transformations, the performance equivalence is perhaps expected.

Finally, we look at two approaches to including WB information during finetuning only (Figure B.7)—with WB tokens or binary index WB embeddings. We find that neither of these approaches improve over the baseline, with the WB tokens approach performing overall slightly worse: averaged across all training scales and datasets, we get 57.5 for default WordPiece′, 57.5 for WordPiece′ f/t binary index, and 57.3 for WordPiece′ f/t WB tokens. This corroborates the results by Abdou et al. (2022), who find that adding position embeddings after pretraining without them does not lead to improved performance. On average, including the WB embeddings during finetuning decreases training stability (increased standard deviation across seeds).

4.6 Discussion

Overall, we find that *incorporating word boundary information in transformer encoders, either explicitly or implicitly, does not lead to substantial performance improvements*. This suggests that: a) modifying tokenisers such as WordPiece to remove space information does not result in the loss of useful information for these models, b) the default MLM task is sufficient for such models to pretrain effectively.

The pre-tokenisation step of splitting on whitespace prevents tokens from ever crossing word boundaries, which is perhaps a sufficient restriction. Our results indicate the importance of a morpheme compared to a word as the key feature which contributes to meaning.

For English, across all models and training scales, we only see a weak correlation between performance on NER and GLUE—if we compare the difference compared to WordPiece′ for the implicit and explicit models, we find a correlation with Pearson's $\rho = 0.332$. This might be a result of the unusual WNUT2017 task, and further work is needed to investigate the effect of our tokeniser modifications on NER.

The Superbizarre task is much less affected by model scaling than the other tasks we evaluate on, but much more affected by the choice of tokeniser. This suggests that morphologically valid tokenisation is vital for generating good representations of complex words in the absence of context. This task is also less likely to be dependent on spurious correlations (annotation artefacts) in the data.

All of our models at the High and V High training scales outperform the dev results reported by Hofmann et al. (2022) on the FLOTA ArXiv-S datasets using their tokenisation method. We hypothesise this is likely an effect of hyperparameters, e.g. we use a batch size of 32 rather than their 64, and we use a learning rate scheduler with warm-up, whereas they do not.

4.7 Related Work

This work aligns with other works that aim to improve the morphological validity of subword tokenisers: Westhelle et al. (2022) introduce Morphologically Informed Segmentation (MIS), a tokeniser based on Morfessor for Portuguese; Hofmann et al. (2022) introduce Few Longest Token Approximation (FLOTA), which preserves the morphology of complex words without necessarily keeping all the characters. Jimenez Gutierrez et al. (2023) introduce a tokeniser for the biomedical domain that is better aligned with morpheme segmentation, and then train their BioVocabBERT model using it. There has also been work looking at the impact of how subword tokens are marked, either with word-initial or word-final prefixes (Jacobs and Pinter, 2022).

There is previous work which has passed additional position indices to transformer models. Jia et al. (2021) introduce a model for neural text-to-speech called PnG BERT which uses word-position embeddings to provide alignment between phonemes and graphemes at the word level. In NLP, Bai et al. (2021) introduce Segatron, a model which modifies the Transformer-XL (Dai et al., 2019) with two additional position embeddings: a sentence index and a paragraph index. They also apply the same modifications to BERT, giving SegaBERT. They find that SegaBERT gives lower validation losses during pretraining, lower language modelling perplexities, and improves upon the GLUE score of BERT. Cheng et al. (2023) include POS tags as additional input embeddings during BERT pretraining, which they find to reduce performance on (Super)GLUE (Wang et al., 2019) and MSGS (Warstadt et al., 2020).

There has also been work which has modified the pretraining objective of transformer models. Yamaguchi et al. (2021) introduce various alternatives to MLM, and pre-train models using them, finding that default MLM is superior in the higher-parameter setting. There have been various works using linguistically-motivated pretraining objectives (Zhou et al., 2020; Levine et al., 2020), with the closest to our work being that by Cui et al. (2022), who find improved performance through simply adding additional MLM heads for linguistic tasks and summing their losses.

4.8 Conclusion

In this work we investigate whether word boundary information is useful for transformer encoders. In particular, we start with WordPiece′, a version of WordPiece modified to remove word boundary information, and show that it leads to more linguistically meaningful tokenisations, as well as improved performance on tasks involving morphologically complex words, whilst having no significant effect on performance for general domain tasks across English and Finnish. We also investigate modifications to the default model architecture which involve incorporating word boundary information, either explicitly (through the input), or implicitly (through the pretraining task), and through pretraining or fine-tuning alone. Across all models and training scales, we find that these modifications give no substantial improvements in performance, which suggests transformer encoders can perform well without word boundary information, either in the form of prefixes ("##" or "_"), word boundary tokens, word boundary embeddings, or through a modification to the pretraining task.

Chapter 5

Publication III: Multilingual Complex Word Classification From Internet Forums for Evaluating Subword Tokenisation

Edward Gow-Smith[1], Dylan Phelps[1], Carolina Scarton[1], Aline Villavicencio[1,2]

[1]Department of Computer Science, University of Sheffield

[2]Institute of Data Science and Artificial Intelligence, University of Exeter

Abstract

Despite significant interest in subword tokenisation and its effect on language models, there are limited datasets which evaluate tokeniser performance on morphologically complex words: an area which is expected to be most impacted by tokeniser quality. In particular, existing datasets are limited to English. To address this, we introduce mCWIF (multilingual classification of Complex Words from Internet Forums) for complex word classification in English, German, Turkish, and Finnish. We extract topically-relevant words of 10 or more characters from internet forums, and introduce two dataset sizes (1 000 and 10 000 training examples). We run finetuning experiments on mCWIF to assess the performance of WordPiece vs WordPiece′, a space-aware tokeniser that ameliorates issues with the space suffix "##" added to tokens. We find the biggest improvement of WordPiece′ for German, and the least for Turkish. We investigate potential factors leading to this result, and introduce two metrics (position diversity and tokeniser consistency) to evaluate

the impact of space-aware tokenisation across languages. We hypothesise that topically-relevant subunits in Turkish typically occur either a word-initial or non-word-initial position, but rarely both, which we suggest contributes to the similar performance of WordPiece and WordPiece′ in that language.

5.1 Introduction

Subword tokenisation is a widely-used approach in natural language processing to convert text into subunits (typically below the word level), which can then be vectorised (Mielke et al., 2021). The impact of the choice of tokeniser – such as BPE (Sennrich et al., 2016; Gage, 1994), WordPiece (Schuster and Nakajima, 2012), and Unigram (Kudo, 2018) – on the performance of language models is something which has gained attention. Intrinsic evaluation of these tokenisers has uncovered weaknesses when handling morphologically complex words, often producing tokenisations that do not align well with morphological boundaries (Church, 2020; Bostrom and Durrett, 2020; Hofmann et al., 2021; Arnett and Bergen, 2025; Nayak et al., 2020), and this has spawned alternative tokenisation algorithms attempting to ameliorate such issues (Hofmann et al., 2022, 2021; Gow-Smith et al., 2022). There have also been investigations into other intrinsic tokeniser properties, such as compression abilities (Gallé, 2019; Schmidt et al., 2024; Goldman et al., 2024; Uzan et al., 2024; Zouhar et al., 2023) or cognitive plausibility (Beinborn and Pinter, 2023).

Nevertheless, the link between intrinsic metrics and extrinsic performance (downstream evaluation) of subword tokenisers is an open question, and evaluation is typically performed on general-domain tasks (Schmidt et al., 2024; Goldman et al., 2024; Bostrom and Durrett, 2020; Gallé, 2019; Yehezkel and Pinter, 2023; Zouhar et al., 2023; Rust et al., 2021), mostly in English. One area of evaluation which previous work has shown to be impacted highly by the choice of tokeniser is the processing of complex words (Gow-Smith et al., 2022, 2024; Schick and Schütze, 2020). But, almost all existing datasets for complex word classification are in English (Schick and Schütze, 2020; Hofmann et al., 2022, 2021; Batsuren et al., 2024).

To ameliorate the English-only nature of existing evaluations, we introduce **mCWIF** (**m**ultilingual classification of **C**omplex **W**ords from **I**nternet **F**orums), a dataset consisting of complex words extracted from internet forums in English, German, Turkish, and Finnish. We extract long words (10 or more characters) with strong topicality signals, with the task being to classify each word by its forum of origin. We use mCWIF to run a multilingual evaluation of WordPiece and WordPiece′ (Gow-Smith et al., 2022), a variant which handles whitespaces by always treating them as individual tokens. Encoder-only models up to 370M parameters are pretrained with either tokeniser, and finetuned on our dataset. We find the biggest performance improvement for German, then English,

	Word (Translation)	Forum (Topic)
English	gravimagnetic	cosmoquest (Astronomy)
English	neuroendocrinology	phoenixrising (Chronic Illness)
German	graupapageienhenne (grey parrot hen)	vogelforen (Birds)
German	flugunfallforschung (aviation accident research)	flugzeug (Airplanes)
Turkish	kaldırabileceğiniz (you can lift)	bodyforum (Bodybuilding)
Turkish	defterdarlığının (of the finance office)	alomaliye (Accounting)
Finnish	ympäristövaikutusten (environmental impacts)	talo (Urban Planning)
Finnish	kohdunsuulla (at the cervix)	vau (Childbirth)

Table 5.1: Two examples per language from mCWIF Small, each from a different forum. The task is to classify which forum (and hence topic) a word belongs to.

then Finnish, with inconsistent performance differences for Turkish. We investigate possible reasons for this. By introducing two new metrics, we show that topically-relevant subunits in Turkish typically occur either at a word-initial or non-word-initial position, but rarely both, whereas German has the highest degree of position diversity, which we suggest contributes to these results.

5.2 Datasets

We develop multilingual datasets for complex word classification using naturalistic language. Given the fecundity of social media for novel morphologically complex words (Onyedum, 2012; Del Tredici and Fernández, 2018; Shahlee and Ahmad, 2022), we choose to use internet forums as our basis for collecting data: these have strong semantic signals from the topics they involve. This is motivated by the previous success in Reddit as a source of complex words (Hofmann et al., 2021), which is however almost exclusively in English (Ye et al., 2023; Koncar et al., 2021). In order to extract internet forums, we use Common Crawl as a starting point,[1] specifically the colossal, cleaned version provided by

[1] https://commoncrawl.org/

	Forum URL	Topic
English	avforums.com	Home Entertainment
	forums.phoenixrising.me	Chronic Illness
	forum.cosmoquest.org	Astronomy
	forum.hairsite.com	Hair Regeneration
	bikeforums.net	Cycling
German	vogelforen.de	Birds
	bauexpertenforum.de	Buildings
	flugzeugforum.de	Airplanes
	h0-modellbahnforum.de	Model Railways
	imkerforum.de	Beekeeping
Turkish	bodyforumtr.com	Bodybuilding
	keyfimuzik.net	Music
	forum.alomaliye.com	Accounting
	risaleforum.com	Islam
	forum.bordomavi.net	Trabzonspor (Football)
Finnish	forum.ylikerroin.com	Betting
	fillarifoorumi.fi	Cycling
	taloforum.fi	Urban Planning
	forum.vau.fi	Childbirth
	muinainensuomi.foorumi.eu	Ancient Finland

Table 5.2: Forums from which the datasets were created. Each forum provides the classification label in the dataset.

Raffel et al. (2020).

We take four languages to form our dataset: English, German, Turkish, and Finnish. These languages are chosen for their range of language families,[2] as well as their range of morphological complexities (see Appendix C.1). These languages also form complex words in different ways, which we investigate in Section 5.5.1. We describe our curation procedure so our work can be extended to other languages—we investigate an approach to doing this for Swahili and Indonesian in Appendix C.3.

5.2.1 Forum Choice

For each language, we take five forums as our classification labels. We take forums which have a large number of pages, but are similar in size to one another, and with distinct, specific topics. This ensures that our dataset creation process provides words with the strongest semantic signals. The lists of forums which are included in mCWIF are shown in Table 5.2.

[2]Uralic, Indo-European, and Turkic.

	# UniqueWords				
	Total	Valid	One Forum	Len > 9	%
English	534 019	115 921	74 428	21 815	29.3%
German	6 119 060	1 463 698	1 161 187	879 071	75.7%
Turkish	4 424 117	1 340 959	1 002 516	555 844	55.4%
Finnish	1 924 811	756 981	593 212	408 831	68.9%

Table 5.3: Counts showing the number of unique words for each language, the number of words that we filter as being valid, the number of valid words occurring in a single forum out of five, and the number of those words with a length of 10 or more characters. The "%" column indicates the percentage of valid words occuring in one column that are 10 or more characters long.

5.2.2 Word Extraction

We begin by extracting word frequencies for each of the forums. We then filter for "valid" words by excluding: words containing non-alphabetic characters, camel-cased words, and words with letters repeating more than twice consecutively. Following this, we lowercase all instances of words to remove effects of capitalisation. Due to the high correlation of word length and morphological complexity (Lewis and Frank, 2016), we filter for words 10 or more characters long—this heuristic, along with the filtering described below, ensures that every word generated is polymorphemic. This process allows us to extract words without having a pre-defined list of morphemes or a way to combine them using morpho-orthographic rules, as in previous work (Hofmann et al., 2020b). In Table 5.3, we show the sizes of the number of words extracted, the number of valid words, the number of words occurring in only one forum, and the number with a length greater than 9 characters (see Appendix C.2 for the counts of non-unique words). Three things of note are that: 1) English has much fewer unique words than the other languages, despite a similar amount of raw data to Finnish (Appendix C.2), which reflects its lower morphological complexity; 2) the number of words occurring in only one forum is a high percentage of the total words, indicating that many words used in a forum are specific; 3) a high percentage of words have a length of 10 or more characters, with the highest percentage for German (76%), and the lowest for English (29%). These last two factors are a result of a Zipfian distribution (Zipf, 1949), with a few words being used many times, and a long tail of less common words. This highlights the importance of models' ability to process complex words: even if the individual words are rare, they make up a large percentage of encountered vocabulary.

5.2.3 Typo Filtering

Since the aim of these datasets is to evaluate the ability of models to handle complex words, the presence of a large amount of typos would impact their quality, as typos may

	mCWIF Small				mCWIF Large			
	N	Raw	-Typos	-Toxic	N	Raw	-Typos	-Toxic
English	1	6 376	4 132	4 087	-	-	-	-
German	75	4 924	4 299	4 264	15	26 156	22 527	22 269
Turkish	20	6 371	4 096	4 034	2	87 149	53 019	52 087
Finnish	13	4 814	3 796	3 766	2	45 598	37 647	37 257

Table 5.4: The values of N per language and dataset size, which is the cut-off frequency per word for inclusion in the dataset. We also show the raw number of words with a count greater than N, along with the counts after typo and toxicity filtering.

be modifications of words which don't have semantics specific to a forum. So, we process the raw data to remove typos. We use GPT-4o (OpenAI, 2023) for typo filtering by passing batches of 50 words and prompting to return any words containing misspellings. We also specify to return proper nouns (including usernames), which would either not be polymorphemic or not of topical relevance, and words not in the relevant language. We specifically tell the model to allow non-standard words. This process is run for three iterations, which we find sufficient to remove all typos. Following success in previous work (Lai et al., 2023; Lin et al., 2022), we prompt the model in English.

