UNIVERSITY OF LEEDS

Exploring Syntactic Representations in Pre-trained Transformers to Improve Neural Machine Translation by a Fusion of Neural Network Architectures

Yuqian Dai

Submitted in accordance with the requirements for the degree of Doctor of Philosophy

The University of Leeds

Faculty of Arts, Humanities and Cultures

School of Languages Cultures and Societies

Dec 2023

Abstract

Neural networks in Machine Translation (MT) engines may not consider deep linguistic knowledge, often resulting in low-quality translations. In order to improve translation quality, this study examines the feasibility of fusing two data augmentation strategies: the explicit syntactic knowledge incorporation and the pre-trained language model BERT.

The study first investigates what BERT knows about syntactic knowledge of the source language sentences before and after MT fine-tuning through syntactic probing experiments, as well as using a Quality Estimation (QE) model and the chi-square test to clarify the correlation between syntactic knowledge of the source language sentences and the quality of translations in the target language. The experimental results show that BERT can explicitly predict different types of dependency relations in source language sentences and exhibit different learning trends, which probes can reveal. Moreover, experiments confirm a correlation between dependency relations in source language sentences and translation quality in MT scenarios, which can somewhat influence translation quality. The dependency relations of the source language sentences frequently appear in low-quality translations are detected. Probes can be linked to those dependency relations, where prediction scores of dependency relations tend to be higher in the middle layer of BERT than those in the top layer.

The study then presents dependency relation prediction experiments to examine what a Graph Attention Network (GAT) learns syntactic dependencies and investigates how it learns such knowledge by different pairs of the number of attention heads and model layers. Additionally, the study examines the potential of incorporating GAT-based syntactic predictions in MT scenarios by comparing GAT with fine-tuned BERT in dependency relations prediction. Based on the paired t-test and prediction scores, GAT outperforms MT-B, a version of BERT specifically fine-tuned for MT. GAT exhibits higher prediction scores for the majority of dependency rela-

tions. For some dependency relations, it even outperforms UD-B, a version of BERT specifically fine-tuned for syntactic dependencies. However, GAT faces difficulties in predicting accurately by the quantity and subtype of dependency relations, which can lead to lower prediction scores.

Finally, the study proposes a novel MT architecture of Syntactic knowledge via Graph attention with BERT (SGB) engines and examines how the translation quality changes from various perspectives. The experimental results indicate that the SGB engines can improve low-quality translations across different source language sentence lengths and better recognize the syntactic structure defined by dependency relations of source language sentences based on the QE scores. However, improving translation quality relies on BERT correctly modeling the source language sentences. Otherwise, the syntactic knowledge on the graphs is of limited impact. The prediction scores of GAT for dependency relations can also be linked to improved translation quality. GAT allows some layers of BERT to reconsider the syntactic structures of the source language sentences. Using XLM-R instead of BERT still results in improved translation quality, indicating the efficiency of syntactic knowledge on graphs. These experiments not only show the effectiveness of the proposed strategies but also provide explanations, which bring more inspiration for future fusion that graph neural network modeling linguistic knowledge and pre-trained language models in MT scenarios.

Intellectual Property

The candidate confirms that the work submitted is his own, except where work which has formed part of jointly authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

Chapter 3 is based on work from a jointly authored publication:

Dai, Yuqian, Marc de Kamps, and Serge Sharoff. "BERTology for Machine Translation: What BERT Knows about Linguistic Difficulties for Translation." Proceedings of the Thirteenth Language Resources and Evaluation Conference. 2022.

Signed 

Acknowledgements

I would like to express my gratitude towards my supervisors, Professor Serge Sharoff and Dr. Marc de Kamps, for their invaluable supervision, support, and guidance throughout my Ph.D. studies. Even though English is not my first language, my supervisors I have worked with have displayed great patience and generosity in sharing their extensive knowledge and experience to support my academic research.

Many thanks to Professor Eric Atwell and Dr. Callum Walker for their research advice and feedback on the First Formal Progress Report (FFPR). I would also like to thank the University of Leeds for their academic and technical support for my research. It is their help and support that has made my study and life in the UK a great time.

I also want to express my gratitude to my parents and my girlfriend. My parents have shown consistent respect and support for my Ph.D. studies in the UK, as they are aware of my research interests and desire to expand my knowledge. Despite her busy schedule, my girlfriend Yingyi Iu, who also graduated from the University of Leeds with a Master's degree, remains a constant source of support and encouragement for me. I am grateful for their trust in me, which has kept me motivated and uplifted during my studies. Their understanding and encouragement over the past few years have been invaluable to my success.

List of Abbreviations

H_0 Null Hypothesis.

H_1 Alternative Hypothesis.

AE Auto Encoder.

AGCNN Adaptive Graph Convolution Neural Network.

ASTGCN attention based spatial-temporal Graph Convolutional Network.

BERT Bidirectional Encoder Representations from Transformers.

BLEU BiLingual Evaluation Understudy.

CNN Convolutional Neural Network.

CV Computer Vision.

De German.

EBMT Example-Based Machine Translation.

ELMo Embeddings from Language Models.

En English.

Europarl European Parliament Proceedings Parallel Corpus.

GAE Graph Auto-Encoder.

GAN Generative Adversarial Network.

GAT Graph Attention Network.

GCN Graph Convolutional Neural Network.

GloVe Global Vectors for Word Representation.

GNN Graph Neural Network.

GPT-1 Generative Pre-trained Transformer-1.

GPT-2 Generative Pre-trained Transformer-2.

GPU Graphics Processing Unit.

GraphSAGE Graph Sample and Aggregate.

GRU Gated Recurrent Unit.

GSD Universal Dependencies GSD.

K Key.

L Long-length.

LSTM Long Short-term Memory.

M Medium-length.

mBERT Multilingual Bidirectional Encoder Representations from Transformers.

MLM Masked Language Model.

MLP Multi-layer Perceptron.

MT Machine Translation.

MWE Multi-Word Expression.

NLG Natural Language Generation.

NLP Natural Language Processing.

NLU Natural Language Understanding.

NMT Neural Machine Translation.

NSP Next Sentence Prediction.

Pascal Parent-scaled self-attention.

PixEL Pixel-based Encoder of Language.

PUD Parallel Universal Dependencies.

Q Query.

QE Quality Estimation.

RBMT Rule-Based Machine Translation.

ReLU Rectified Linear Unit.

RGSE Recurrent Graph Syntax Encoder.

RNN Recurrent Neural Network.

RSA Representational Similarity Analysis.

RTD Replaced Token Detection.

Ru Russian.

S Short-length.

Sen L Sentence Length.

SGB Syntactic knowledge via Graph attention with BERT.

SGBC Syntactic knowledge via Graph attention with BERT Concatenation.

SGBD Syntactic knowledge via Graph attention with BERT and Decoder.

SMT Statistical Machine Translation.

UD Universal Dependencies.

UNPC United Nations Parallel Corpus.

V Value.

XLM Cross-lingual Language Model.

XLM-R XLM-RoBERTa.

Zh Chinese.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Background

Machine Translation (MT) refers to the process of automatically translating text from one natural language to another while retaining the same meaning. It is an interdisciplinary subject that involves linguistics, computer science, and mathematics. MT is also considered one of the crucial tasks of Natural Language Processing (NLP), which is closely related to the disciplines of Natural Language Generation (NLG) and Natural Language Understanding (NLU). With the internet constantly growing and big data becoming more common, the need for processing multilingual information is on the rise. People are becoming more willing to use MT to overcome language barriers. However, MT is not solely dependent on the hardware performance of computers, but human expertise in linguistics and psychology is also necessary. Despite the increasing demand, creating an automated, high-quality MT engine remains a significant challenge.

The idea of a universal language dates back to the 17th century when the concept of MT first emerged (Hutchins, 2007). There has been much debate about the range of applications for computers since the first generation is introduced. Many researchers have been working on creating an MT engine after being inspired by the idea that computers might be able to translate between different languages. Therefore, translation technology has undergone significant advancements, progressing from older Rule-Based Machine Translation (RBMT) and Example-Based Machine Translation (EBMT) to the more modern Statistical Machine Translation (SMT), where these iterations have notably enhanced the overall quality of translations. Many commercial organiza-

tions also have developed online translation tools that utilize SMT technology. These tools are designed to overcome language barriers, promote international trade, and encourage cultural exchange. They provide text, image, and language translation services, as well as facilitate cross-domain communication through translation.

The adoption of Graphics Processing Unit (GPU) and the availability of parallel corpus resources have facilitated the development of deep learning-based Neural Machine Translation (NMT) (Kalchbrenner & Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014), which has become the most popular paradigm for MT tasks replacing the traditional SMT paradigm. The majority of NMT is built on an encoder-decoder framework, which simplifies the model structure and training process. The attention mechanism proposed for the encoder-decoder framework allows NMT to capture dynamic alignment information (Bahdanau et al., 2014), where the MT engine can take full advantage of the contextual information on the source sentence side and more flexibly select the translation information on the target sentence side to achieve a more fluent translation result. Researchers have also conducted research on and made enhancements to MT models in several ways, including addressing the issue of out-of-vocabulary words (Jean et al., 2014), discovering more effective model training algorithms (Shen et al., 2015), and integrating linguistic information (S. Wu et al., 2017a). The success of NMT is similar to the success of other related aspects of deep learning: complicated neural network structures, large amounts of data, and efficient computational resources are the prerequisites for deep learning to exploit its data-fitting capabilities as much as possible and achieve breakthroughs.

Such characteristics also reflect that the performance of NMT is not only related to the model structure but also depends on the quality, data size, and domain coverage of the parallel corpus. Many languages around the world lack the necessary high-quality, large-scale, broad-coverage bilingual corpus needed for effective NMT training, except for rich-resource languages such as Chinese, English, German, and Russian, which have plenty of resources available. Poor translation quality often occurs when NMT is used in low-resource language and out-of-domain translation scenarios. A basic approach is to use manual annotation to create a scenario-specific, high-quality parallel corpus. This approach, however, is expensive in terms of labor and time. An alternative is to utilize the internet to collect a vast parallel corpus. However, the bilingual corpus acquired through this method may be plagued with misspelling of words and mistranslations between corresponding sentences. Besides, most of such a corpus would be

from the news domain, which has limited data diversity. As a result, the amount of bilingual data in MT scenarios is often limited. There is still a need for discussion on how to improve the effectiveness of an MT engine in recognizing the structure and features of input sentences, which could lead to an improvement in the quality of translations in scenarios with a limited amount of bilingual data.

1.2 Research Motivation

NMT has become a dominant approach and paradigm in MT research and applications. Despite the existence of bilingual corpuses as the training set, the diversity of languages and the constant renewal of vocabularies still limit the performance of MT engines, such as out-of-vocabulary, poor cross-domain adaptation, and insufficient utilization of prior linguistic knowledge (Koehn & Knowles, 2017; Y. Liu, 2017). Currently, one approach to alleviate this is to make full use of the available bilingual data in MT tasks. Data augmentation is a strategy to increase the features of training data, which can be accomplished by either extracting new features from existing data or by creating new pseudo-data. It has been successfully applied to Computer Vision (CV) (Cubuk et al., 2019; S. G. Müller & Hutter, 2021) and NLP (Mallinson et al., 2017; Wieting et al., 2017). By increasing the number of features in the training data, the neural network becomes more robust and can learn more potential information. Moreover, this approach makes the limited training data more valuable and reduces the reliance of the model on a single image or text feature, thus preventing overfitting.

1.2.1 Syntactic Knowledge

Both humans and machines suffer from comprehension confusion caused by sentence ambiguity and unclear expressions when the syntactic structure of the sentence or contextual information is not provided. For a given English sentence: *Lucy insisted on a talk with the manager herself.* There are two possible interpretations of this sentence: *Lucy wanted to talk with the manager* or *Lucy herself wanted to talk with the manager.* Another scenario is that some words are used in informal situations or internet buzzwords, which may not be officially defined in dictionaries but are used in casual writing, such as *nahmsayin*, and *wassup.* A more common sentence expression for them should be: *do you know what I am saying* and *what is up*, where the NMT engines (DeepL and Google Translate) do not effectively recognize them in translation into Chinese.

Since the bilingual corpus used as the training set tends to be drawn from more formal contexts such as news and conferences, such informal words or phrases do not appear in the training set, which could cause NMT to be unable to accurately determine the sentence structure and lexicality thus obtain a non-fluent translation.

The way humans use language follows a certain structure, which is shown in how words are put together. Linguistics refers to this as syntactic knowledge, the understanding of how words are combined to create sentences, phrases, and utterances. It focuses on the manner in which language is structured, including the arrangement of words and the formation of sentences (S. Müller, 2016). Syntactic knowledge can help an MT engine to better learn the sentence structure in the training set, and it can also provide explicit sentence structure information to guide the MT engine in translation when dealing with out-of-vocabulary and out-of-domain sentences. Therefore, it can improve the generalization ability of the MT engine and corpus utilization efficiency, leading to better quality translations to a certain extent. In this study, incorporating explicit syntactic knowledge (usually from the parser or gold annotated corpus) in MT scenarios is considered a means of data augmentation. Although explicit syntactic knowledge does not transform and expand the original data to generate more pseudo-data, it provides additional explicit syntactic information on the bilingual data to help the MT engine better understand the input sentence structure and linguistic rules to maximize data utilization.

In most current NMT engines, important linguistic and structural information in the sentence is not explicitly modeled since the source and target sentences are treated as sequential strings, which the encoder reads as sequences, and the decoder generates translations word by word. Shi et al. (2016) find that sequence-to-sequence models can learn some implicit source sentence syntax from a sentence-aligned parallel corpus. However, it still cannot capture many deep structural details. Inspired by studies in which explicit syntactic knowledge is widely used in SMT, incorporating explicit syntactic knowledge in NMT has become a popular topic in research, as it can help alleviate the challenges of limited bilingual data and improve translation quality. Eriguchi et al. (2016) propose a tree-to-sequence attention mechanism NMT engine, in which the head-driven phrase structure grammar encodes the source language from the bottom up to obtain the structural information. J. Li et al. (2017) transform syntactic trees into syntactic tags and then mix them with words into the same linearized sequence, which has the advantage of avoiding the complex network structure of trees. Bastings et al. (2017) use a

Graph Convolutional Neural Network (GCN) to encode the dependency structure of the source language sentences. They use the hidden vectors encoded by the Convolutional Neural Network (CNN) as input and perform graph learning on the hidden vectors by the dependency structure to generate vectors containing syntactic information for each token.

The subsequently proposed Transformer model (Vaswani et al., 2017) based on a self-attention mechanism also benefits from explicit syntactic knowledge in the MT tasks. Duan et al. (2019) demonstrate the usefulness of syntactic knowledge cues in MT tasks by adding dependency information of source language sentences to positional encoding and word embedding to improve translation quality while keeping the Transformer model constant. Z. Zhang et al. (2020) integrate the syntactic dependency information into the self-attention module, creating syntax attention that specifies syntactic structures. The MT engine not only maintains the interpretability of the syntactic tree structure but also supports compatibility with various forms of explicit and structured knowledge. In the low-resource translation task, Chakrabarty et al. (2020) propose two methods called self relevance and word-based relevance to incorporate different levels of linguistic knowledge into the NMT engine, where the combination of three types of linguistic features, lemma, part-of-speech, and dependency labels, gives the best translation performance.

1.2.2 Pre-trained Language Models

Another approach to data augmentation is to apply more accessible monolingual data. In order to construct an MT engine, it is essential to use monolingual data for language model training in SMT. However, it is not typical for NMT to follow such a trend. In most NMT paradigms, it is not common to have a large amount of monolingual data to assist the MT engine, and the language model does not function as a separate module to guide the target language generation. This is because neural networks in NMT can already implicitly capture more linguistic knowledge from bilingual data and play the role of language models on the decoder side. Although neural networks make the structure and principles of the NMT engine more straightforward and intelligent, neural networks are highly data-dependent. Currently, there is not enough bilingual data available for NMT in terms of quantity and distribution. On the other hand, monolingual data is not only more abundant but also encompasses a broader range of topic scenarios.