5.2.4 Toxicity Filtering

After filtering for typos, we remove any toxic language present in the datasets. GPT-4o was again used, which has been shown in prior work to be effective at this task (Goldzycher et al., 2024). The filtering was run for just one iteration, after which the outputs were judged to be of sufficiently low toxicity. Our prompts for both typo and toxicity filtering are in Appendix C.5.

5.2.5 Final Datasets

To create the final datasets, we filter for words which have the strongest topicality signals for each forum. For each language, we do this by selecting words which occur more than N times in one forum, and not at all in any other forums, where N is tuned to create two dataset sizes. This criterion can be considered as a heavily-weighted version of tf-idf (Sparck Jones, 1972). This value creates a trade-off: for higher N, the words will have stronger topicality signals, but we will be excluding all but the more common words, and there will be less training data available and hence higher variance in results; for lower N, we will include words which are rarer, with more training data and hence more robust results, but at the cost of more noise. To balance these factors, and to investigate their effects, we introduce two datasets per language by tuning the parameter N. We create one dataset with 1 250 total examples (**mCWIF Small**), and one with 12 500 (**mCWIF**

Large), split into train:dev:test sets of 80%:10%:10%. Each class is sampled equally to create balanced datasets. For English, we only have enough data for the Small version.

The counts (values of N) that we use are shown in Table 5.4, along with, for each language, the raw total word counts and the number of words after filtering for both typos and toxic language. The "-Toxic" column is then sampled equally into the five classes, and with final dataset sizes of 1 250 and 12 500 examples. We see that German, with the greatest amount of raw data and the longest average words, has the highest value of N, so we expect the German data to have the strongest topicality signals. English, on the other hand, has the lowest value of N, which gives likely the largest amount of noise, although we still get meaningful results from our experiments on this data (Section 5.4).

We show two examples per language from the dataset in Table 5.1.

5.3 Experiments

We contribute to a growing body of work which analyses the effect of good-quality tokenisation on the performance of language models. Our dataset allows us, for the first time, to examine how the choice of tokeniser impacts the performance of language models on complex word classification across different languages.

As a case study of the utility of our dataset, we use mCWIF to compare two tokenisers: WordPiece and WordPiece′ (Gow-Smith et al., 2022). WordPiece′ is a variant which handles whitespaces by always treating them as individual tokens, in order to ameliorate the issue of subword prefixes such as "##" and "_". For example, BERT's (Devlin et al., 2019) WordPiece tokenises "beat" as `[beat]`, and "unbeatable" as `[un]+[##beat]+[##able]`. Both of these are morphologically valid tokenisations, but the presence of the subword prefix "##" means the two tokens for the substring "beat" are treated as distinct by the model (IDs of 3786 and 19422, respectively). Alongside this issue, WordPiece′ has shown improved morphological performance on English and Finnish when compared to WordPiece (Gow-Smith et al., 2022, 2024). We train both of these tokenisers on the same 1 million sentences from English Wikipedia. In Table 5.5, we repeat those findings and extend them to German and Turkish. In particular, we use MorphyNet (Batsuren et al., 2021), and compare the tokenisations from WordPiece and WordPiece′, following the evaluation protocol introduced by Creutz et al. (2004), reporting precision and F1, along with the average number of tokens (see Section 3.4). Across all four languages, we see better performance for WordPiece′ vs WordPiece.

5.3.1 Finetuning

In order to investigate how WordPiece′ and WordPiece compare on mCWIF across languages, we finetune a range of pretrained encoder language models across the four lan-

	English			German			Turkish			Finnish		
	Len	Precis.	F1	Len	Precis.	F1	Len	Precis.	F1	Len	Precis.	F1
WordPiece	3.43	13.2	19.7	3.22	28.0	38.5	2.65	42.3	52.7	3.21	28.3	38.9
WordPiece′	2.95	**25.5**	**36.1**	3.00	**30.4**	**40.5**	2.40	**47.2**	**55.0**	2.86	**34.7**	**45.0**

Table 5.5: Tokenisation performance of WordPiece and WordPiece′ across four languages: English, German, Turkish, and Finnish, showing the average sequence length, morphological precision, and morphological F1 score (computed on MorphyNet). Best precision and F1 scores are in bold.

guages, and across three or four training scales. Each model is trained with either one of the tokenisers, with everything else fixed. The pretrained models in English and Finnish are taken from Gow-Smith et al. (2024), and we additionally train models in German and Turkish for this work (pretraining details in Appendix C.4). Finetuning is run for 20 epochs on each dataset, updating all parameters, with the following hyperparameters: batch size 32, max sequence length 128, learning rate of 2e-5, warm-up for 5% of steps. We evaluate every epoch on the dev set, taking the best-performing epoch and evaluating it on the test set. We train five seeds for every model on mCWIF Large, and ten seeds for every model on mCWIF Small due to the higher performance variability. The evaluation metric is macro F1, and we report the average across seeds.[3]

5.3.2 Existing Models

In addition to our pretrained models, we run two experiments with existing models. The first uses GPT-4o, zero-shot prompted with the list of forums, the topic of each, and the word of interest (see Appendix C.5 for the prompts). The second is a range of RoBERTa style models: RoBERTa base (Liu et al., 2019) for English, GottBERT (Scheible et al., 2024) for German, BERTurk (Schweter, 2020) for Turkish, and Finnish Bert (Virtanen et al., 2019) for Finnish. These models are run for 10 seeds on mCWIF Small, and 5 seeds on mCWIF Large, and we report the average scores.

5.4 Results

We show the dev and test results on mCWIF Small in Figure 5.1, and those on mCWIF Large in Figure 5.2. The full breakdown of results is in Appendix C.7.

[3]As in Gow-Smith et al. (2024), we find that the English WordPiece model at the V High scale often fails to train. We thus exclude any anomalously low results.

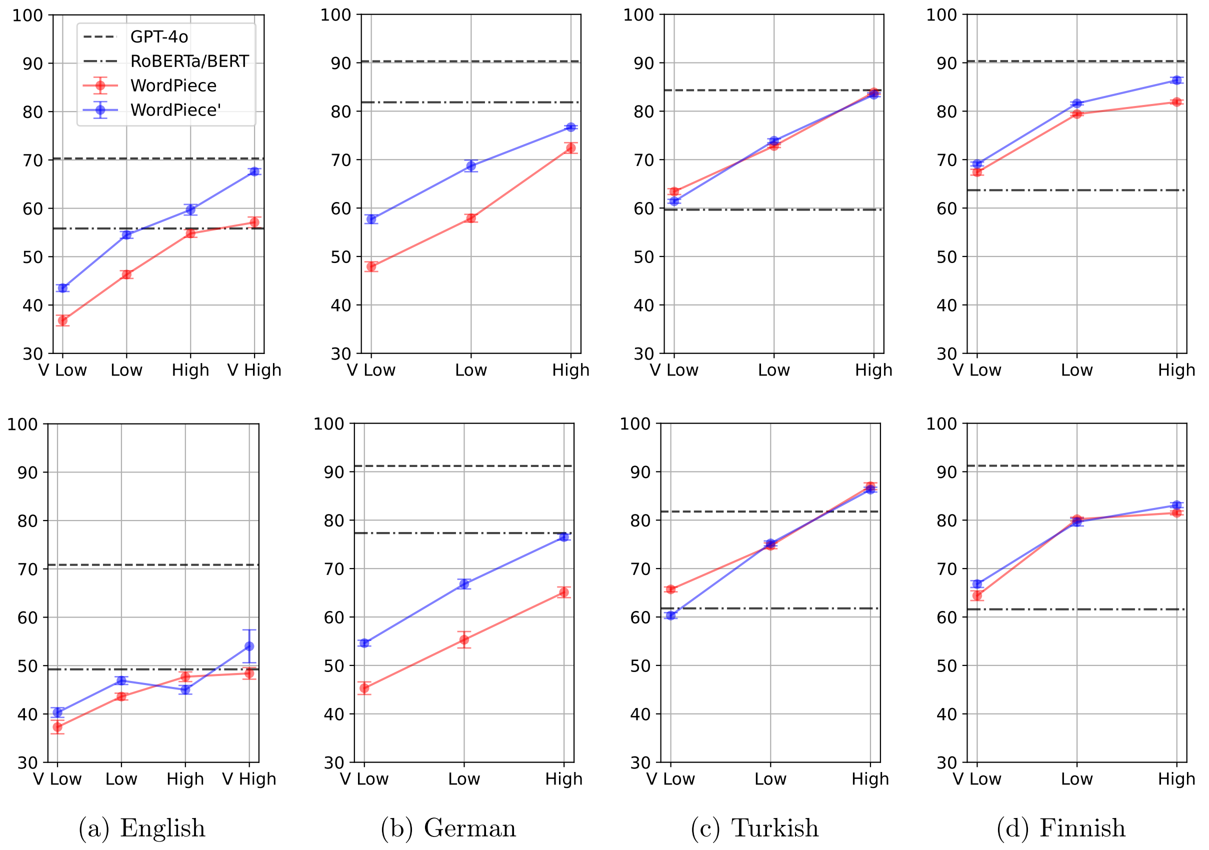
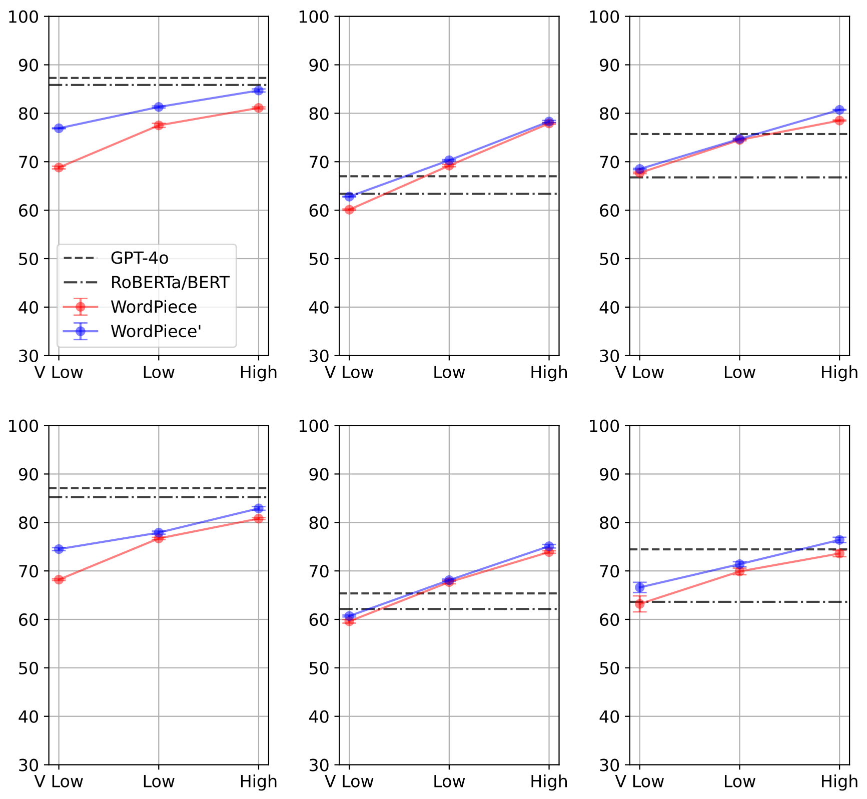


Figure 5.1: Average macro F1 scores across English, German, Turkish, and Finnish for the mCWIF Small dataset on the dev set (top), and the test set (bottom). The red lines are WordPiece, and the blue lines are WordPiece′, across three training scales, with an additional training scale for English. The error bars represent the standard errors. We also report GPT-4o (black dashed line) and a monolingual BERT baseline for each language (black dashed and dotted line).

5.4.1 Existing Models

GPT-4o This model provides a strong baseline in every language, with particularly strong performance relative to the finetuned encoder models for English and German, the two highest-resource languages. We see that its performance is consistently lower for mCWIF Large compared to mCWIF Small, which could either be a result of the weaker topicality signals of words when the cut-off frequency is lower, or the fact that these words are rarer.

Encoder Baselines These models perform variably. GottBERT performs the best, being the only one which outperforms all WordPiece and WordPiece′ models. BERTurk and Finnish Bert perform very poorly: the reasons for this are unclear, and require further investigation.



(a) German (b) Turkish (c) Finnish

Figure 5.2: Average macro F1 scores across German, Turkish, and Finnish for the mCWIF Large dataset on the dev set (top), and the test set (bottom). The red lines are WordPiece, and the blue lines are WordPiece′, across three training scales. The error bars represent the standard errors. We additionally report GPT-4o (black dashed line) and a monolingual BERT baseline for each language (black dashed and dotted line).

5.4.2 WordPiece vs. WordPiece′

When looking at our pretrained models, we find that those trained with WordPiece′ generally outperform those trained with WordPiece, by a magnitude that varies across languages:

English For English, the performance differences are large on the mCWIF Small dev set (6.3 average F1), which is in line with previous results on the Superbizarre dataset (Gow-Smith et al., 2024) where the average F1 improvement was found to be smaller (1.1), but consistent. For the test set, at the High training scale WordPiece′ anomalously underperforms WordPiece, which we attribute to random variability, although further investigation is required; at all other training scales, WordPiece′ performs the best, but with a smaller difference than on the dev set (1.2 average F1).

German For German, there is the greatest performance difference between the two tokenisers. On mCWIF Small, WordPiece′ gives an average improvement of 8.3 dev F1, and 10.7 test F1. On mCWIF Large, the improvements are 5.2 and 3.2 F1 on the dev and test sets, respectively.

Turkish On Turkish, the results are the least conclusive. On mCWIF Small Turkish, WordPiece outperforms WordPiece′ on average (by 0.4 dev F1 and 1.9 test F1), although this is due to the much higher performance at the V Low training scale, which could be anomalous. On mCWIF Large, WordPiece′ follows the trend in other languages and outperforms WordPiece on average, but by the smallest margin (by 1.4 dev F1 and 0.9 test F1).

Finnish For Finnish, averaging results across training scales gives better performance for WordPiece′ compared to WordPiece for both mCWIF Small dev and test, and mCWIF Large dev and test: 2.8, 1.1, 1.1, and 2.6 F1 improvements, respectively. Nevertheless, there is some variability: for mCWIF Small at the Low training scale, WordPiece′ performs worse on the test set by 0.6 F1.

5.5 Discussion

When comparing WordPiece′ to WordPiece, the performance improvement depends on the language. German gives the biggest performance difference, then English, then Finnish, with inconsistent differences on Turkish. In this section, we investigate possible factors influencing this result.