Pre-trained language models, also referred to as pre-trained models, have become increasingly popular in different NLP tasks with the advancement of pre-training techniques in recent years. During the past few years, Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) has gained significant breakthroughs in NLP tasks and has inspired the proposal of additional pre-trained language models (Y. Liu et al., 2019; Conneau et al., 2020). BERT adopts the Transformer model architecture to perform self-supervised learning on a large-scale unlabeled monolingual corpus with Masked Language Model (MLM) and Next Sentence Prediction (NSP) as pre-training objectives. There have been many studies demonstrating that BERT is equipped with some implicit linguistic knowledge (Jawahar et al., 2019a; Tenney et al., 2019) and can possess world knowledge (Forbes et al., 2019; Poerner et al., 2019). The implicit knowledge learned from the large-scale corpus can be transferred to the downstream NLP tasks through two steps of pre-training and fine-tuning to improve task performance (C. Sun et al., 2019; Yu et al., 2019). Although BERT does not directly work on a quantitative expansion of bilingual data in MT scenarios, it can be regarded as a data augmentation due to its contextual word embeddings and rich linguistic knowledge from a monolingual pre-training, such as syntax and semantics, providing more explicit and implicit feature indications and complementations for the limited bilingual data. As a result, BERT also has promising applications in MT tasks, which can help NMT engines better understand the linguistic information in both source and target languages and improve translation accuracy and fluency. Z. Zhang et al. (2021) propose the BERT-JAM engine for translation tasks, which combines the multi-layer representation of BERT into a fused representation and allows the NMT engine to acquire linguistic features from the encoding layer of BERT. X. Wu et al. (2022) discuss the possibility of BERT as an additional encoder for aggregating contextual features in context-aware NMT and propose a context-aware NMT engine with BERT to aggregate contextual features. Sharoff et al. (2023) examine the contribution of BERT to the establishment of comparable data and apply it in MT engines.

To effectively overcome the problems of translation quality and data diversity in MT scenarios, many studies have also examined various approaches (Gulcehre et al., 2015; R. Wang et al., 2017; Cheng et al., 2019), the most effective of which is probably that of describing and augmenting the data with the knowledge of a large amount of monolingual data. Since monolingual data is widely available, utilizing large-scale monolingual data can help researchers gain a more comprehensive understanding of language laws, this; in turn, can lead to better guidance for

designing model structures and also highlights the importance of monolingual data and the possibility of using pre-trained language models trained with large monolingual data to improve the efficiency of bilingual data in MT tasks.

1.2.3 Strategies Fusion

Neural networks in NMT model sentence information sequentially, while syntactic knowledge is tree-structured. As a result, more complex neural networks are needed to convert from tree structure to sequence information to model and represent the syntactic knowledge. Most studies have been achieved by building tree Long Short-term Memory (LSTM) and Gated Recurrent Unit (GRU) through Recurrent Neural Network (RNN) variants for encoding syntactic knowledge (Eriguchi et al., 2016; H. Chen et al., 2017). However, RNN models may struggle to process long sentences and their complex syntactic structures via their processing range. Since the input to the RNN model is based on the combination of historical words, the information read earlier may be informationally forgotten by the time of the last word. RNN and its variants also suffer from vanishing and exploding gradients, making it more difficult for RNN-based NMT engines to be effectively trained. All these make the modeling and representation of explicit syntactic knowledge inadequate. There have been subsequent attempts to use CNN and GCN to replace RNN for modeling explicit syntactic information (Bastings et al., 2017; K. Chen et al., 2017). However, the complex model structure reduces the efficiency with which sequence information is processed. Multiple complicated neural networks also make training the MT engine more challenging and affect the structural clarity of the MT engine.

Most studies in MT scenarios focus on modeling and representing explicit syntactic knowledge sequentially, whereas the graph-based topological manner of syntactic information that NMT uses to guide language learning and comprehension is currently under-discussed. The syntactic tree is not structured according to a fixed arrangement rule and node order, and each node (words as nodes in the tree) may have a different number of neighbors (other nodes) and dependencies as edges between them. Recent research has aimed to incorporate syntactic tree information into the Transformer model through the use of embeddings or an attention mechanism (Duan et al., 2019; Bugliarello & Okazaki, 2020). However, syntactic embeddings still rely on the initial sequential positional embeddings in the Transformer model. The nodes in the syntax tree can mainly only be used as indicators to guide where the current token should focus if an attention mechanism is used. The propagation of information between the entire tree

structure and the tree node dependencies is not fully utilized. Moreover, whether the attention mechanism can be used as explicit evidence for modeling information is still under discussion (Jain & Wallace, 2019; Serrano & Smith, 2019). A powerful airplane engine built into a car does not mean that the car can reach the speed of an airplane. The modeling and representation of syntactic knowledge in most studies are done with a Transformer model. Although these syntactic knowledge strategies benefit the MT engine, it remains doubtful whether the Transformer model can fully exploit the maximum effectiveness of such strategies. The most effective use of the strategies in the MT tasks might be to model and represent explicit syntactic knowledge using other neural networks that do not rely on the Transformer model in the MT tasks.

Many researchers have tried to redesign the pre-training objectives of BERT in an effort to extract and integrate more linguistic knowledge features since it can only acquire generic syntactic knowledge to solve some basic language tasks. StructBERT (W. Wang et al., 2020) introduces the text sequence recovery task into the MLM pre-training objective and improves another objective NSP by increasing the task difficulty to force the model to learn more about linguistic knowledge. SenseBERT (Levine et al., 2020) presents a new masked word sense prediction task to improve the lexical disambiguation ability of the model by incorporating the actual semantic knowledge of words into the pre-training process. ERNIE (Y. Sun et al., 2019) proposes two masking strategies, phrase-level, and entity-level, that enable the model to learn more about potential entities and long-distance semantic dependencies and produce a more conceptually complete semantic representation. Although they allow BERT to learn richer linguistic knowledge, such as lexical, syntactic, and semantic information, more efficiently, they all require training the model from scratch. It requires a lot of computational resources and time, which may take weeks or months to complete. Meanwhile, the expensive data collection and annotation costs make such an approach face the risk of overfitting and optimization difficulty in training. Therefore, in order to avoid time-consuming and labor-intensive training efforts, most of the studies proceed to fine-tune BERT in MT tasks so that its implicit knowledge can help the MT engines and gain more linguistic knowledge from MT scenarios (Zhu. et al., 2020; J. Yang et al., 2020).

As reported in a survey on data augmentation (Shorten et al., 2021), syntactic knowledge and BERT are widely used in different NLP downstream tasks, where they can be an effective approach for addressing the problem of data sparsity in neural network training and focus on

different strategies to make the given data more valuable for the model, thus improving the performance and robustness of the neural network. So far, most studies only discuss the performance gain of syntactic knowledge on MT engines through either explicit syntactic knowledge incorporation strategies or fine-tuning BERT. It remains unclear whether the fusion of two data augmentation strategies, explicit syntactic knowledge and implicit BERT knowledge, indeed enables an improvement of translation quality. In addition, most of the studies verify the benefit of either strategy on the MT engine from the perspective of BiLingual Evaluation Understudy (BLEU) scores. However, the BLEU score emphasizes sentence surface similarity, and linguistic knowledge, such as syntactic structure and semantic information, is not taken into account, which does not well reflect the robustness of the MT engine. There is no specific explanation of how the applied strategies benefit translation quality and the impact on other neural network modules in the MT engine from the perspective of syntactic knowledge. It is still unclear what syntactic knowledge the proposed strategies focus more on modeling, which syntactic knowledge benefits translation quality, and how explicit syntactic knowledge impacts BERT if they work together in MT tasks. Understanding the interpretability of such strategies, including but not limited to syntactic knowledge, can shed light on how neural networks consider and apply linguistic knowledge, leading to more significant progress in developing fusion strategies regarding linguistic knowledge and pre-trained language models for MT scenarios.