5.5.1 Problematic Space Prefixes

WordPiece′ was introduced to solve the issue of space prefixes occuring in WordPiece tokenisations, which means identical subtokens are treated as separate depending on whether they occur at the start of a word or not. In mCWIF Small English, this is exemplified by the substring "grav" which occurs twice in the training set ("gravimagnetic" and "nongravitational"), once in the dev set ("gravitationally"), and no times in the test set, all from the "cosmoquest" forum. The tokenisations for these by WordPiece are `[gr]+[##av]+[##ima]+[##gn]+[##etic]`, `[non]+[##gr]+[##av]+[##itation]+[##al]` and `[gr]+[##av]+[##itation]+[##al]`. Those by WordPiece′ are `[grav]+[imag]+[netic]`, `[non]+[grav]+[itation]+[al]`, and `[grav]+[itation]+[ally]`. Due to its consistent tokenisation, WordPiece′ has essentially double the training examples of the substring "grav" compared to WordPiece, and this reflects the general improved efficiency of WordPiece′ at leveraging semantic composition for root forms that can occur at different positions across words. But, how does this phenomenon vary across the languages of mCWIF?

		Consistency		
	Position Diversity	WP	WP′	Δ P
English	0.13	0.83	1.00	3.9
German	0.23	0.46	1.00	9.5
Turkish	0.01	0.95	1.00	-1.2
Finnish	0.21	0.63	1.00	2.0

Table 5.6: The position diversity and consistency of WordPiece and WordPiece′ for the four language subsets of mCWIF Small. The position diversity is a measure of how often topically relevant substrings occur in both initial and non-initial word positions. The consistency is a measure of whether a tokensiser gives the same tokenisation for topically relevant substrings across multiple words. The last column is the average performance improvement of WordPiece′ on the dev and test sets of mCWIF Small.

To investigate this, per language and per forum, we look for all substrings with a length between 4 and 8 characters (inclusive), and calculate their frequency as the fraction of the total count of all substrings at that length. Then, for each forum, we order the substrings by the difference between their fraction for that forum and the highest other fraction, and we remove any substrings that only occur in words already covered by a longer susbtring. We then take the first 10 substrings. As an example, for English this gives us [“shift”, “wheel”, “heel”, “tighten”, “applicat”] as the first five substrings for bikeforums, and [“neuro”, “neur”, “opath”, “cobalami”, “viru”] as the first five substrings for phoenixrising. These are taken as the most topically-relevant substrings. Next, for each substring we count how many times it occurs at the start of a word, and how many times it occurs otherwise, and we divide the smaller value by the larger one (the **position diversity**). For example, in English bikeforums, the subword “shift” occurs 0 times at the start of a word, and 8 times otherwise, so its position diversity is 0. The substring “wheel”, on the other hand, occurs 4 times at the start of a word, and 3 times otherwise, giving it a position diversity of 0.75. Next, per language, we average the position diversity for all of the substrings for all forums, shown in Table 5.6. A higher position diversity indicates a potential bigger hinderance to performance from using space prefixes.

We also introduce a **consistency** measure for evaluating the impact of this on a non-space-aware tokeniser. For this, we look at the top 10 most topically-relevant substrings for a forum, and how each substring is tokenised across the words it occurs in. For a tokeniser, we take all substrings that are correctly split from the rest of the word, with more than one occurence. Then, for each substring, we get a consistency score of 1 if every occurence is tokenised the same, and 0 otherwise. Finally, we average this across all substrings from all forums with multiple correctly-split occurences.

We can see that German has the highest position diversity, at 0.23, and Turkish has the lowest, at 0.01. German Wordpice also has the lowest consistency (0.46), and Turkish the highest (0.95). All WordPiece′ models have a consistency of 1. These results indicate that the topically-relevant substrings in Turkish tend to only occur either at the start of a word, or not at the start of the word, with very little flexiblity. Thus, we expect WordPiece′'s removal of space prefixes to have almost no beneficial impact for topicality classification in Turkish, when it comes to leveraging repeated occurence of semantically revelant substrings, which is reflected in our results (Section 5.4.2).

5.5.2 Morphological Validity

The issue of space prefixes isn't the only influence on performance. This is evident from Table 5.6, since English has a lower position diversity than Finnish, and WordPiece′ has a higher consistency in English compared to Finnish, despite the performance difference on mCWIF being larger for English than Finnish. Another factor influencing performance is the morphological validity, with tokenisers with higher morphological alignment having been shown to lead to improved performance on complex words (Gow-Smith et al., 2022; Hofmann et al., 2021, 2022; Gow-Smith et al., 2024). In Table 5.5, we see the biggest difference in F1 for English WordPiece′ compared to WordPiece (+16.4 F1), then Finnish (+6.1 F1), then Turkish (+2.3 F1), then German (+2.0 F1). Perhaps the much bigger difference in morphological validity for WordPiece′ and WordPiece in English when compared to Finnish results in the bigger performance difference on mCWIF Small.

5.5.3 Fertility

The fertility of a tokeniser, defined as the average sequence length per word, is hypothesised to be an indicator of its quality (Scao et al., 2022; Rust et al., 2021). Lower fertility means better compression and subunits which have more defined semantic roles. Nevertheless, there is work which has questioned the correlation between fertility and downstream performance (Ali et al., 2024; Schmidt et al., 2024). In Table 5.5, we see the biggest improvement in fertility ("Len") for English WordPiece′ vs WordPiece (+0.48), then Finnish (+0.35), then Turkish (+0.25) and the least for German (+0.22). This ordering reflects the differences in morphological validity, and could be a further contributor to the good performance of English WordPiece′.

5.5.4 Word Rarity

It is expected that rare morphologically complex words will be more impacted by the quality of tokenisation (Hofmann et al., 2021)—if a word hasn't been seen during pretraining, then the model will have to compute its meaning from the meaning of its subwords. Our mCWIF, German has the highest cut-off value N (Table 5.3), due to the largest starting data and longer words on average, which should give more common words in general than the other languages. On the other hand, English is expected to have more rare words, which gives an unclear effect of this factor.

5.5.5 Overall Analysis

There are many possible factors for the varying performance difference of WordPiece and WordPiece′ across languages. The impact of the issue of space prefixes seems to be the clearest indicator for the biggest performance gap on German, and the lowest on Turkish. Better morphological alignment is also expected to lead to improvements in complex word processing, and we hypothesise that the big improvement in morphological validity for English WordPiece′ compared to WordPiece results in its big performance improvement on mCWIF, despite lower position diversity that Finnish. Beyond that, a disentangling of the factors affecting performance is required, which we leave for future work. Nevertheless, we show that there are language-specific factors (for example for Turkish) which can limit the effect of our tokeniser modifications.

5.6 Conclusion

We introduce a new dataset (**mCWIF**) to evaluate the ability of language models to process complex words across four languages: English, German, Turkish, Finnish. We extract and filter words containing 10 or more characters from internet forums, with the task being to identify a word's forum of origin (and hence topic). Using these datasets, we run extensive finetuning experiments to investigate the performance difference of using a space-aware tokeniser (WordPiece′) when compared to the default (WordPiece). We find that our space-aware tokeniser has the biggest impact on performance for German, then English, then Finnish, then Turkish. We suggest that the impact of WordPiece′ on a language is heightened by whether topically-relevant substrings occur in both word-initial and non-word-initial positions, which we find is very rare in Turkish, and most common in German. Our dataset provides the basis for an investigation of how intrinsic tokenisation and linguistic factors vary across languages, and how they impact the processing of complex words.

Chapter 6

Conclusion

This thesis has investigated the field of subword tokenisation with regards to the performance of language models, looking specifically at the ability of encoder-only transformer models to process complex (polymorphemic) words, and how this is impacted by the way that space symbols such as "##" are handled by state-of-the-art subword tokenisers. In this chapter, we give a summary of the main contributions of the work, along with an overall discussion and possible future directions.

6.1 Main Research Contributions

In **Paper I** (Chapter 3), we introduce a simple modification to existing subword tokenisation algorithms which means they always treat spaces as individual tokens. The resulting space-aware tokenisers are shown to ameliorate known problems with tokenisation, which we investigate through both qualitative and quantitative analysis. Transformer encoder-only models are then trained with BPE and Unigram, along with their space-aware variants (BPE′ and Unigram′), which we find substantially and significantly improves performance on the classification of complex words, whilst having no detrimental impact on general natural language understanding (GLUE) tasks. This bolsters our hypothesis that the morphological alignment of a tokeniser (how well the generated token boundaries align with the morpheme boundaries), alongside the ability to leverage semantic composition across base forms in word-initial and non-word-initial positions, is directly linked to the performance of a language model on complex words.

Having found that our space-aware tokenisers lead to equivalent general-domain language understanding performance when compared to the defaults, despite having no knowledge of word boundaries, the aim of **Paper II** (Chapter 4) is to determine if such information can be included in an alternative way. Specifically, we take WordPiece′ (space-aware WordPiece) and pretrain RoBERTa models with word boundary information either explicitly passed through the input, or used implicitly as part of the pretraining task. We find that, through extensive evaluation in English and Finnish across a range of datasets, these models do not lead to improved performance when compared with the baseline. We thus conclude that we are not losing useful information through removing word boundaries, and that the default masked language modelling task is sufficient to train models effectively.

Paper III (Chapter 5) introduces a novel complex word classification dataset (mCWIF) across four languages: English, German, Turkish, Finnish. This dataset comprises words consisting of 10 or more characters, extracted from internet forums, and then filtered to give the strongest topicality signals. The aim is to classify a word by its forum of origin. We use this dataset to evaluate the performance of WordPiece and WordPiece′, providing a multilingual analysis of the effect of tokenisation on complex word processing for the first time. We find the biggest performance improvement of WordPiece′ on German, and the least on Turkish. We investigate potential reasons for this result, including: the issue of space prefixes in WordPiece and how this impacts performance across languages; morphological validity; fertility; and word rarity. We introduce the metrics of position diversity and consistency to evaluate how space prefixes impact tokenisation across languages, finding from these metrics that topically-relevant subunits in Turkish typically occur either in a word-initial or non-word-initial position, but rarely both, which we suggest contributes to the similar performance of WordPiece and WordPiece′ in that language.

6.2 Discussion

This thesis contributes to a growing body of work surrounding the impact of subword tokenisation. Here we discuss the main hypotheses that we propose and evaluate throughout the work.

The results across all three publications support the hypothesis that the use of space prefixes by tokenisers can be detrimental to the performance of language models when processing complex words. We find that handling spaces as individual tokens leads to better performance of encoder-only transformer models on complex word classification for English, German, and Finnish, with inconsistent results for Turkish. We perform extensive evaluation across languages and datasets for our modified space-aware tokenisers (BPE′, Unigram′, and WordPiece′) compared to the defaults. In Chapter 3, we find that Unigram′ gives improvements upon Unigram by between 0.9 and 1.1 F1 for the dev and test sets of the Superbizarre Reddit and Arxiv datasets. In Chapter 4, we find similar performance improvements for WordPiece′ vs. WordPiece on the same Superbizarre datasets of 1.1 average F1, and we get an average performance increase of 4.5 for FLOTA. In Chapter 5, we evaluate WordPiece and WordPiece′ on our mCWIF dataset, finding the following average test F1 performance increases: 1.2 for the English subset; 10.7 and 3.2 for German Small and Large, respectively; 1.1 and 2.6 for Finnish Small and Large, respectively. For Turkish, we see a decrease of 1.9 test F1 for the Small version, and an increase of 0.9 test

F1 for the Large version.

We link these extrinsic measures to intrinsic ones, and throughout our work, we provide support to the hypothesis that more morphologically valid tokenisation leads to better performance on morphologically complex words, in line with prior results (Hofmann et al., 2021, 2022). In Chapter 3, we find an average morphological validity of 43.0 F1 for BPE and 50.9 F1 for BPE′. For Unigram, we find an average morphological validity of 59.7, and for Unigram′ it's 62.4. We further show that these improvements are mostly down to the improvement on splitting prefixes, which we hypothesise is due to the problematic space symbols ("##") prefixed to subtokens in state-of-the-art subword tokenisers. In Chapter 4, we compare WordPiece and WordPiece′ for English and Finnish, finding WordPiece to give 33.8 F1 and 38.9 F1, respectively, whilst WordPiece′ gives 52.7 F1 and 45.0 F1. In Chapter 5, we extend this analysis to German and Turkish, finding consistent performance improvements (38.5 to 40.5 for German, 52.7 to 55.0 for Turkish). These intrinsic improvements are linked with extrinsic improvements on complex word classification stated prior, except for on Turkish, where the results are inconclusive. We investigate this further in Chapter 5, suggesting that the problematic space symbols may be much less of an issue in Turkish compared to the other languages—for Turkish, we show that topically-relevant subunits occur either at the start of a word, or not, but very rarely in both positions.

In Chapters 3 and 4, we show that, for the range of downstream datasets that we use for evaluation, word boundary information doesn't provide performance improvements for encoder-only transformer models. In Chapter 3, our Unigram′ and BPE′ models perform equivalently on GLUE to Unigram and BPE, despite having no word boundary information. In Chapter 3, we experiment with a range of explicit and implicit approaches to adding word boundary information into RoBERTa models, finding no consistent performance improvements on both GLUE and three named entity recognition datasets in English, and two sequence classification and one named entity recognition dataset in Finnish. These results suggest that the default paradigm for masked language modelling is sufficient to encode all useful subunit information for models. We suggest this is because the morpheme, rather than the word, is the most important subunit.

6.3 Future Work

This thesis has made contributions to the field of subword tokenisation, in particular with regard to the importance of morphological validity when assessing tokenisers, and insights into the drawbacks of current approaches for handling whitespace in language models. Our work has opened up further research avenues and questions.

Firstly, our analysis in Chapters 3 and 4 suggests that word boundary information is not useful for language models, no matter the task. Nevertheless, one can trivially come up with examples of natural language where removing such information leaves ambiguity. Some examples in English are adjective-noun compounds which are not semantically transparent: “he lived in the green house” vs. “he lived in the greenhouse”. Perhaps context is always sufficient for disambiguating such occurrences, but this investigation is left to future work.

Our work has looked at encoder-only transformers for NLP, without considering encoder-decoder or decoder-only set-ups. This is because, at the time of starting this thesis, pretrained and finetuned encoder-only models such as BERT were the state-of-the-art. Since then, there has been a shift to decoder-only large language models such as GPT-4 and Llama, which reduces the relevance of our work: decoder-only models require space information to be generated in the output, so our lossy space-aware tokenisers can’t be used out-of-the-box. Approaches to generating space information in the output for these models are left to future investigations.

In Chapter 5, we evaluate the performance of our space-aware subword tokenisers across four languages (English, German, Turkish, Finnish), providing for the first time a multilingual analysis of the effect of tokenisation on complex word processing. However, the choice of languages is small, euro-centric, and rather high-resource. It would be insightful to extend this analysis to more, and more diverse, languages. We also investigate multiple factors that could contribute to the different performance gaps between WordPiece′ and WordPiece across the four languages, in particular the big gap in German and the inconsistent gap in Turkish. We suggest that our metric of “position diversity” explains partly this results, but further analysis is required to disentangle the effect of other factors such as morphological validity, word rarity, and frequency of semantically relevant subunits. Having more languages in the dataset would also help with determining which factors are contributing most to performance.