1.3 Thesis Outline

This thesis is devoted to exploring the study of explicit syntactic knowledge incorporation and the pre-trained language model BERT as a fusion strategy for data augmentation in MT scenarios. The specific chapters of the thesis are organized as follows.

- Chapter 1 provides a background on using bilingual data in MT scenarios, as well as the motivation for conducting research in syntactic knowledge and the pre-trained language model BERT for fusion strategies to improve translation quality.
- Chapter 2 discusses the relevant knowledge involved in this study, which is machine translation, pre-trained language model BERT, syntactic knowledge, and graph neural networks, as well as their related work in MT scenarios.
- Chapter 3 explores what the pre-trained language model BERT specifies syntactic knowl-

edge before and after fine-tuning on MT scenarios and detects the relationships between translation quality and syntactic knowledge.

- Chapter 4 investigates the performance of graph attention network to learn syntactic knowledge and discusses the possible application in MT tasks compared with BERT fine-tuned for MT scenarios.
- Chapter 5 presents the novel MT engines based on syntactic knowledge and BERT and investigates the interpretability of translation quality improvement from the perspective of including but not limited to syntactic knowledge.
- Chapter 6 summarizes the contributions and limitations of this study, as well as suggestions for future work.

Chapter 2

Background and Related Work

2.1 Machine Translation

2.1.1 Machine Translation Development

MT refers to the process of automatically translating a natural language (source language) into another language (target language) while maintaining identical semantics, all done by a computer. Over the years, it has garnered considerable interest from industry and academia, achieving significant progress and accomplishments. MT frameworks are constantly being updated to enhance the translation performance of MT engines. Typically, they can be classified as RBMT, EBMT, SMT, and NMT.

Rule-Based Machine Translation: Most early MT engines are based on human rules (Kaji, 1988; Nirenburg, 1989). These engines relied entirely on manually compiled bilingual dictionaries and various translation rules summarised by experts, which are then used by the computer to translate sentences from the source language into those of the target language. Such translation engines do not require model training, but they do require significant expert linguistic knowledge, as collecting bilingual dictionaries and defining translation rules is expensive and complex. If a high-coverage and high-performance translation engine needs to be built, many precise translation rules inevitably need to be collected. In addition, languages are constantly updating their vocabulary and usage, thus increasing the cost of maintaining and iterating on such RBMT engines. As a result, it was gradually replaced by new MT frameworks.

Example-Based Machine Translation: In order to achieve automatic learning of translation

rules from a large corpus of text, EBMT engine was proposed (Nagao, 1984). The basic idea of this method is to find examples in the bilingual corpus that are similar to the sentences to be translated. After that, the translation of the example is modified, such as replacing, adding, deleting, and a series of other operations to obtain the final translation. EBMT uses existing translation resources to automatically extract and summarise rules to construct a bilingual knowledge base and to design rules to deal with ambiguities in the bilingual example base. However, this approach is demanding in terms of the accuracy of the translated examples, and an error in one example may result in a sentence type not being translated correctly. Moreover, the construction of example libraries usually requires word-level aligned annotations. Maintaining the library is made more difficult by ensuring word alignment accuracy.

Statistical Machine Translation: MT engines that utilize large-scale text data and statistical learning have been proposed as the corpus expands and becomes enriched (Brown et al., 1993). Translation between the source and target languages is considered a probabilistic problem, and the translation process is carried out using statistical models. In this approach, any sentence on the target language side can be seen as a translation candidate for a sentence in the source language, and only the corresponding translation probabilities are different. Therefore, the main idea behind this approach is to train a translation engine using statistical methods on a large corpus and then use the translation engine to find a sentence in the target language with the highest score for the source language sentence to be translated. SMT engines are more robust and scalable than Rule- and Example-based MT and can cope with linguistic ambiguity naturally. In addition, it allows high-performance translation engines to be built quickly from an existing corpus and can further improve translation performance when the corpus is increased. Moreover, SMT still requires human input to define translation features. In order to enhance the quality of translations, much effort is needed in feature engineering, which involves manual feature design that can significantly affect the translation performance. Moreover, the design of SMT involves several model modules, making it more complex to build and develop.

Neural Machine Translation: As computer hardware has advanced and the era of big data has arrived, deep learning (Hinton & Salakhutdinov, 2006; LeCun et al., 2015) has achieved significant progress in various tasks such as image classification (He et al., 2016; Krizhevsky et al., 2017), speech recognition (Xiong et al., 2016), which have received a great deal of attention from researchers concerned with the field of NLP. They have found that deep neural networks can

automatically learn new features and representations from objects in natural language (Socher et al., 2013; Mikolov et al., 2013b). It provides a new research idea to solve the problem of NLP of sparse feature overfitting and insufficient ability to describe the language structure. As a result, researchers in the field of MT attempted to improve SMT engines using deep learning techniques. These include improved word alignment modeling (N. Yang et al., 2013; Tamura et al., 2014), sequencing modeling (P. Li et al., 2014; J. Zhang et al., 2015), and bilingual phrase representation (Zou et al., 2013; J. Zhang et al., 2014). Subsequently, NMT engines based on the encoder-decoder framework were proposed (Kalchbrenner & Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014), rapidly replacing the traditional MT engines. The NMT framework not only simplifies the model structure and training process of SMT but also has an attention mechanism (Bahdanau et al., 2014). Such a working mechanism dynamically captures alignment information from both the source and target language sides, allowing NMT to use contextual information on the source language side and obtain more accurate target languages resulting in smoother translation. NMT has an advantage over SMT in that it eliminates the need for feature engineering, as all features can be automatically extracted from neural networks. In contrast to the discrete representation (each word is regarded as a discrete symbol and corresponds in isolation to an index in the word list) used in SMT, NMT uses distributed representation (each word is represented as a vector in a multi-dimensional space) through word embeddings to model sentences with richer information. However, NMT is not easily interpretable, and its training process differs from human perception. Moreover, it heavily relies on data, and the quality and size of the data greatly influence its performance, making training NMT engines with sparse data particularly challenging.

2.1.2 Transformer Model

The Transformer, which has been proposed recently, is now a crucial model for NMT. It utilizes a self-attention mechanism to analyze input sequences, consisting of attention modules and other neural networks in both the encoder and decoder. Such approaches enhance the speed of parallel training of the MT engine, as well as its ability to recognize long-distance language dependencies. The Transformer model structure is depicted in Figure 2.1 and consists mainly of several encoders and decoders, which have multi-head attention modules, feed-forward neural networks inside, and positional encoding for word embeddings.

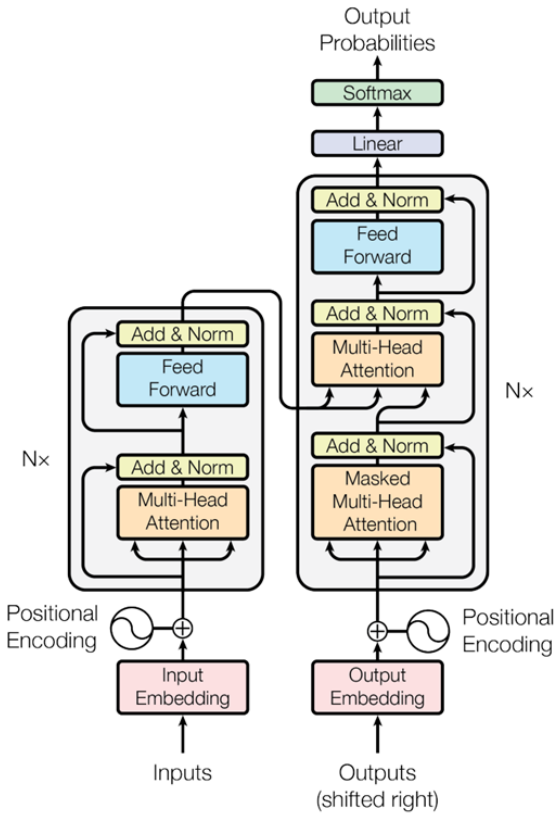


Figure 2.1: Structure of the Transformer model (Vaswani et al., 2017).