Furthermore, the use of the heuristic of 10 or more characters for filtering for complex words in Chapter 5 may bias the results, and this is not something we have yet investigated. For example, looking at only longer words in German may favour the inclusion of noun compounds in mCWIF, rather than words with inflectional morphology.

Across all of our work, we have used a single vocabulary size of 16 000. Due to the high computational (and environmental) cost of pretraining, running our experiments across different vocabulary sizes wasn’t feasible. Nevertheless, this limits our analysis, especially when proposing fundamental modifications to state-of-the-art subword tokenisers. In particular, in Chapter 5, a higher vocabulary size would perhaps help with the agglutinative languages of Finnish and Turkish, and could be a reason for the similar performance for these languages on mCWIF.

References

Mostafa Abdou, Vinit Ravishankar, Artur Kulmizev, and Anders Sø gaard. 2022. Word order does matter and shuffled language models know it. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6907–6919, Dublin, Ireland. Association for Computational Linguistics.

Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Buschhoff, Charvi Jain, Alexander Weber, Lena Jurkschat, Hammam Abdelwahab, Chelsea John, Pedro Ortiz Suarez, Malte Ostendorff, Samuel Weinbach, Rafet Sifa, Stefan Kesselheim, and Nicolas Flores-Herr. 2024. Tokenizer choice for LLM training: Negligible or crucial? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3907–3924, Mexico City, Mexico. Association for Computational Linguistics.

Chantal Amrhein and Rico Sennrich. 2021. How suitable are subword segmentation strategies for translating non-concatenative morphology? In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 689–705, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jeremy M Anglin, George A Miller, and Pamela C Wakefield. 1993. Vocabulary development: A morphological analysis. *Monographs of the society for research in child development*, pages i–186.

Catherine Arnett and Benjamin Bergen. 2025. Why do language models perform worse for morphologically complex languages? In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 6607–6623, Abu Dhabi, UAE. Association for Computational Linguistics.

He Bai, Peng Shi, Jimmy Lin, Yuqing Xie, Luchen Tan, Kun Xiong, Wen Gao, and Ming Li. 2021. Segatron: Segment-aware transformer for language modeling and understanding. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 12526–12534. AAAI Press.

David A Balota, Melvin J Yap, Keith A Hutchison, Michael J Cortese, Brett Kessler, Bjorn Loftis, James H Neely, Douglas L Nelson, Greg B Simpson, and Rebecca Treiman. 2007. The english lexicon project. *Behavior research methods*, 39(3):445–459.

Khuyagbaatar Batsuren, Gábor Bella, and Fausto Giunchiglia. 2021. MorphyNet: a large multilingual database of derivational and inflectional morphology. In *Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 39–48, Online. Association for Computational Linguistics.

Khuyagbaatar Batsuren, Ekaterina Vylomova, Verna Dankers, Tsetsuukhei Delgerbaatar, Omri Uzan, Yuval Pinter, and Gábor Bella. 2024. Evaluating subword tokenization: Alien subword composition and OOV generalization challenge. *CoRR*, abs/2404.13292.

Laurie Bauer. 2003. Introducing linguistic morphology. In *Introducing Linguistic Morphology*. Edinburgh university press.

Lisa Beinborn and Yuval Pinter. 2023. Analyzing cognitive plausibility of subword tokenization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4478–4486, Singapore. Association for Computational Linguistics.

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2009. The fifth PASCAL recognizing textual entailment challenge. In *Proceedings of the Second Text Analysis Conference, TAC 2009, Gaithersburg, Maryland, USA, November 16-17, 2009*. NIST.

Christian Bentz, Tatyana Ruzsics, Alexander Koplenig, and Tanja Samardžić. 2016. A comparison between morphological complexity measures: Typological data vs. language corpora. In *Proceedings of the Workshop on Computational Linguistics for Linguistic Complexity (CL4LC)*, pages 142–153, Osaka, Japan. The COLING 2016 Organizing Committee.

Steven Bird, Ewan Klein, and Edward Loper. 2009. Nltk book.

Harry Bochner. 1993. *Simplicity in generative morphology*, volume 37. Walter de Gruyter.

Geert Booij. 2012. *The grammar of words: An introduction to linguistic morphology*. Oxford University Press.

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, Online. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Andrew Carstairs-McCarthy. 2017. *Introduction to English Morphology: words and their structure*. Edinburgh university press.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Keh-Jiann Chen and Shing-Huan Liu. 1992. Word identification for mandarin chinese sentences. In *14th International Conference on Computational Linguistics, COLING 1992, Nantes, France, August 23-28, 1992*, pages 101–107.

Ziling Cheng, Rahul Aralikatte, Ian Porada, Cesare Spinoso-Di Piano, and Jackie CK Cheung. 2023. McGill BabyLM shared task submission: The effects of data formatting and structural biases. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 207–220, Singapore. Association for Computational Linguistics.

Tung-Hui Chiang, Jing-Shin Chang, Ming-Yu Lin, and Keh-Yih Su. 1992. Statistical models for word segmentation and unknown word resolution. In *Proceedings of Rocling V Computational Linguistics Conference V*, pages 123–146, Taipei, Taiwan. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP).

Kenneth Ward Church. 2020. Emerging trends: Subwords, seriously? *Nat. Lang. Eng.*, 26(3):375–382.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Bernard Comrie. 1989. *Language universals and linguistic typology: Syntax and morphology*. University of Chicago press.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 21–30. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using morfessor 1.0. In *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology.

Mathias Johan Philip Creutz, Bo Krister Johan Linden, et al. 2004. Morpheme segmentation gold standards for finnish and english. *Publications in Computer and Information Science Report A77*.

Yiming Cui, Wanxiang Che, Shijin Wang, and Ting Liu. 2022. LERT: A linguistically-motivated pre-trained language model. *CoRR*, abs/2211.05344.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Marco Del Tredici and Raquel Fernández. 2018. The road to success: Assessing the fate of linguistic innovations in online communities. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1591–1603, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147, Copenhagen, Denmark. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Rezarta Islamaj Dogan, Robert Leaman, and Zhiyong Lu. 2014. NCBI disease corpus: A resource for disease name recognition and concept normalization. *J. Biomed. Informatics*, 47:1–10.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online (v2020.4)*. Zenodo.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. 2020. CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6903–6915, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Zhangyin Feng, Duyu Tang, Xiaocheng Feng, Cong Zhou, Junwei Liao, Shuangzhi Wu, Bing Qin, Yunbo Cao, and Shuming Shi. 2024. Pretraining without wordpieces: learning over a vocabulary of millions of words. *Int. J. Mach. Learn. Cybern.*, 15(9):3989–3998.

Philip Gage. 1994. A new algorithm for data compression. *C Users Journal*, 12(2):23–38.

Christina L. Gagné, Thomas L. Spalding, and Daniel Schmidtke. 2019. Ladec: The large database of english compounds. *Behavior Research Methods*, 51(5):2152–2179.

Matthias Gallé. 2019. Investigating the effectiveness of BPE: The power of shorter sequences. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1375–1381, Hong Kong, China. Association for Computational Linguistics.

Hélène Giraudo and Jonathan Grainger. 2003. On the role of derivational affixes in recognizing complex words. In R.H. Baayen & R. Schreuder, editor, *Morphological Structure in Language Processing*. De Gruyter Mouton.

Omer Goldman, Avi Caciularu, Matan Eyal, Kris Cao, Idan Szpektor, and Reut Tsarfaty. 2024. Unpacking tokenization: Evaluating text compression and its correlation with model performance. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2274–2286, Bangkok, Thailand. Association for Computational Linguistics.

Janis Goldzycher, Paul Röttger, and Gerold Schneider. 2024. Improving adversarial data collection by supporting annotators: Lessons from GAHD, a German hate speech dataset. pages 4405–4424.

Edward Gow-Smith, Dylan Phelps, Harish Tayyar Madabushi, Carolina Scarton, and Aline Villavicencio. 2024. Word boundary information isn‘t useful for encoder language models. In *Proceedings of the 9th Workshop on Representation Learning for NLP (RepL4NLP-2024)*, pages 118–135, Bangkok, Thailand. Association for Computational Linguistics.

Edward Gow-Smith, Harish Tayyar Madabushi, Carolina Scarton, and Aline Villavicencio. 2022. Improving tokenisation by alternative treatment of spaces. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11430–11443, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Gregory Grefenstette and Pasi Tapanainen. 1994. What is a word, what is a sentence? problems of tokenisation. Technical report, Rank Xerox Research Centre, Grenoble Laboratory, Meylan, France.

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. 2020. Morfessor EM+Prune: Improved subword segmentation with expectation maximization and pruning. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 3944–3953, Marseille, France. European Language Resources Association.

Martin Haspelmath and Andrea Sims. 2013. *Understanding morphology.* Routledge.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. 2020. Dynamic programming encoding for subword segmentation in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3042–3051, Online. Association for Computational Linguistics.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. 2022. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Valentin Hofmann, Janet Pierrehumbert, and Hinrich Schütze. 2020a. DagoBERT: Generating derivational morphology with a pretrained language model. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3848–3861, Online. Association for Computational Linguistics.

Valentin Hofmann, Janet Pierrehumbert, and Hinrich Schütze. 2020b. Predicting the growth of morphological families from social and linguistic factors. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7273–7283, Online. Association for Computational Linguistics.

Valentin Hofmann, Janet Pierrehumbert, and Hinrich Schütze. 2021. Superbizarre is not superb: Derivational morphology improves BERT's interpretation of complex words. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3594–3608, Online. Association for Computational Linguistics.

Valentin Hofmann, Hinrich Schuetze, and Janet Pierrehumbert. 2022. An embarrassingly simple method to mitigate undesirable properties of pretrained language model tokenizers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–393, Dublin, Ireland. Association for Computational Linguistics.

Cassandra L. Jacobs and Yuval Pinter. 2022. Lost in space marking. *CoRR*, abs/2208.01561.

Ye Jia, Heiga Zen (Byungha Chun), Jonathan Shen, Yu Zhang, and Yonghui Wu. 2021. Png bert: Augmented bert on phonemes and graphemes for neural tts. In *Interspeech*.

Bernal Jimenez Gutierrez, Huan Sun, and Yu Su. 2023. Biomedical language models are robust to sub-optimal tokenization. In *The 22nd Workshop on Biomedical Natural Language Processing and BioNLP Shared Tasks*, pages 350–362, Toronto, Canada. Association for Computational Linguistics.

Patrick Juola. 1998. Measuring linguistic complexity: The morphological tier. *Journal of Quantitative Linguistics*, 5(3):206–213.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Kimmo Kettunen. 2014. Can type-token ratio be used to show morphological complexity of languages? *Journal of Quantitative Linguistics*, 21(3):223–245.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Stav Klein and Reut Tsarfaty. 2020. Getting the ##life out of living: How adequate are word-pieces for modelling complex morphology? In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 204–209, Online. Association for Computational Linguistics.

Philipp Koncar, Simon Walk, and Denis Helic. 2021. Analysis and prediction of multilingual controversy on reddit. In *WebSci '21: 13th ACM Web Science Conference 2021, Virtual Event, United Kingdom, June 21-25, 2021*, pages 215–224. ACM.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Viet Dac Lai, Nghia Ngo, Amir Pouran Ben Veyseh, Hieu Man, Franck Dernoncourt, Trung Bui, and Thien Huu Nguyen. 2023. ChatGPT beyond English: Towards a comprehensive evaluation of large language models in multilingual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13171–13189, Singapore. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*. AAAI Press.

Yoav Levine, Barak Lenz, Or Dagan, Ori Ram, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2020. SenseBERT: Driving some sense into BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4656–4667, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Molly L Lewis and Michael C Frank. 2016. The length of words reflects their conceptual complexity. *Cognition*, 153:182–195.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona Diab, Veselin Stoyanov, and Xian Li. 2022. Few-shot learning with multilingual generative language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9019–9052, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Krister Lindén, Tommi Jauhiainen, and Sam Hardwick. 2023. Finnsentiment: a finnish social media corpus for sentiment polarity annotation. *Lang. Resour. Evaluation*, 57(2):581–609.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

G Long, I Morrison, and D Barnett. 1985. Text compression using word tokenization. Technical report, Lawrence Livermore National Lab., CA (USA).

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Peter Hugoe Matthews. 1991. *Morphology*. Cambridge university press.

Arya D. McCarthy, Christo Kirov, Matteo Grella, Amrit Nidhi, Patrick Xia, Kyle Gorman, Ekaterina Vylomova, Sabrina J. Mielke, Garrett Nicolai, Miikka Silfverberg, Timofey Arkhangelskiy, Nataly Krizhanovsky, Andrew Krizhanovsky, Elena Klyachko, Alexey Sorokin, John Mansfield, Valts Ernštreits, Yuval Pinter, Cassandra L. Jacobs, Ryan Cotterell, Mans Hulden, and David Yarowsky. 2020. UniMorph 3.0: Universal Morphology. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 3922–3931, Marseille, France. European Language Resources Association.

Sabrina J. Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y. Lee, Benoît Sagot, and Samson Tan. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP. *CoRR*, abs/2112.10508.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Anmol Nayak, Hariprasad Timmapathini, Karthikeyan Ponnalagu, and Vijendran Gopalan Venkoparao. 2020. Domain adaptation challenges of BERT in tokenization and sub-word representations of out-of-vocabulary words. In *Proceedings of the First Workshop on Insights from Negative Results in NLP*, pages 1–5, Online. Association for Computational Linguistics.

Akunna Onyedum. 2012. Social media neologisms: A morpho-semantic analysis. *Lagos: University of Lagos*.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

David D Palmer. 2000. Tokenisation and sentence segmentation. *Handbook of natural language processing*, pages 11–35.

Hyunji Hayley Park, Katherine J. Zhang, Coleman Haley, Kenneth Steimel, Han Liu, and Lane Schwartz. 2021. Morphology matters: A multilingual language modeling analysis. *Transactions of the Association for Computational Linguistics*, 9:261–276.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Yuval Pinter, Cassandra L. Jacobs, and Jacob Eisenstein. 2020. Will it unblend? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1525–1535, Online. Association for Computational Linguistics.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Tony Rose, Mark Stevenson, and Miles Whitehead. 2002. The Reuters corpus volume 1 -from yesterday's news to tomorrow's language resources. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, Las Palmas, Canary Islands - Spain. European Language Resources Association (ELRA).

Teemu Ruokolainen, Pekka Kauppinen, Miikka Silfverberg, and Krister Lindén. 2020. A finnish news corpus for named entity recognition. *Lang. Resour. Evaluation*, 54(1):247–272.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135, Online. Association for Computational Linguistics.

Claudia H Sánchez-Gutiérrez, Hugo Mailhot, S Hélène Deacon, and Maximiliano A Wilson. 2018. Morpholex: A derivational morphological database for 70,000 english words. *Behavior research methods*, 50(4):1568–1580.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilic, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, and et al.

2022. BLOOM: A 176b-parameter open-access multilingual language model. *CoRR*, abs/2211.05100.

Raphael Scheible, Johann Frei, Fabian Thomczyk, Henry He, Patric Tippmann, Jochen Knaus, Victor Jaravine, Frank Kramer, and Martin Boeker. 2024. GottBERT: a pure German language model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21237–21250, Miami, Florida, USA. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2020. BERTRAM: Improved word embeddings have big impact on contextualized model performance. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3996–4007, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2020. Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8766–8774. AAAI Press.