Encoder: The encoder is stacked with N identical layers. Each layer consists of two sub-layers, the first of which is a multi-head attention network, and the second is a fully connected feed-forward neural network. The number of neural networks and non-linear transformations in the Transformer model complicates the information transfer. It therefore introduces a residual connection layer and layer normalization to make information transfer more efficient and to solve the gradient vanishing/exploding problem that tends to occur during the training of deep neural networks. The output of each sub-layer is $LayerNorm(x + Sublayer(x))$, where $Sublayer(x)$ is a function implemented by the particular layer itself. To ensure the implementation of these residual connections, the model generates the same output dimension d_m for all sub-layers and embedding layers.

Decoder: The decoder is also stacked with N identical layers. Not only does it have multi-head attention and a fully connected feed-forward neural network like the encoder, but the decoder inserts a third sub-layer called encoder-decoder attention. Such a sublayer helps the model to use the representation information of the source language sentences to generate target language representations in different sentence sequences. The encoder-decoder attention sublayer shares

the same structure as the self-attention sublayer. The only difference is that the encoder-decoder attention sublayer obtains its features from a different source, which not only considers target language information but also takes the output of the encoder to help the decoder to obtain vector representations at the source language side. As with the encoder, the decoder uses the residual connection layer and layer normalization between each sub-layer.

Multi-head Attention Mechanism: The basic idea of the attention mechanism is to construct the mapping functions from source or target language to retrieve information related to a given Query (Q) from a set of Key (K) - Value (V) pairs, where the Q, K, and V are representation vectors for one given sentence. The output of the attention network is a weighted sum of values, where the weight assigned to each V is calculated from the matching function of the Q with the corresponding K. A particular attention network structure called the Scaled Dot-Product attention is designed in the Transformer model, shown in Figure 2.2.

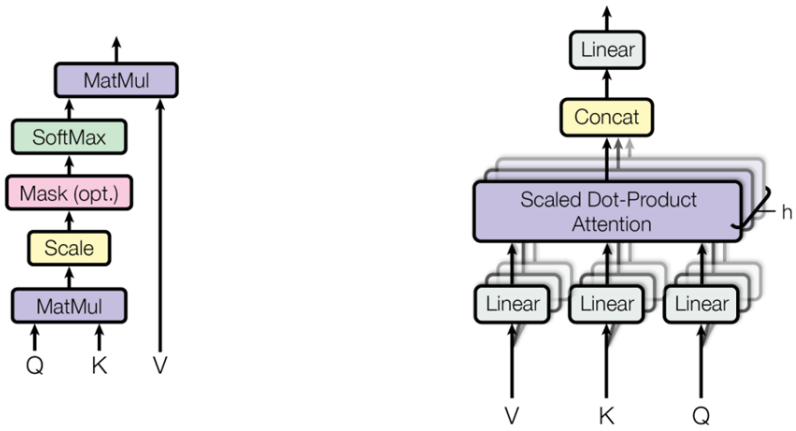


Figure 2.2: The process of calculating Scaled Dot-Product attention (Vaswani et al., 2017). The left figure illustrates the process of self-attention calculation, and the right one shows the implementation of multi-head attention with other functions.

Assume that the input Q and K are of dimension d_k and the value V is of dimension d_v. The process of calculating the attention is to calculate the dot-product of Q and K and divide by $\sqrt{d_k}$, then apply the Softmax function to calculate the weights, and obtain the final output by weighted sum, as shown in the Equation 2.1 below.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

The multi-head attention mechanism refers to the idea of an ensemble and further extends the

existing attention structure horizontally. The given Q, K, and V are first represented in different spaces using different linear mappings. In order to produce the final representation, context vectors are concatenated after being computed in various subspaces using various attention networks. The specific calculation formula is as follows in Equation 2.2 and 2.3.

$$MultiHead(Q, K, V) = Concat(head_1, head_2..., head_n)W \tag{2.2}$$

$$head_i = Attention(QW_i^q, KW_i^k, VW_i^v) \tag{2.3}$$

where W, W_i^q, W_i^k, and W_i^v are matrix parameters, n means the number of heads of the self-attention mechanism. There are three places in the Transformer model where the multi-head attention mechanism is used: (i) At the encoder attention layer, the source language sentence is treated as Q, K, and V. The input sequence performs the multi-head attention calculation on itself to obtain the internal contextual features. (ii) The attention network with a mask in the decoder is also a multi-head attention network with a similar motivation to (i) for accessing features within its sentences. (iii) Between the encoder and the decoder, multi-head attention is used to obtain a representation of the information related to the source and target language sentence. The attention input Q comes from the output of the previous attention layer of the decoder, while K and V come from the output of the encoder.

Two masking strategies are applied in the multi-head attention mechanism. The first one is the padding mask. During training, multiple sentences are processed and batched. However, the length of the sentence sequences within each batch can be different. In order to make it easier to represent the sequences in a matrix, an alignment operation is performed. The shorter sequences are padded with zeros to fill the remaining positions. Such zeros do not have any practical significance and do not affect the computation of the attention mechanism. The second one is the future mask. The decoder makes predictions from left to right, meaning that its output at a given time can only be based on the output before that time. Future information must be masked to avoid the model observing future information at each location on the target language side during training. In order to achieve this, a mask matrix is created where all the upper triangular values are set to infinity. When decoding, suppose the current position is t. The future mask blocks any information from positions after t from impacting the current attention calculation.

Feed-forward Neural Network: After calculating the self-attention, the input is transformed with the following feed-forward neural network, including two linear transformations and a Rectified Linear Unit (ReLU) activation function, as shown in Equation 2.4.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{2.4}$$

where W_1, W_2, b_1, and b_2 are the matrix and vector parameters, respectively. The primary function of the feed-forward neural network is to map the representation obtained from the multi-head attention mechanism into a new space to make the following operations, like non-linear transformations, easier to perform. The feed-forward neural network part of the vanilla Transformer model has a hidden layer dimension of 2048. In order to improve performance, one option is to increase the size of the hidden layer in a feed-forward neural network to 4096 or 8192 if GPU support and training costs are available. Therefore, when implementing the Transformer model, balancing translation accuracy and storage/speed is essential.

Positional Encoding: As the Transformer model does not contain any recurrent and convolutional network structures, in order for the model to make use of the sequence order information, the model needs to encode the position of words to represent the position relationships of different words in the sequence. Therefore, positional encoding is applied to the input representation at the beginning of the encoder and decoder structures. The Equations are shown in 2.5 and 2.6 below.

$$PE(pos, 2i) = sin(pos/10000^{2i/d_m}) \tag{2.5}$$

$$PE(pos, 2i+1) = cos(pos/10000^{2i/d_m}) \tag{2.6}$$

$$PE(pos+k, 2i) = PE(pos, 2i) * PE(k, 2i+1) + PE(pos, 2i+1) * PE(k, 2i) \tag{2.7}$$

$$PE(pos+k, 2i+1) = PE(pos, 2i+1) * PE(k, 2i+1) - PE(pos, 2i) * PE(k, 2i) \tag{2.8}$$

where *pos* is the word position in the sentence and i is the vector dimensions, d_m is the model dimension. Positional encoding, as shown in Equations 2.7 and 2.8, represents the distance between words by expressing $PE(pos + k)$ as a linear function of $PE(pos)$, where k is an arbitrary fixed offset, providing the possibility of representing relative positional information.

Making improvements to it can also result in additional performance improvements (Shaw et al., 2018; Dufter et al., 2022).

Recently, the Transformer model has gained popularity and has spawned many improved versions, such as relative position coding (Shaw et al., 2018), local attention mechanisms (B. Yang et al., 2019), multi-layer information interaction (Q. Wang et al., 2018), and deep networks (X. Liu et al., 2020a). Studies are also focusing on improving the Transformer model, particularly its self-attention network component. They are exploring ways to make the operations more efficient to achieve better results. F. Wu et al. (2019) used a dynamic CNN to replace the self-attention network of encoders and decoders. The translation performance is comparable to or even slightly better than the Transformer model while maintaining inference efficiency. An innovative model called the Reformer (Kitaev et al., 2020) has been proposed to help process longer input texts more efficiently in the transformer model, which replaces the self-attention mechanism with a local sensitive hash. Some studies have also found that attention mechanisms can capture linguistic phenomena in the Transformer model (Voita et al., 2019; B. Li et al., 2020), where each attention head of the model appears to specialize in capturing particular information, such as identifying low-frequency words or distinguishing between syntactic information and lexical disambiguation. Since the Transformer model relies on costly GPU devices, trimming and accelerating the model is also another interest for many researchers working on MT engines. Reducing the computational intensity is a desirable strategy, such as using low-precision floating-point numbers (Ott et al., 2018) and integers (Bhandare et al., 2019) for computation, or pruning the model parameter matrix to reduce the size of the overall model (J. Li et al., 2021). Even most of the subsequent pre-training models have been proposed based on the framework and theory of the Transformer model, e.g., Generative Pre-trained Transformer-1 (GPT-1) (Radford & Narasimhan, 2018), Generative Pre-trained Transformer-2 (GPT-2) (Radford et al., 2019), Pixel-based Encoder of Language (PixEL) (Rust et al., 2022).

2.1.3 Machine Translation with Neural Networks

Early NMT engines (Kalchbrenner & Blunsom, 2013; Cho et al., 2014) utilized a simple and intuitive approach to translation called the encoder-decoder framework. Such MT engines typically use an RNN as an encoder to convert source language sentences into a vector representation. They then utilize another RNN as a decoder to generate the target language sentences from this representation. Subsequently, Bahdanau et al. (2014) introduced an attention mechanism in

the encoder-decoder framework to improve the NMT performance significantly. The attention mechanism allows the MT engine to better deal with long-distance dependencies in sentences and to solve the problem of RNN that tend to lose and forget information when processing long texts. Because of its simple structure and remarkable performance, NMT with attention mechanism has received widespread attention and investigation by researchers. Luong et al. (2015) proposed using a local attention model, an improved version of global attention that can significantly reduce the computational effort required. Y. Wu et al. (2016) published a Google NMT engine with an attention mechanism, which effectively solves the problem of gradient vanishing in deep models by introducing residual connections between layers. In addition, researchers have analyzed various components of current MT engines for their effectiveness in completing translation tasks: dealing with out-of-vocabulary words (Jean et al., 2014; Sennrich et al., 2016a; X. Li et al., 2016), designing more efficient model training algorithms (Shen et al., 2015; Bahdanau et al., 2016), fusing linguistic information (Eriguchi et al., 2016; S. Wu et al., 2017a), and explaining the internal mechanisms of NMT (Y. Ding et al., 2017). Researchers have also looked into developing MT engines based on CNN after the success of RNN-based engines. Meng et al. (2015) used a CNN to encode the source language and integrate it into a MT engine. Gehring et al. (2016) suggested using a CNN-based encoder that allows for a quicker way to capture long-range connections in source language sentences.

However, both RNN and their related variants are limited by several factors, such as non-parallel processing of inputs leading to less efficient model training, and long-term memory can cause some information to be lost. Even though the CNN can grasp both local and global details of a sentence, using the convolutional layer for feature extraction can consume many computational resources in the translation model. In addition, although the attention mechanism can establish a connection between source and target sentences, there is still room for improvement in describing the correlation between words within sentences and modeling linguistic knowledge features.

Vaswani et al. (2017) proposed a Transformer model based on the self-attention mechanism, which is characterized by the standard feed-forward neural network and the self-attention as primary working mechanisms. The transformer model does not rely on recurrent and convolutional operations. Instead, it models feature from text units at any distance in a text sequence through the self-attention mechanism and implement parallelism, allowing it to balance model

performance and time spent on model training. The Transformer model has inspired many studies on MT tasks, therefore. P. J. Liu et al. (2018) presented memory compressed Transformer, an early variant of a Transformer model for long sequence processing. The model can process longer input sequences while generating fluent and continuous text sentences by modifying measures such as the input matrix module and the attention matrix size. Hassan et al. (2018) used the Transformer model as a basis to exploit the full potential of the deep neural network model with the help of pairwise learning, joint training, and consistency regularisation to achieve translation performance close to or at the level of amateur translators on certain news test sets. X. Liu et al. (2020b) improved the training efficiency of Transformer models in MT scenarios by introducing norm-based curriculum learning. There are currently several ways to improve the Transformer model for MT tasks. These include but are not limited to improving the attention mechanism (B. Yang et al., 2018; F. Wu et al., 2019), optimizing neural network connections (Dou et al., 2018; X. Wang et al., 2019), syntactic incorporation strategies (Bugliarello & Okazaki, 2020; Duan et al., 2020), and optimizing the MT engines through structural search (So et al., 2019; Y. Li et al., 2020).

The Transformer model yields more significant improvements in sentence fluency compared to the RNN-based MT engines. However, the Transformer model still produces translations that do not conform to grammatical constraints (Bugliarello & Okazaki, 2020; Slobodkin et al., 2022), although its attention mechanism can capture some linguistic knowledge (Voita et al., 2019). Neural networks use implicit modeling of linguistic knowledge likened to a black-box operation, which makes it challenging to interpret and direct the translation process from a linguistic standpoint. MT engines also have a tendency to overfit sparse linguistic knowledge and may even fail to effectively learn the underlying linguistic knowledge when not provided with explicit linguistic guidance and limited training set scenarios (Kumar et al., 2021; Soky et al., 2022). Typically, NMT operates without human intervention, and the translation outcome cannot be directly linked to human understanding. To address this, a useful approach is to integrate existing linguistic knowledge into the NMT engine, enabling it to function more like a human thought process. Therefore, how to model and represent linguistic knowledge more efficiently and incorporate it into NMT with a limited training set to improve the translation quality is still a challenging MT task.

2.2 Pre-training and Fine-tuning

2.2.1 Pre-trained Language Model

A language model measures the fluency of target language translations by learning the word sequence generation laws of the target language from a monolingual corpus. It is trained to comprehend the structure, grammar, and context of language, where such an understanding enables them to predict the likelihood of a sequence of words or to generate coherent and contextually relevant text. A pre-trained language model refers to a language model that is updated by learning linguistic and general knowledge from a large corpus through self-supervised learning, where such knowledge is not related to any specific NLP task. As a type of transfer learning, fine-tuning allows the learned model parameters to be shared in some way with other neural networks in a downstream task, thereby speeding up training and optimizing the knowledge needed for the model without learning it from scratch. A two-stage pre-training and fine-tuning approach not only save the time and computational power needed to train a model but the rich knowledge from pre-training can be implicitly applied to the downstream tasks.

Two types of knowledge are focused on in pre-training and fine-tuning: general knowledge and specific knowledge. The general knowledge is usually acquired during the pre-training phase. The pre-trained language model learns common word combinations, generic objects, and basic linguistic knowledge through co-occurrence patterns in the text by pre-training on the large corpus. However, such knowledge is generic and not specific to any downstream task or need, which is relatively general. In contrast, specific knowledge is usually acquired during fine-tuning and is task-specific oriented. The deeper the neural network, the more difficult it is to train the underlying parameters effectively. The higher-level abstractions may not be represented if the underlying features are insufficient. The gradients in back-propagation can vanish at the bottom layer, which makes updating the model parameters difficult. Fine-tuning involves learning generalized features from pre-training to obtain the underlying parameters necessary to produce accurate knowledge and cues for the downstream tasks. It is crucial to follow this process since some parameters are challenging or impossible for neural networks to train directly.

A distributed representation or word embedding is when each word is mapped as a vector point in a multidimensional space. Early pre-training focuses on how to learn word embeddings, with representative works such as Word2vec (Mikolov et al., 2013a) and Global Vectors for Word

Representation (GloVe) (Pennington et al., 2014). They utilize shallow neural network models to train such static word embeddings to compensate for limited computing power and a lack of algorithms. While static word embeddings can capture specific linguistic knowledge, they do not account for the dynamic contextual information surrounding the words, where the ambiguity problem in sentences is not well resolved.