Craig W Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, and Chris Tanner. 2024. Tokenization is more than compression. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 678–702, Miami, Florida, USA. Association for Computational Linguistics.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2012, Kyoto, Japan, March 25-30, 2012*, pages 5149–5152. IEEE.

Stefan Schweter. 2020. Berturk - bert models for turkish.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Shakina Shahlee and Salawati Ahmad. 2022. Morphological processes of social media neologisms. *Development in Language Studies*, 2(1):19–29.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Richard William Sproat. 1992. *Morphology and computation.* MIT press.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. ERNIE: enhanced representation through knowledge integration. *CoRR*, abs/1904.09223.

Peter Svenonius and Patrik Bye. 2011. Non-concatenative morphology as epiphenomenon.

Ann Taylor, Mitchell Marcus, and Beatrice Santorini. 2003. The penn treebank: an overview. *Treebanks*, pages 5–22.

Harish Tayyar Madabushi, Edward Gow-Smith, Carolina Scarton, and Aline Villavicencio. 2021. AStitchInLanguageModels: Dataset and methods for the exploration of idiomaticity in pre-trained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3464–3477, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ken Thompson. 1968. Programming techniques: Regular expression search algorithm. *Commun. ACM*, 11(6):419–422.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Omri Uzan, Craig W. Schmidt, Chris Tanner, and Yuval Pinter. 2024. Greed is all you need: An evaluation of tokenizer inference methods. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 813–822, Bangkok, Thailand. Association for Computational Linguistics.

C.J. Van Rijsbergen. 1975. *Information Retrieval*. Butterworths.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. pages 5998–6008.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, Mikko Kurimo, et al. 2013. Morfessor 2.0: Python implementation and extensions for morfessor baseline.

Antti Virtanen, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, and Sampo Pyysalo. 2019. Multilingual is not enough: BERT for finnish. *CoRR*, abs/1912.07076.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3261–3275.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. 2020. Structbert: Incorporating language structures into pre-training for deep language understanding. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. 2023. Findings of the BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 1–34, Singapore. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Alex Warstadt, Yian Zhang, Xiaocheng Li, Haokun Liu, and Samuel R. Bowman. 2020. Learning which features matter: RoBERTa acquires a preference for linguistic generalizations (eventually). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 217–235, Online. Association for Computational Linguistics.

Jonathan J. Webster and Chunyu Kit. 1992. Tokenization as the initial phase in NLP. In *COLING 1992 Volume 4: The 14th International Conference on Computational Linguistics*.

Junqiu Wei, Qun Liu, Yinpeng Guo, and Xin Jiang. 2021. Training multilingual pre-trained language model with byte-level subwords. *CoRR*, abs/2101.09469.

Junqiu Wei, Xiaozhe Ren, Xiaoguang Li, Wenyong Huang, Yi Liao, Yasheng Wang, Jiashu Lin, Xin Jiang, Xiao Chen, and Qun Liu. 2019. NEZHA: neural contextualized representation for chinese language understanding. *CoRR*, abs/1909.00204.

Matheus Westhelle, Luciana Bencke, and Viviane P. Moreira. 2022. Impact of morphological segmentation on pre-trained language models. In *Intelligent Systems - 11th Brazilian Conference, BRACIS 2022, Campinas, Brazil, November 28 - December 1, 2022, Proceedings, Part II*, volume 13654 of *Lecture Notes in Computer Science*, pages 402–416. Springer.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Atsuki Yamaguchi, George Chrysostomou, Katerina Margatina, and Nikolaos Aletras. 2021. Frustratingly simple pretraining alternatives to masked language modeling. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3116–3125, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Meng Ye, Karan Sikka, Katherine Atwell, Sabit Hassan, Ajay Divakaran, and Malihe Alikhani. 2023. Multilingual content moderation: A case study on Reddit. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3828–3844, Dubrovnik, Croatia. Association for Computational Linguistics.

Shaked Yehezkel and Yuval Pinter. 2023. Incorporating context into subword vocabularies. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 623–635, Dubrovnik, Croatia. Association for Computational Linguistics.

Junru Zhou, Zhuosheng Zhang, Hai Zhao, and Shuailiang Zhang. 2020. LIMIT-BERT : Linguistics informed multi-task BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4450–4461, Online. Association for Computational Linguistics.

George Kingsley Zipf. 1949. Human behavior and the principle of least effort.

Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Mrinmaya Sachan, and Ryan Cotterell. 2023. Tokenization and the noiseless channel. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5184–5207, Toronto, Canada. Association for Computational Linguistics.

Appendix A

A.1 Training Details

Hyperparameters for tokenisation, pretraining, finetuning are shown in Table A.1, Table A.2 and Table A.3, respectively. We do not use stochastic tokenisation (BPE-dropout or subword regularisation).

Implementation	SentencePiece (Kudo and Richardson, 2018)
Vocabulary Size	16000
BPE-dropout	0
Unigram Subword Regularisation	0

Table A.1: Hyperparameters for tokenisation.

Implementation	fairseq (Ott et al., 2019)
Architecture	RoBERTa (base) (Liu et al., 2019)
Precision	16 bit
Optimizer	ADAM (Kingma and Ba, 2015), ϵ =1e-6, $\beta = (0.9, 0.98)$
Sequence length	512
Learning rate scheduler	Linear warm-up for 10000 updates to 5e-4, then reduce to 1e-4 upon increased training loss at epoch
Training for	100000 updates / 30 epochs
Batch size	2048
Dropout	0.1
Attention Dropout	0.1
Weight Decay	0.01

Table A.2: Hyperparameters for pretraining.

A.1.1 Pretraining

Pretraining was run on 8 NVIDIA Tesla V100s. We ran pretraining on the text of English Wikipedia. A Wikipedia dump was processed with the Python package WikiExtractor[1],

[1]https://github.com/attardi/wikiextractor/

Implementation	fairseq (Ott et al., 2019)
Architecture	RoBERTa (base) (Liu et al., 2019)
Precision	16 bit
Optimizer	ADAM (Kingma and Ba, 2015), ϵ =1e-6, $\beta = (0.9, 0.98)$
Sequence length	512
Learning rate scheduler	Linear warm-up to 2e-3 for 6% of updates, then linear decay to 0
Training for	20 epochs
Batch size	32
Dropout	0.1
Attention Dropout	0.1
Weight Decay	0.01

Table A.3: Hyperparameters for finetuning.

and then split into sentences using BlingFire[2]. In order to perform a fair comparison across models, we removed all sentences with sequence lengths longer than 510 when tokenised with the modified models including spaces. However, this was a very small amount of the data (~0.002%) and would therefore have a negligible effect on performance. Loss curves are shown in Figure A.1.

A.1.2 Finetuning

Finetuning was run on a single NVIDIA Tesla V100. All finetuning experiments were ran with a batch size of 32, and a peak learning rate of 2e-3 with linear warm-up for 6% of updates, then linear decay to 0. All other parameters were kept the same as for pretraining. Experiments were ran for 20 epochs, and the best performing epoch was taken, with 10 random seeds per model. For the Superbizarre datasets, we took the best performing epoch for each seed on the dev set and evaluated it on the test set.

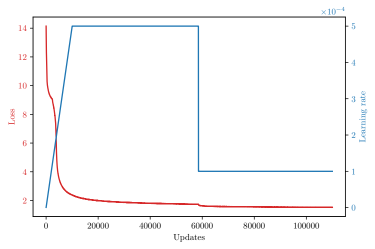
A.2 Detailed Results

Detailed results are shown in Table A.4 and Table A.6 for fixed pretraining updates and fixed pretraining epochs, respectively. We show the standard deviations in Tables A.5 and A.7: on the mean GLUE score, these are calculated assuming zero covariance between tasks.

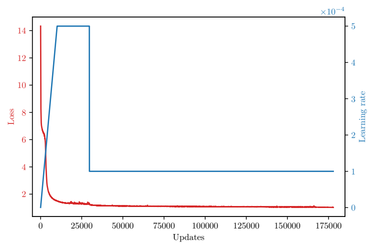
A.3 Significance Tests

Here we give full Welch's t-test results comparing the best performing model to all the others for each dataset, shown in Table A.8 and Table A.9 for fixed pretraining updates and fixed pretraining epochs, respectively.

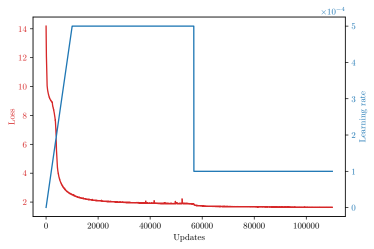
[2]https://github.com/microsoft/BlingFire



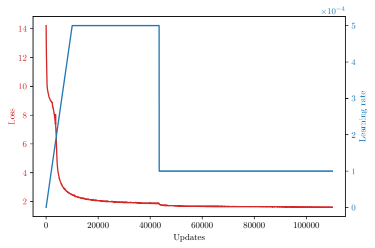
(a) BPE



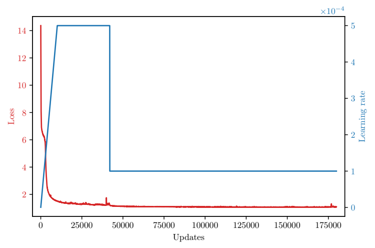
(b) BPE′



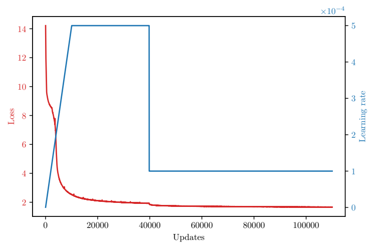
(c) BPE′ no spaces



(d) Unigram



(e) Unigram′



(f) Unigram′ no spaces

Figure A.1: Pretraining loss curves for the six models.

	Epochs	GLUE										SB Reddit		SB Arxiv	
		MRPC	CoLA	STS-B	RTE	SST-2	QQP	QNLI	MNLI-m	MNLI-mm	Mean	Dev	Test	Dev	Test
BPE	27	84.5	55.4	87.1	68.6	91.6	89.7	91.3	83.1	83.5	81.6	66.8	66.6	71.1	70.2
BPE′	16	83.0	48.9	86.0	59.5	91.6	89.2	90.7	81.6	82.3	79.2	66.6	66.2	70.3	69.3
BPE′ no spaces	28	84.4	54.4	87.0	70.3	92.2	89.7	91.1	83.1	83.2	81.7	67.2	66.9	70.9	70.0
Unigram	27	85.0	52.3	87.3	69.8	91.7	89.5	91.9	83.1	83.1	81.5	68.0	67.8	72.2	71.4
Unigram′	16	83.3	39.5	84.8	64.0	91.3	89.1	89.8	81.4	82.1	78.4	68.2	68.2	72.5	71.6
Unigram′ no spaces	27	85.2	54.6	87.8	71.1	91.6	89.5	91.3	83.0	83.1	81.9	68.8	68.8	73.0	72.3

Table A.4: Full finetuning results after pretraining for 100000 updates. Shown are mean dev set results across 10 seeds. SB is short for Superbizarre.

	Epochs	GLUE										SB Reddit		SB Arxiv	
		MRPC	CoLA	STS-B	RTE	SST-2	QQP	QNLI	MNLI-m	MNLI-mm	Mean	Dev	Test	Dev	Test
BPE	27	0.8	2.5	0.3	2.7	0.4	0.1	0.2	0.2	0.3	1.3	0.8	0.9	0.2	0.2
BPE′	16	1.0	2.9	0.2	1.9	0.4	0.1	0.3	0.2	0.1	1.2	0.2	0.2	0.1	0.2
BPE′ no spaces	28	0.6	1.4	0.2	0.8	0.5	0.1	0.2	0.2	0.2	0.6	0.2	0.2	0.1	0.2
Unigram	27	1.2	1.4	0.2	1.9	0.5	0.1	0.4	0.2	0.2	0.9	0.2	0.3	0.3	0.2
Unigram′	16	0.6	15.4	0.4	1.8	0.4	0.1	0.3	0.2	0.2	5.2	0.4	0.3	0.2	0.3
Unigram′ no spaces	27	1.4	1.4	0.3	1.5	0.4	0.1	0.3	0.2	0.2	0.9	0.1	0.3	0.2	0.3

Table A.5: Standard deviations for finetuning results after pretraining for 100000 updates. Shown are deviations across 10 seeds. SB is short for Superbizarre.

	Updates	GLUE										SB Reddit		SB Arxiv	
		MRPC	CoLA	STS-B	RTE	SST-2	QQP	QNLI	MNLI-m	MNLI-mm	Mean	Dev	Test	Dev	Test
BPE	109761	84.4	53.5	87.2	68.7	91.8	89.7	91.4	83.1	83.5	81.5	67.1	66.8	71.0	70.1
BPE′	177845	83.2	48.9	86.6	60.0	92.0	89.2	90.7	82.2	82.9	79.5	66.8	66.5	70.5	69.8
BPE′ no spaces	106485	85.0	53.4	86.9	69.1	92.0	89.5	91.2	83.2	83.2	81.5	67.1	67.1	70.8	70.1
Unigram	108606	84.8	53.1	87.4	70.1	91.6	89.6	91.3	83.0	83.2	81.6	67.9	67.9	72.2	71.6
Unigram′	179909	82.0	45.9	84.7	64.9	91.5	89.0	90.1	81.5	82.0	79.1	68.3	68.3	72.5	71.8
Unigram′ no spaces	108441	84.8	54.5	87.8	70.0	91.5	89.6	91.5	83.2	83.2	81.8	68.8	69.0	73.2	72.5

Table A.6: Full finetuning results after pretraining for 30 epochs. Shown are mean dev set results across 10 seeds. SB is short for Superbizarre.

	Updates	GLUE										SB Reddit		SB Arxiv	
		MRPC	CoLA	STS-B	RTE	SST-2	QQP	QNLI	MNLI-m	MNLI-mm	Mean	Dev	Test	Dev	Test
BPE	109761	0.8	1.7	0.2	0.9	0.3	0.1	0.2	0.2	0.3	0.7	0.2	0.3	0.2	0.3
BPE′	177845	1.1	1.4	0.2	2.6	0.2	0.0	0.3	0.2	0.2	1.1	0.3	0.1	0.1	0.2
BPE′ no spaces	106485	0.6	0.9	0.3	0.6	0.4	0.1	0.3	0.2	0.2	0.5	0.2	0.3	0.2	0.2
Unigram	108606	0.9	2.3	0.2	1.8	0.3	0.1	0.5	0.1	0.2	1.0	0.2	0.3	0.1	0.1
Unigram′	179909	0.9	2.0	0.2	1.5	0.3	0.1	0.2	0.1	0.1	0.9	0.5	0.4	0.2	0.3
Unigram′ no spaces	108441	0.8	1.9	0.2	1.8	0.3	0.1	0.2	0.1	0.2	0.9	0.2	0.2	0.2	0.2

Table A.7: Standard deviations for finetuning results after pretraining for 30 epochs. Shown are deviations across 10 seeds. SB is short for Superbizarre.