The current focus of pre-training is to create contextual word embeddings or gain other knowledge from a large corpus via the language models, with representative work such as Embeddings from Language Models (ELMo) (Peters et al., 2018), BERT (Devlin et al., 2019), and GPT-1 (Radford & Narasimhan, 2018). The word embeddings of these models are no longer static but dynamic to provide more implicit linguistic information to the neural networks by taking into account the word position in the sentence and contextual information. With the development of computing hardware, these efforts have also allowed using more sophisticated model structures. Researchers have proposed numerous enhanced versions of BERT after it showed impressive performance in various NLP tests, e.g., RoBERTa (Y. Liu et al., 2019) used a broader corpus as the training set and implement dynamic masks in MLM while removing the NSP. ALBERT (Lan et al., 2019) could share parameters across layers, significantly reducing the number of parameters, and ELECTRA (K. Clark et al., 2020) replaced the MLM in BERT with Replaced Token Detection (RTD), solving the inconsistency problem in the pre-training and fine-tuning phases of [MASK] token.

Several multilingual models have also been proposed inspired by BERT in addition to the large monolingual pre-trained language models. Pires et al. (2019) developed Multilingual Bidirectional Encoder Representations from Transformers (mBERT), which provides a contextual representation for 104 languages via pre-training on a monolingual corpus of different languages and is improved with a regularisation strategy. Conneau and Lample (2019) released the cross-lingual pre-trained language model called Cross-lingual Language Model (XLM), which used two approaches for cross-lingual modeling: unsupervised learning relying on monolingual datasets and supervised learning using the parallel corpus. XLM-RoBERTa (XLM-R) (Conneau et al., 2019) inherits the training method of XLM but borrowed ideas from RoBERTa to train 100 languages using over 2 TB of text data, demonstrating for the first time the possibility of implementing multilingual modeling without sacrificing performance.

2.2.2 BERTology: Working Mechanism and Interpretability

BERT is a deep learning-based pre-trained language model proposed by Google AI, which is crucial in recent research in NLP, achieving the best results in different NLP tasks since its creation. The basic model structure of BERT is similar to the encoder in the vanilla Transformer model, as shown in Figure 2.3, where the BERT-base and BERT-large models use 12 and 24 layers of deep neural networks, respectively. The Transformer architecture is more robust than RNN-based models when it comes to text encoding and is also more efficient for large-scale training on high-performance devices like GPU.

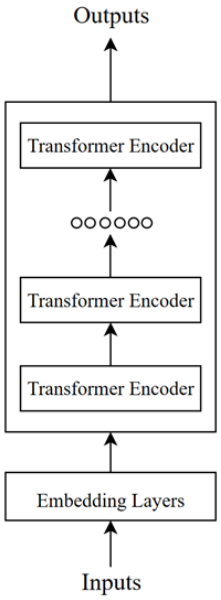


Figure 2.3: Model structure of the pre-trained language model BERT.

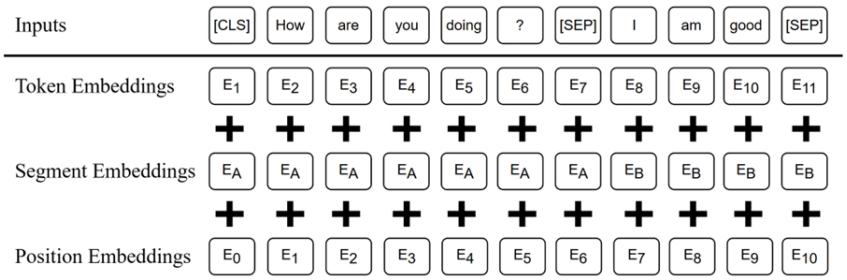


Figure 2.4: The construction of contextual word embeddings in BERT.

Input Representation: The Input Representation of BERT consists of adding token embeddings, segment embeddings, and position embeddings, as shown in Figure 2.4. Also, BERT inserts a [CLS] before the first sentence as the start token. If the inputs are two sentences, two [SPE] are added as the separator of the two sentences and the end token of the last sentence,

respectively.

Similar to the standard neural networks process tokens in a given sentence, BERT also converts word vectors into word embeddings with the help of a word vector matrix. Assuming that the one-hot vector corresponding to the input sequence is denoted as $e^t \in \mathbb{R}^{N\times|\mathbb{V}|}$, the corresponding word embeddings v^t are shown in the following Equation 2.9.

$$v^t = e^t W^t \tag{2.9}$$

where $W^t \in \mathbb{R}^{|\mathbb{V}|\times e}$ denotes the trainable matrix for word embeddings, $|\mathbb{V}|$ is the vocabulary size, and e is the word embeddings dimension.

The segment embeddings encode the segment to which the current word belongs. When the input sequence is a single sentence, all words are encoded as 0. When two sentences as input sequences, each word in the first sentence is encoded as 0, whereas words in the second sentence are encoded as 1. Assuming that the segment embeddings matrix W^s converts the segment encoding $e^s \in \mathbb{R}^{N\times|\mathbb{S}|}$ into vectors, the segment embeddings v^s are obtained from Equation 2.10.

$$v^s = e^s W^s \tag{2.10}$$

where $W^s \in \mathbb{R}^{|\mathbb{S}|\times e}$ is the matrix for segment embeddings, $|\mathbb{S}|$ is the number of segments, e is the segment embeddings dimension.

The position embeddings are used to encode the absolute position of each word, and each word in the input sequence is sequentially converted to position one-hot coding. The position embeddings matrix W^p converts the position one-hot encoding $e^p \in \mathbb{R}^{N\times N}$ into the vectors to obtain the position embeddings v^p, as shown in Equation 2.11.

$$v^p = e^p W^p \tag{2.11}$$

where $W^p \in \mathbb{R}^{N\times e}$ is the matrix for position embeddings, N is the maximum position length and e is the position embeddings dimension.

Instead of traditional auto-regressive language modeling, BERT uses auto-encoding approaches to implement pre-training. The primary pre-training objectives of BERT include the MLM and NSP, more ore details are described below.

Masked Language Model: In order to achieve bidirectional modeling of text, where the predictions at the current moment depend on both 'history' and 'future', BERT has adopted an approach similar to Cloze test, which it calls the MLM. During the MLM pre-training task, certain words in the input sentence are masked, which requires the model to use the contextual information surrounding the masked word to predict the masked location. BERT masks 15% of the WordPiece in the input sequence, using [MASK] tokens and replacing the original word to indicate that the position has been masked. However, this causes inconsistency between the pre-training and the downstream task fine-tuning, since the [MASK] artificially introduced tokens do not appear in the actual downstream tasks. To alleviate this problem, when words in input sentences are masked, they are not always replaced with [MASK] tokens, but one of the following three operations is chosen on a probabilistic basis.

- 80% probability of replacement with [MASK] token.
- 10% probability of replacement with any random word in the word list.
- 10% probability of keeping the word unchanged.

Next Sentence Prediction: During the MLM pre-training process, the model uses contextual information to fill in masked words, which helps to enhance its semantic understanding and representation. However, for tasks such as text comprehension that require two input sentences, MLM cannot explicitly learn the association between the two inputs, e.g., modeling the chapter and question to find the answer. Therefore, BERT also introduces a second pre-training objective called NSP to construct the relationship between the two input sentences. NSP is a binary classification task in which the model must determine whether sentence B follows sentence A. The training process is shown below.

- Given sentence pairs A and B, where there is a 50% probability that B is the next sentence of A.
- Given sentence pairs A and B, there is also a 50% probability that B is randomly selected from the corpus and not the next sentence of A.

Large-scale pre-trained language models have proven to be effective in achieving desirable results in various NLP tasks, which suggests that BERT may already have acquired a significant amount of linguistic knowledge, such as syntactic and semantic knowledge. A considerable amount of effort has been put into identifying the areas where BERT has limitations in terms of comprehension and detecting the acquisition of knowledge.