	GLUE	Superbizarre Reddit		Superbizarre Arxiv	
		Dev	Test	Dev	Test
BPE	0.61	2.15e-05	1.34e-05	5.70e-15	5.26e-13
BPE′	2.7e-05	1.50e-16	5.26e-14	3.82e-17	8.61e-15
BPE′ no spaces	0.58	7.22e-14	9.04e-12	1.75e-15	5.54e-14
Unigram	0.36	2.27e-08	1.69e-06	5.15e-07	1.11e-07
Unigram′	6.0e-02	6.22e-04	7.83e-05	1.74e-05	6.05e-06

Table A.8: P values for Welch's t-test comparing Unigram′ no spaces to other models for fixed pretraining updates.

	GLUE	Superbizarre Reddit		Superbizarre Arxiv	
		Dev	Test	Dev	Test
BPE	0.41	1.47e-13	2.25e-13	2.77e-15	1.48e-13
BPE′	8.72e-05	1.19e-12	7.84e-16	3.46e-16	4.53e-14
BPE′ no spaces	0.41	1.28e-12	2.01e-15	1.21e-15	2.35E-12
Unigram	0.66	2.92e-08	1.19e-09	3.96e-09	8.78e-08
Unigram′	2.8e-06	1.45e-02	1.69e-06	1.55e-06	3.90e-04

Table A.9: P values for Welch's t-test comparing Unigram′ no spaces to other models for fixed pretraining epochs.

Appendix B

B.1 Morphological Alignment Results for English

In Table B.1, we show the performance of WordPiece vs WordPiece′ for the individual morphological datasets.

B.2 Training Scales

In Tables B.2 and B.3, we show full details of our training set-ups for pretraining the English and Finnish models, respectively. In Table B.4, we show the parameters used for each training scale.

B.3 Dataset Info

In Table B.5, we show information on all of the datasets we use for evaluation, including the size of the train and dev sets, the metric used for evaluation, and the domain.

B.4 Pretraining Losses

We show the pretraining loss curves for our English and Finnish models in Figures B.1 and B.2, respectively. We compare the pretraining losses averaged over the final 100 updates for all the English models, at all three scales, in Figure B.3a. We look at the pretraining evaluation accuracies for WordPiece′, and WordPiece′ extra loss in Figure B.3b.

	LADEC			MorphoLex			MorphyNet			DagoBERT			MEAN		
	Len	Precis.	F1	Len	Precis.	F1	Len	Precis.	F1	Len	Precis.	F1	Len	Precis.	F1
WordPiece	3.34	38.0	53.3	2.91	26.0	31.4	3.43	13.2	19.7	3.47	21.9	30.7	3.29	24.8	33.8
WordPiece′	2.66	**53.7**	**67.1**	2.55	**50.0**	**55.1**	2.95	**25.5**	**36.1**	2.85	**41.1**	**52.5**	2.75	**42.6**	**52.7**

Table B.1: Performance of WordPiece and WordPiece′ across four English morphological datasets, showing the average sequence length, precision and F1 score generated following the standard introduced by Creutz et al. (2004).

	Base Dataset	# Articles (M)	Examples (M)			Params (M)	Batch Size	# GPUs	Steps (k)	Epochs			Train Time (h)
			WP	WP'	WP' spaces					WP	WP'	WP' Spaces	
V Low	Wikipedia	0.1	1.2	1.1	1.8	5.8	1024	1	25	21.5	23.0	13.9	11.0
Low	Wikipedia	0.5	4.1	3.8	6.3	21.2	512	1	50	6.3	6.7	4.1	19.0
High	Wikipedia	6.5	19.8	18.5	30.2	98.2	256	1	400	5.2	5.5	3.4	29.2
V High	C4	40	88.0	82.2	-	370.4	128	4	400	2.3	2.5	-	70.9

Table B.2: The four training scales we use to evaluate our models in English.

	Base Dataset	# Articles (M)	Examples (M)		Params (M)	Batch Size	# GPUs	Steps (k)	Epochs		Train Time (h)
			WP	WP'					WP	WP'	
V Low	Wikipedia	0.1	0.3	0.3	5.8	1024	1	25	78.9	84.8	4.1
Low	Wikipedia	2	1.0	1.0	21.2	512	1	50	24.7	26.6	7.8
High	C4	10	37.6	34.6	98.2	256	1	400	1.4	1.5	78.4

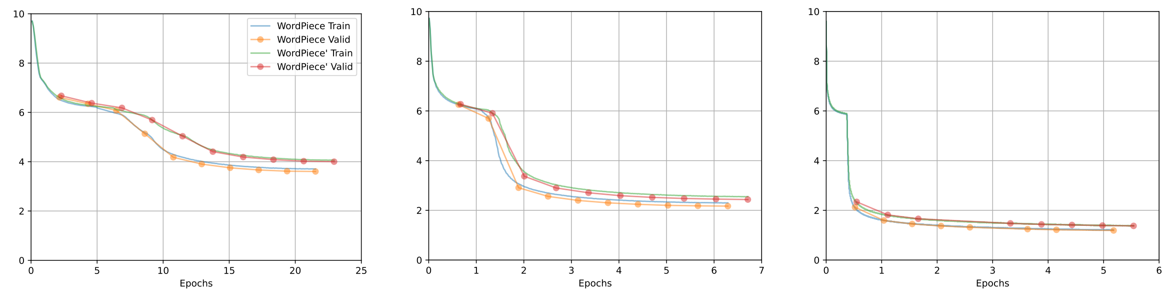
Table B.3: The three training scales we use to evaluate our models in Finnish.

	Layers	Att. Heads	Embed. Dim.
V Low	2	4	256
Low	4	8	512
High	12	12	768
V High	26	16	1024

Table B.4: Layers, attention heads, and embedding dimension for the four training scales.

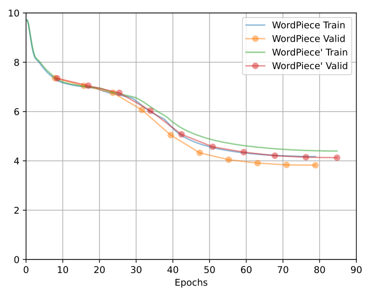
	|Train| (k)	|Dev| (k)	Metric	Domain
CoLA (Warstadt et al., 2019)	8.5	1	Matthew's Correlation	Books and Journal Articles
SST-2 (Socher et al., 2013)	67	1	Accuracy	Film Reviews
MRPC (Dolan and Brockett, 2005)	3.7	0.4	F1 / Accuracy	Online News
STS-B (Cer et al., 2017)	5.8	1.5	Pearson / Spearman Correlation	Various
QQP (`https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs`)	364	40	F1 / Accuracy	Quora questions
MNLI (Williams et al., 2018)	393	9.8	Accuracy	Various
QNLI (Rajpurkar et al., 2016)	105	5.5	Accuracy	Wikipedia
RTE (Bentivogli et al., 2009)	2.5	0.3	Accuracy	Wikipedia and News
Superbizarre-Arxiv (Hofmann et al., 2021)	58	19	F1	Arxiv Papers
Superbizarre-Reddit (Hofmann et al., 2021)	51	17	F1	Reddit
FLOTA (Hofmann et al., 2022)	1.2	0.4	F1	Arxiv Paper Titles
FiNER (Ruokolainen et al., 2020)	13.5	1.0	F1	Online News
Eduskunta (`https://github.com/aajanki/eduskunta-vkk`)	49.1	3.0	Accuracy	Parliamentary Questions
FinnSentiment (Lindén et al., 2023)	24.3	2.7	Accuracy	Social Media

Table B.5: Information for the datasets we use for evaluation.

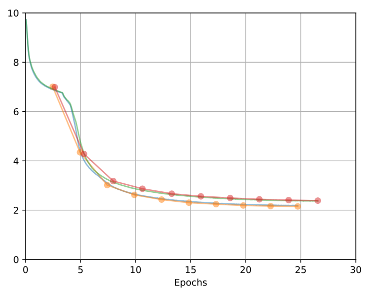


(a) V Low (b) Low (c) High

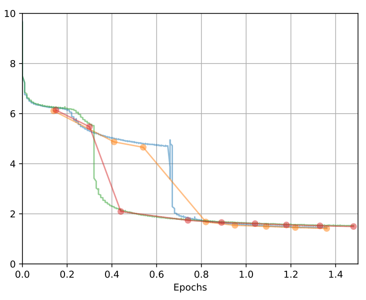
Figure B.1: Training and valid losses for WordPiece and WordPiece′ across three training scales for English.



(a) V Low

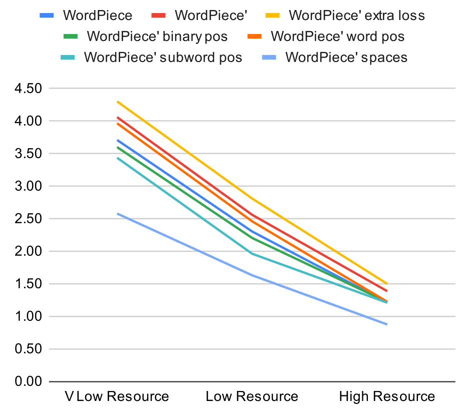


(b) Low

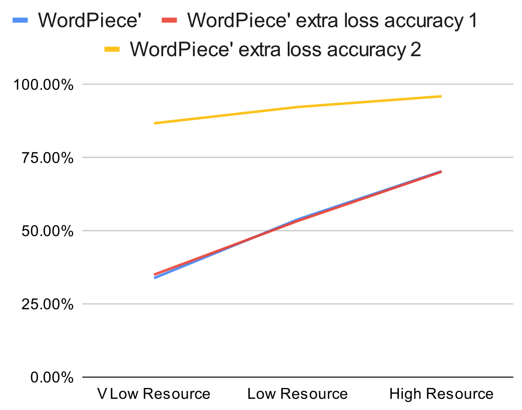


(c) High

Figure B.2: Training and valid losses for WordPiece and WordPiece′ across three training scales for Finnish.



(a) Pretraining MLM losses for all English models across three training scales, averaged across the last 100 steps.



(b) English pretraining evaluation accuracies for WordPiece′ and the two MLM heads for WordPiece′ extra loss.

Figure B.3: Comparison of MLM pretraining losses and evaluation accuracies.

B.5 Full Results

We show our full results for English and Finnish in Tables B.6 and B.7, respectively.

B.6 Standard Deviations

We show standard deviations in Tables B.8 to B.10.

B.7 Log Plots

We show all of the plots in the main paper, reproduced with a logarithmic training scale on the x-axis in Figures B.8 to B.12. The training scale K is defined as $K = x * p * e$, where x is the number of training examples, p is the number of parameters, and e is the number of training epochs.

										mnli				SB		FLOTA		
		conll	ncbi	wnut17	cola	sst2	mrpc	stsb	qqp	m	mm	qnli	rte	A	R	CS	M	P
V Low	WP	79.5	59.5	24.0	6.9	79.7	76.0	15.7	74.0	59.9	61.1	64.6	54.2	66.1	63.9	20.8	16.9	20.8
	WP′	80.2	60.4	20.4	3.5	80.8	76.2	15.8	77.4	63.8	65.1	67.3	56.0	68.3	65.4	23.0	23.1	24.7
	WP′ extra loss	80.1	61.3	23.0	6.7	80.6	75.6	16.0	76.7	63.3	64.9	65.7	55.3	68.4	65.4	21.6	20.3	28.7
	WP′ binary	79.7	60.8	22.6	7.4	82.7	76.0	15.2	74.7	62.5	63.6	63.6	55.4	68.3	65.4	24.2	21.8	27.4
	WP′ word pos	80.3	60.5	23.9	7.7	81.4	76.1	20.6	78.6	63.7	65.1	68.0	53.8	68.3	65.4	21.7	19.6	25.6
	WP′ subword pos	80.9	62.5	21.6	5.7	82.8	75.9	13.1	74.2	63.0	64.3	65.0	56.0	68.6	65.3	26.0	22.3	24.6
	WP′ spaces	78.0	58.8	20.4	7.3	80.6	76.5	14.4	75.6	62.0	62.6	64.9	53.6	67.8	65.3	20.0	19.9	23.2
	WP′ f/t binary	80.3	60.3	20.1	4.7	81.0	75.5	16.0	77.3	63.8	65.0	66.4	55.7	68.4	65.4	22.1	22.1	25.6
	WP′ f/t spaces	80.2	60.4	20.4	1.9	81.4	76.5	14.0	74.4	63.3	64.6	65.2	54.9	-	-	22.3	21.9	26.0
Low	WP	89.3	78.0	39.5	11.9	85.0	78.2	66.1	85.2	71.9	72.2	82.0	56.7	68.1	64.4	30.3	28.7	34.7
	WP′	89.9	77.0	37.3	16.9	85.3	79.0	78.0	85.6	72.2	72.7	81.6	56.6	69.3	66.0	44.1	38.7	46.6
	WP′ extra loss	90.7	77.6	39.3	15.5	84.6	77.3	75.2	85.2	72.6	73.1	81.2	56.3	69.2	65.9	46.3	41.0	48.1
	WP′ binary	89.9	77.5	37.2	18.5	87.1	77.0	76.3	85.3	73.4	73.4	82.7	57.3	69.1	66.0	44.6	41.7	47.3
	WP′ word pos	89.7	77.1	38.3	16.3	84.2	78.0	76.4	84.8	71.4	72.0	81.8	57.5	69.2	66.0	42.9	39.4	47.4
	WP′ subword pos	90.0	77.0	37.1	18.4	86.3	78.9	77.5	85.6	72.8	73.0	82.5	58.1	69.5	66.0	40.2	33.6	40.8
	WP′ spaces	89.1	76.3	37.5	16.3	84.3	76.0	74.3	85.0	72.0	72.2	80.0	58.1	69.2	65.7	43.2	40.4	47.0
	WP′ f/t binary	90.0	77.3	38.6	16.0	84.9	79.1	77.5	85.5	72.4	73.0	82.3	58.6	69.3	65.8	44.0	40.6	46.1
	WP′ f/t spaces	90.0	76.9	38.1	16.3	84.3	78.2	79.6	85.3	71.9	72.9	81.4	57.1	-	-	45.8	39.3	46.0
High	WP	95.0	83.7	52.1	34.7	90.0	87.2	85.6	88.9	80.3	80.4	89.1	65.1	69.5	65.2	51.6	47.6	52.0
	WP′	94.9	83.7	48.6	40.2	90.8	87.3	85.7	88.6	79.9	80.0	87.1	62.8	70.3	66.5	53.2	49.8	53.8
	WP′ extra loss	94.9	83.2	48.7	34.6	90.4	87.2	85.7	88.5	79.5	80.0	88.5	65.6	70.6	66.1	53.5	48.5	53.3
	WP′ binary	94.6	84.0	47.4	40.4	90.5	87.6	85.7	88.7	79.9	80.3	88.5	64.3	70.1	66.2	52.0	48.1	53.0
	WP′ word pos	94.6	83.1	48.5	40.0	90.7	88.2	86.3	88.7	80.5	80.4	88.2	66.1	70.3	66.5	51.9	49.0	54.7
	WP′ subword pos	94.4	83.7	48.1	38.4	90.4	86.5	85.7	88.8	80.4	80.6	87.9	63.8	70.0	66.4	47.3	46.3	51.0
	WP′ spaces	93.8	83.8	44.6	38.0	90.2	86.6	84.9	88.3	79.3	79.5	86.4	64.3	70.3	66.2	54.0	49.6	53.2
	WP′ f/t binary	94.8	83.3	48.2	39.0	90.9	87.1	85.8	88.5	79.9	80.0	87.0	61.9	70.1	66.0	51.6	51.6	54.7
	WP′ f/t WB tokens	94.9	83.7	48.6	36.3	90.5	87.1	85.7	88.4	80.1	80.6	86.7	63.3	-	-	52.9	49.9	54.7
V High	WP	95.6	86.1	62.9	61.3	92.3	89.2	89.0	89.9	85.6	85.7	91.2	63.9	70.6	66.5	59.2	51.4	54.3
	WP′	95.7	86.5	62.2	61.3	93.1	90.9	89.4	90.0	85.2	85.3	90.9	67.1	71.6	67.4	60.4	49.4	55.6

Table B.6: Full English results across all datasets, training scales, and models.

		FiNER	Eduskunta	FinnSentiment
V Low	WP	72.2	64.6	81.6
	WP′	73.0	65.2	80.9
Low	WP	84.2	71.3	86.3
	WP′	85.0	71.1	86.8
High	WP	89.9	75.9	91.3
	WP′	89.8	75.3	92.9

Table B.7: Full Finnish results across all datasets, training scales, and models.

	GLUE				NER				Superbizarre				FLOTA			
	V Low	Low	High	V High	V Low	Low	High	V High	V Low	Low	High	V High	V Low	Low	High	V High
WordPiece	0.6	1.5	0.4	0.4	0.5	0.4	0.3	0.4	0.1	0.1	0.1	0.1	0.8	3.7	0.7	1.1
WordPiece′	0.4	0.5	0.2	1.1	0.6	0.5	0.2	0.4	0.1	0.1	0.3	0.2	0.4	0.2	0.5	1.0

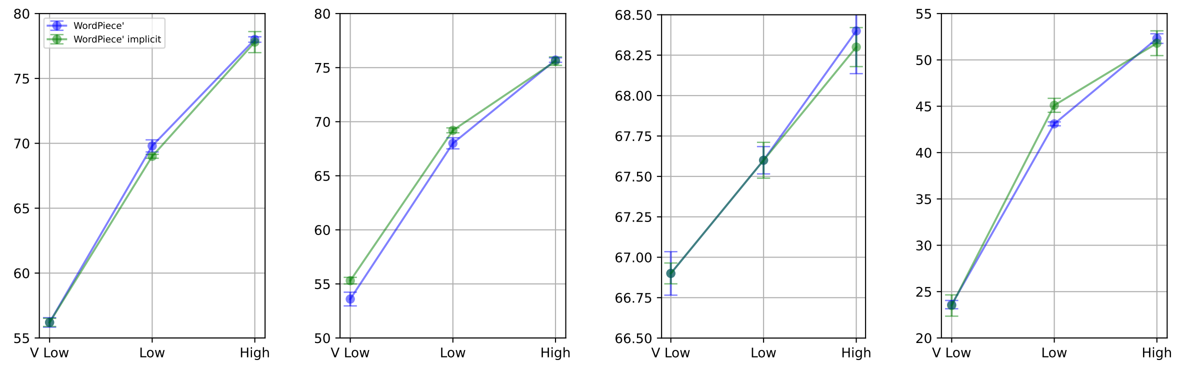
Table B.8: Standard deviations for the English results across the four tasks and training scales for WordPiece and WordPiece′.

	GLUE			NER			Superbizarre			FLOTA		
	V Low	Low	High	V Low	Low	High	V Low	Low	High	V Low	Low	High
WordPiece′	0.4	0.5	0.2	0.6	0.5	0.2	0.1	0.1	0.3	0.4	0.2	0.5
WordPiece′ implicit	0.3	0.2	0.8	0.3	0.2	0.4	0.1	0.1	0.1	1.1	0.8	1.3
WordPiece′ explicit binary	0.4	0.2	0.5	0.4	0.8	0.4	0.1	0.1	0.1	1.7	0.9	0.6
WordPiece′ explicit word	0.4	0.1	0.3	0.3	0.3	0.4	0.1	0.1	0.1	0.7	1.0	2.7
WordPiece′ explicit subword	0.6	0.2	0.4	0.3	0.4	0.3	0.1	0.2	0.2	1.2	4.9	2.8
WordPiece′ explicit WB tokens	0.6	0.2	2.0	0.5	0.2	0.2	0.1	0.1	0.2	1.1	0.2	0.1
WordPiece′ explicit f/t WB tokens	0.2	0.3	0.5	0.6	0.4	0.2	-	-	-	1.5	0.7	0.8
WordPiece′ explicit f/t binary	0.6	0.4	0.4	0.3	0.5	0.4	0.1	0.1	0.4	1.2	1.5	1.3

Table B.9: Standard deviations for the English results across the four tasks and three training scales for WordPiece′ and the modified architectures which include word boundary information.

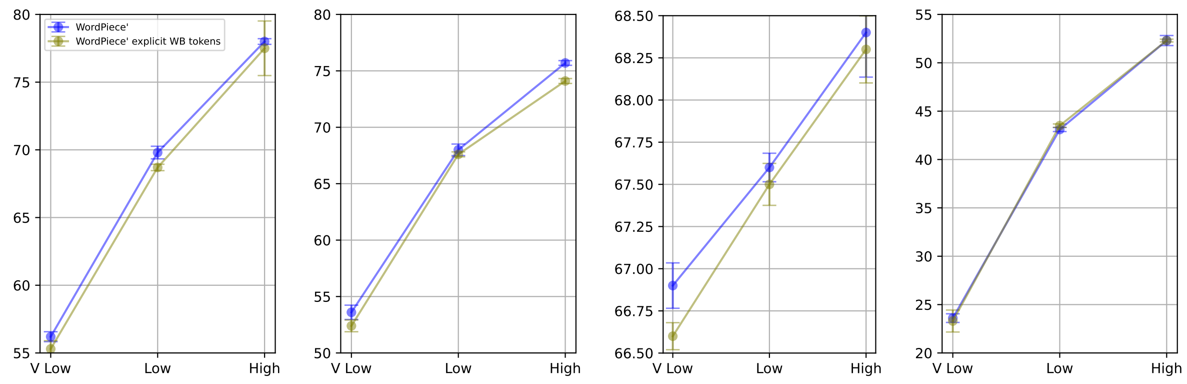
	NER			SeqClass		
	V Low	Low	High	V Low	Low	High
WordPiece	0.2	0.6	0.3	0.2	0.3	0.2
WordPiece′	0.6	0.4	0.2	0.6	0.5	0.3

Table B.10: Standard deviations for the Finnish results across the three tasks and training scales for WordPiece and WordPiece′.



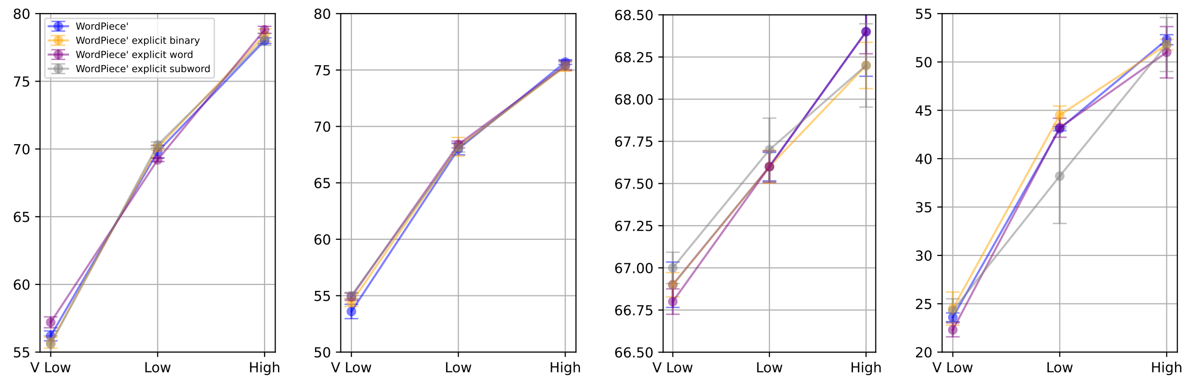
(a) GLUE (b) NER (c) Superbizarre (d) FLOTA

Figure B.4: Results for WordPiece′ and WordPiece′ implicit.



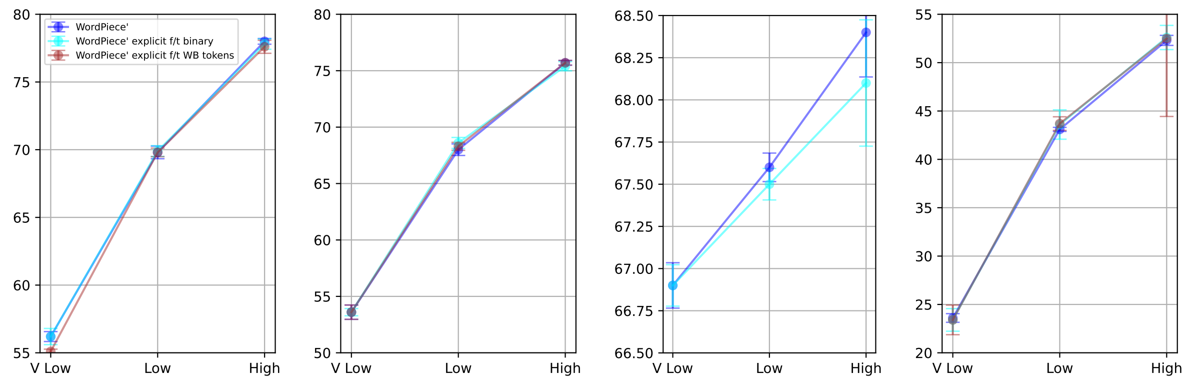
(a) GLUE (b) NER (c) Superbizarre (d) FLOTA

Figure B.5: Results for WordPiece′ and WordPiece′ explicit with word boundary tokens.



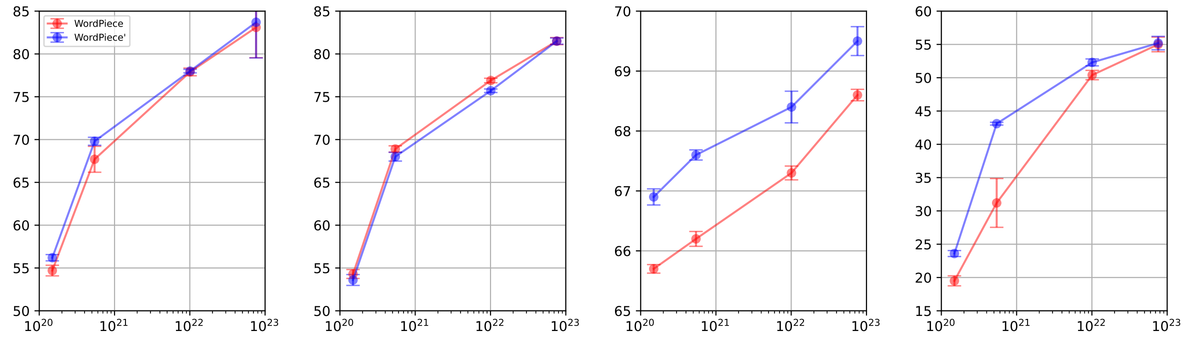
(a) GLUE (b) NER (c) Superbizarre (d) FLOTA

Figure B.6: Results for WordPiece′ and WordPiece′ explicit with word boundary embeddings.



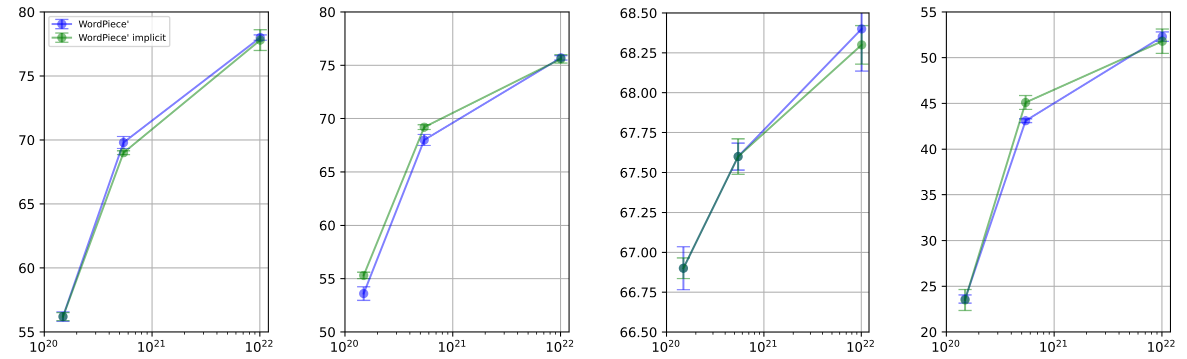
(a) GLUE (b) NER (c) Superbizarre (d) FLOTA

Figure B.7: Results for WordPiece′ and WordPiece′ finetuned with either word boundary tokens or binary index word boundary embeddings.



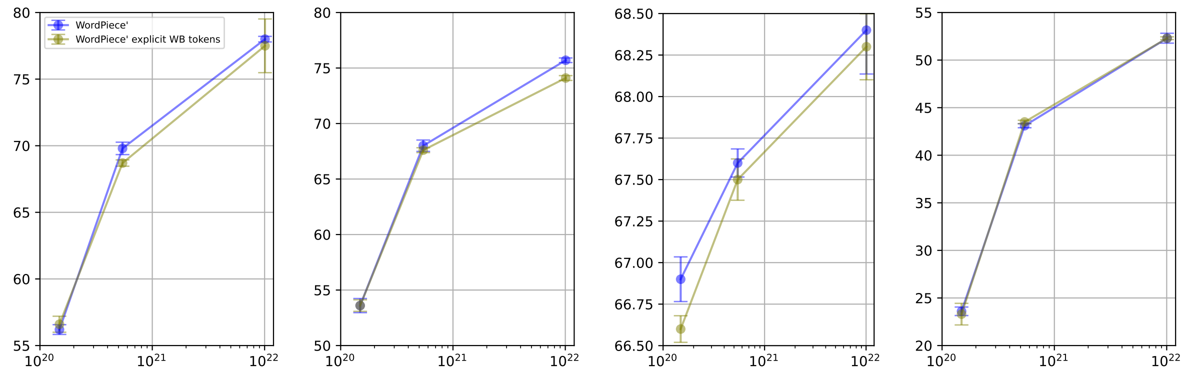
(a) GLUE (b) NER (c) Superbizarre (d) FLOTA

Figure B.8: Results for WordPiece and WordPiece′ with log training scale on the x-axis.



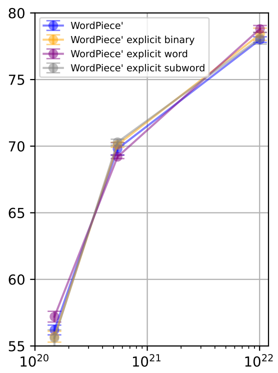
(a) GLUE (b) NER (c) Superbizarre (d) FLOTA

Figure B.9: Results for WordPiece′ and WordPiece′ implicit with log training scale on the x-axis.

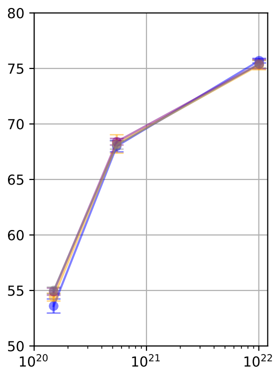


(a) GLUE (b) NER (c) Superbizarre (d) FLOTA

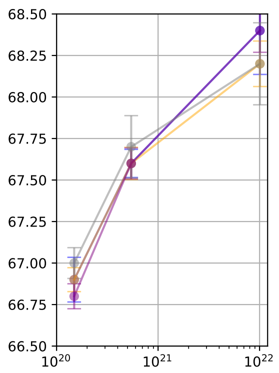
Figure B.10: Results for WordPiece′ and WordPiece′ explicit with word boundary tokens with log training scale on the x-axis.



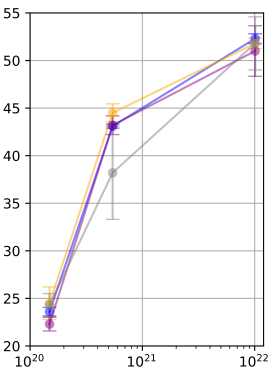
(a) GLUE



(b) NER

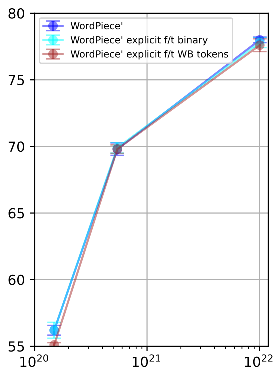


(c) Superbizarre

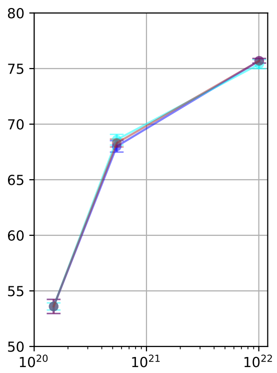


(d) FLOTA

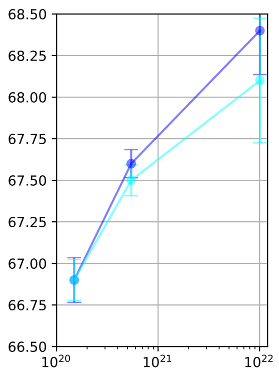
Figure B.11: Results for WordPiece′ and WordPiece′ explicit with word boundary embeddings with log training scale on the x-axis.



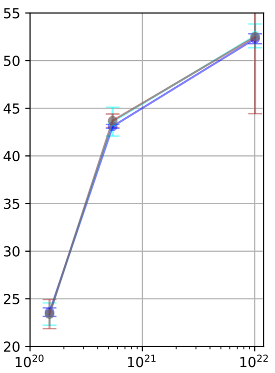
(a) GLUE



(b) NER



(c) Superbizarre



(d) FLOTA

Figure B.12: Results for WordPiece′ and WordPiece′ finetuned with either word boundary tokens or binary index word boundary embeddings with log training scale on the x-axis.

Appendix C

C.1 Morphological Complexities

The range of morphological complexities in the four languages of mCWIF has been shown in prior work. Bentz et al. (2016) used WALS (Dryer and Haspelmath, 2013) features to determine morphological complexity, giving scores of 0.78, 0.40, and 0.33 for Turkish, German, and English, respectively. Juola (1998) used Kolmogorov complexity as a measure of morphological complexity, giving a R/C value of 1.12 for Finnish (highest) and 0.97 for English (second lowest). Kettunen (2014) recalculated this metric across the EU constitution, including German, giving 1.17 for German (highest), 1.16 for Finnish (second highest), and 1.05 for English (joint lowest).

C.2 Word Counts

In Table C.1, we show the number of words occurring per language. There is significantly less data available for English, which might either indicate a limitation of the Common Crawl data, or a lower popularity of forums in English compared to other languages: Reddit may replace the function of these sites in English. The Indonesian and Swahili counts are much larger than the other languages, since the forums we start with are amongst the largest (by number of pages) in Common Crawl—we choose general forums and then take their subtopics, rather than taking forums with specific topics, as in the other languages.

C.3 mCWIF for Indonesian and Swahili

We hope that our dataset provides the groundwork for extending the evaluation of tokenisers on language model performance, in particular when handling complex words, to a multilingual setting. Our procedure can be replicated with any language where internet forum data is sufficient. We investigated whether we could extend our dataset to Indonesian and Swahili, two morphologically complex languages from different families

	# Words	
	Total	Valid
English	17 889 574	14 955 926
German	242 826 009	187 559 157
Turkish	97 559 513	71 525 196
Finnish	20 708 985	14 381 304
Indonesian	111 045 172	94 731 780
Swahili	513 370 204	444 789 032

Table C.1: Counts showing the number of words for each language, both with and without filtering for "valid" words.

	Forum URL	**Topic**
Indonesian	pengetahuan-penting-penunjang-kesehatan	Health
	pc-games-online	PC Games
	flora-dan-fauna	Flora and Fauna
	make-money-online	Money
	soccer	Football
Swahili	entertainment	Entertainment
	jamii-health-jukwaa-la-afya	Health
	tech-gadgets-science-forum	Technology
	jukwaa-la-elimu-education-forum	Education
	jamii-sports	Sports

Table C.2: Forums from which the datasets were created for Indonesian and Swahili. Each forum provides the classification label in the dataset. The base URLs are forum.idws.id/forums/ for Indonesian, and jamiiforums.com/forums/ for Swahili

to those already included. Due to the more limited data in these languages, rather than looking for five different forums, we instead took five sub-areas from the biggest forums in each language: forum.idws.id/forums/ for Indonesian, and jamiiforums.com/forums/ for Swahili. However, upon inspection of the final dataset we found that a high proportion of examples were not evidently semantically related to the sub-area, which requires further investigation. In Table C.2, we show the forums that we use to extract data for Indonesian and Swahili in an attempt to extend our dataset to these languages.

C.4 Pretraining

In this section, we describe our pretraining approach for the Turkish and German models. Both of these models are trained across three training scales: Low, V Low, and High. We show the resource set-ups in Table C.3. The V Low and Low training scales take their training data from Wikipedia, and the Low training scales take all available Wikipedia articles in the respective languages. The High training scales take data from the mC4 dataset. We train our models in the manner of RoBERTa (Liu et al., 2019), which, in comparison to BERT (Devlin et al., 2019), performs dynamic masking, and doesn't use the next sentence prediction task. We mask 15% of tokens. Across all set-ups, we linearly warmup the learning rate to a maximum value of 1e-4, and then linearly decay to 0. We use the Adam optimiser (Kingma and Ba, 2015), and a sequence length of 256. All training is performed on A100 or H100 GPUs, and implemented in Hugging Face (Wolf et al., 2020).

	Articles (M)		Params (M)	Batch Size	Steps (k)
	Tr	De			
V Low	0.1	0.1	5.8	1024	25
Low	0.6	2.9	21.2	512	50
High	10	10	98.2	256	400

Table C.3: The three training scales we use to pretrain our models for Turkish and German.

C.5 Prompts

Here, we provide the prompts that we pass to GPT-4o using the OpenAI API, both for the dataset filtering and for the experiments. For English, our prompts for typo filtering are given in Table C.4, and our prompts for toxicity filtering are given in Table C.5. For zero-shot prompting, we show our prompts in Table C.6. We pass max tokens as 300, and set the temperature at 0.

Role	**Value**
System	`You are a helpful assistant that identifies words containing typos, proper nouns, potential usernames, and words that are not English. You are given a list of English words, and return only the words which match those criteria.`
Content	`Check the following list of English words:{batch}. Only return words with spelling errors (typos), words that are proper nouns (including usernames), or words that are not English. Non-standard words are allowed, do not return them. Return only words, nothing else.`

Table C.4: The prompts used for typo filtering.

Role	**Value**
System	`You are an assistant that identifies offensive English words.`
Content	`Check the following list of English words and return any that are offensive:{batch}. Return words that are profane, derogatory, discriminatory, or otherwise offensive. Return only words, nothing else.`

Table C.5: The prompts used for toxicity filtering.

Role	**Value**
System	`You are a helpful assistant trained to classify words into the forums they come from. You will be given a list of possible forums, along with their descriptions, and you will be given a word. You return only the name of the forum that the word comes from, nothing else.`
Content	`Classify the given word into one of these forums: phoenixrising, cosmoquest, bikeforums, hairsite, avforums. The forum descriptions are: phoenixrising - chronic illness, cosmoquest - astronomy, bikeforums - bikes, hairsite - hair regeneration, avforums - home entertainment. Only return the forum name, do not return any other text. Word: {word}`

Table C.6: The prompts used for our zero-shot GPT-4o baseline.

	Forum	String	Example	WordPiece	WordPiece′
English	comosquest	tions	separations	separ ##ations	separation s
			glaciations	gl ##ac ##iations	glac iations
	phoenixrising	neuro	neurovirulence	ne ##uro ##vir ##ule ##n ##ce	neuro vir ule n ce
			neurosurgery	ne ##uro ##s ##urg ##ery	neuro surgery
	avforums	board	plasterboard	pl ##aster ##board	plas ter board
			plasterboards	pl ##aster ##board ##s	plas ter boards
German	bauexperten	stric	estrichdicke	est ##rich ##di ##cke	est rich dick e
			sichtestrich	sicht ##est ##rich	sicht est rich
	imker	biene	bienenkästen	bie ##nen ##k ##äste ##n	bie nen kä sten
			sommerbienen	so ##m ##mer ##bie ##nen	so mmer bie nen
	modellbahn	gleis	aufstellgleis	auf ##stell ##gleis	auf stell gleis
			gleisplanvorschläge	gl ##eis ##plan ##vor ##schläge	gleis plan vor schläge
Turkish	bordomavi	rimiz	maçlarimizi	maç ##lar ##imiz ##i	maçlar imiz i
			taraftarimiza	taraftar ##imiz ##a	taraftar imiz a
	risale	etler	lezzetlerin	le ##z ##ze ##tle ##rin	lez zet lerin
			ibadetlerinin	ib ##adet ##lerinin	ibadet lerinin
	keyfimuzik	klavy	klavyelerinizde	k ##lav ##ye ##lerini ##z ##de	klav yel eri niz de
			klavyemizi	k ##lav ##ye ##mi ##zi	klav yem izi
Finnish	vau	raska	raskaudestani	raska ##udesta ##ni	raska udesta ni
			alkuraskaudesta	alku ##ras ##kau ##desta	alku raska udesta
	fillari	pyörä	pyöräilijää	pyörä ##ili ##jää	pyörä ilijä ä
			maastopyörässä	maasto ##pyörä ##ssä	maasto pyörä ssä
	ylikerroin	maali	vähämaalisia	vähä ##maali ##sia	vähä maalis ia
			ykkösmaalivahti	ykkös ##maali ##vahti	ykkös maalivahti

Table C.7: The most disproportionately frequent substrings of length five, for three forums per language, along with two examples containing this substring. We also show the tokenisations generated by WordPiece and WordPiece′.

C.6 Tokenisations

In Table C.7, we show the most disproportionately frequent substrings of length 5 for 3 forums per language, along with two examples of words containing those substrings, and their tokenisations by WordPiece and WordPiece′.

C.7 Full Results

We provide the full results of all of our experiments on mCWIF Small in Table C.8, and on mCWIF Large in Table C.10, along with the standard errors in Tables C.9 and C.11.

	English				German				Turkish				Finnish			
	WordPiece		WordPiece′		WordPiece		WordPiece′		WordPiece		WordPiece′		WordPiece		WordPiece′	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test
V Low	36.8	37.3	43.5	40.3	47.9	45.3	57.7	54.6	63.4	65.7	61.4	60.3	67.4	64.4	69.1	66.8
Low	46.3	43.6	54.5	46.9	57.9	55.3	68.7	66.8	72.8	74.7	73.9	75.2	79.4	80.2	81.6	79.6
High	54.8	47.7	59.7	45.0	72.4	65.1	76.7	76.5	83.9	87.0	83.4	86.3	81.9	81.5	86.4	83.1
V High	57.1	48.4	67.6	54.0												

Table C.8: Results for all models across mCWIF Small.

	English				German				Turkish				Finnish			
	WordPiece		WordPiece′		WordPiece		WordPiece′		WordPiece		WordPiece′		WordPiece		WordPiece′	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test
V Low	1.1	1.4	0.7	1.0	1.0	1.3	0.9	0.6	0.6	0.5	0.4	0.6	0.6	1.0	0.4	0.7
Low	0.8	0.7	0.7	0.8	0.8	1.7	1.2	1.0	0.3	0.6	0.4	0.5	0.3	0.4	0.8	0.8
High	0.8	1.0	1.1	0.9	1.1	1.1	0.3	0.6	0.4	0.7	0.4	0.5	0.4	0.4	0.6	0.5
V High	1.1	1.2	0.6	0.8												

Table C.9: Standard errors for all models across mCWIF Small.

	German				Turkish				Finnish			
	WordPiece		WordPiece′		WordPiece		WordPiece′		WordPiece		WordPiece′	
	Eval F1	Test F1	Eval F1	Test F1	Eval F1	Test F1	Eval F1	Test F1	Eval F1	Test F1	Eval F1	Test F1
V Low	68.8	68.2	76.9	74.5	60.1	59.6	62.8	60.7	67.7	63.2	68.5	66.6
Low	77.5	76.7	81.3	77.9	69.2	67.7	70.3	68.1	74.5	69.9	74.7	71.4
High	81.1	80.8	84.7	82.9	77.9	73.9	78.3	75.1	78.5	73.6	80.7	76.4

Table C.10: Results for all models across mCWIF Large.

	German				Turkish				Finnish			
	WordPiece		WordPiece′		WordPiece		WordPiece′		WordPiece		WordPiece′	
	Eval F1	Test F1	Eval F1	Test F1	Eval F1	Test F1	Eval F1	Test F1	Eval F1	Test F1	Eval F1	Test F1
V Low	0.3	0.2	0.1	0.3	0.1	0.4	0.1	0.2	0.2	0.2	0.1	0.1
Low	0.4	0.2	0.2	0.3	0.2	0.3	0.1	0.2	0.2	0.4	0.1	0.3
High	0.2	0.2	0.3	0.4	0.2	0.3	0.2	0.4	0.1	0.3	0.1	0.5

Table C.11: Standard errors for all models across mCWIF Large.