Many studies have shown the existence of syntactic knowledge in BERT. Goldberg (2019) showed that BERT considers a subject-predicate agreement in completing the cloze task, even with interfering clauses and nonsense sentences between subjects and verbs. Jawahar. et al. (2019b) found that BERT starts with surface features at the bottom, follows by syntactic features in the middle, and then semantic features at the top. In addition, Hewitt and Manning (2019) and Niven and Kao (2019) also discovered similar phenomena of syntactic knowledge distribution within the BERT model and derive the corresponding syntactic trees from it.

BERT knows not only syntactic knowledge but also semantic knowledge and common sense. Coenen et al. (2019) investigated how BERT encodes generic linguistic features within the model and find that at a high-level, linguistic features appear to be represented in separate semantic and syntactic subspaces, and they also provide evidence for a fine-grained geometry representation of lexical meaning. Tenney et al. (2019) demonstrated that utilizing a probing classifier reveals that BERT contains encoded information regarding entity types, relations, and semantic roles. E. Wallace et al. (2019) claimed that BERT has difficulty representing numerical values and lacks the ability to effectively represent floating point numbers. Forbes et al. (2019) found that BERT has the ability to determine the potential uses of various objects and their characteristics. However, it lacks knowledge of how these objects interact with one another.

Many attempts have been made to apply BERT to obtain remarkable outcomes in downstream tasks. Nevertheless, there has not been enough investigation into how the fine-tuning of these downstream tasks impacts the internal knowledge of BERT. Furthermore, due to the complexity of the probe classifier, the neural network might have unobserved knowledge, and it is even unclear whether the observed comprehension comes from the probe classifier or the primary model. Researchers still face challenges when it comes to analyzing and exploring deep knowledge inside pre-trained language models, such as BERT.

2.2.3 BERT in Neural Machine Translation

The translation engine can improve the quality of its output by using BERT for the encoder or decoder, a pre-trained language model that helps create better sentence representations. It means that the engine can better understand the context and meaning of the text being translated. Additionally, BERT can provide the MT engine with richer implicit linguistic knowledge through fine-tuning, making the training process easier and more efficient.

Z. Zhang et al. (2021) proposed a BERT-based MT engine called BERT-JAM, which dynamically combines the BERT representation with the encoder and decoder representations. They also use a three-stage optimization strategy to fine-tune BERT-JAM, which allows the model to alleviate catastrophic forgetting problems during fine-tuning. Shavarani and Sarkar (2021) enhanced NMT with aspect-level semantic information in the form of dense vectors obtained by BERT to ensure the stability and reliability of NMT training. Their experiments show that this approach improves the quality of translation without increasing the computational complexity. Weng et al. (2020) proposed APT, an NMT framework based on BERT and GPT, which uses adapters to transform general knowledge into task-specific representations, while BERT and GPT dynamically fuse such representations into MT engine. Some studies also focus on the model parameters used by BERT to initialize the encoder in MT tasks (Clinchant et al., 2019; Imamura & Sumita, 2019b). However, BERT does not always improve MT engine performance but brings about performance degradation on some rich-resource languages (Zhu. et al., 2020). It may be due to the fact that the sentence pairs of some rich resource languages are relatively sufficient, and simply initializing the model parameters by BERT is not urgently needed for the MT engine.

As a milestone on pre-trained language models, BERT validates the possibilities of applying monolingual data in MT tasks via pre-training and inspires the development of NMT based on multilingual pre-trained language models (G. Chen et al., 2021; Üstün et al., 2021). The empirical analysis and applications of BERT also help humans to understand pre-trained language models and support future improvements. BERT has made significant contributions to MT tasks, where its contextual word embeddings and generic linguistic knowledge learned from pre-training can enhance the generalization ability of MT engines, especially in cases with limited bilingual data. Most studies show that incorporating BERT improves the performance of the MT engine, as demonstrated by metrics like the BLEU score. (Imamura & Sumita,

2019b; Zhu. et al., 2020; Z. Zhang et al., 2021). There has not been enough discussion about the links and understanding of translation quality and BERT knowledge, specifically concerning linguistic knowledge. BERT has the ability to recognize syntactic knowledge, which can provide clarity on sentence structure. However, there needs to be more exploration into how BERT acquires and applies such syntactic knowledge in MT scenarios, as well as what kinds of syntactic knowledge prove challenging when BERT is inside an MT engine. In addition, it remains to be investigated whether fusing other linguistic-based strategies in MT tasks, such as explicit syntactic knowledge, with BERT knowledge can help improve translation quality.

2.3 Linguistic Knowledge - Syntactic Dependency

2.3.1 Syntactic Dependency

The primary purpose of dependency parsing in NLP is to analyze the syntactic structure of a sentence and clarify the dependencies between words. A proper sentence should have a legitimate internal syntactic structure, where such a structure used to specify the connections between words in a sentence can be called syntactic dependency. There are multiple language dependency corpora and annotation frameworks, including Swedish Treebank (Marneffe & Nivre, 2019), Danish Dependency Treebank (Kromann, 2003), Stanford Typed Dependencies (De Marneffe et al., 2006), and Universal Dependencies (UD) (Nivre et al., 2016). In order to further clarify the definition of syntactic dependencies and reduce ambiguity, the knowledge covered in this section is based on the UD representations.

In a sentence, the basic unit of syntactic dependency is the words, and the dependency relation refers to the relationship of mutual constraints/dependencies between words. Syntactic dependencies are usually labeled directly on the sentence or presented as a syntactic tree, as illustrated in Figure 2.5. Words in a sentence or syntactic tree are connected by an edge known as the dependency arc, where the beginning of the arrow is termed the head and the word being pointed to is dependent along with a dependence relation, as shown in Figure 2.6. Whether syntactic dependencies are labeled directly on the sentence or presented as a syntactic tree, there is one word in it that is the head of the sentence, dependent on a nominal dependency relation called *root*, and all other words are dependent on another word in the sentence along with their dependency relations.

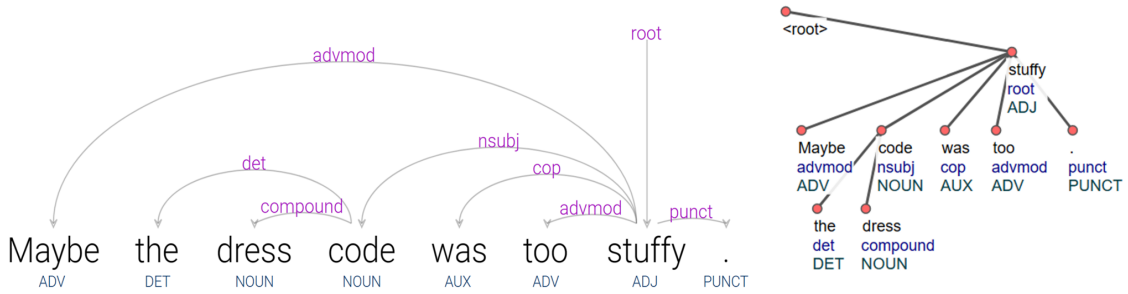


Figure 2.5: The representation of syntactic dependencies in a sentence can consist of direct labeling on the sentence or by a syntactic tree. Words serve as the basic units of a sentence or tree, and dependency relations specify the structure of sentences and the interdependencies between words.

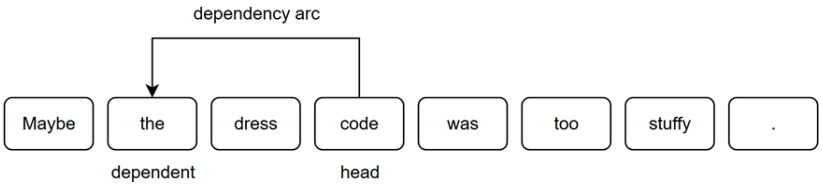


Figure 2.6: The form of syntactic dependency in a given sentence. The line with the arrow connects the words *code* and *the*, and the arrow points from *code* (head) to *the* (dependent) to indicate that they are dependent in the sentence.

The relations in syntactic dependencies between words might be either adjacent or non-adjacent, e.g., the dependency relation called *nsubj* (nominal subject) between *stuffy* and *code* in Figure 2.5 indicates that *code* is the syntactic subject of the sentence and the proto-agent of a clause (if the clause exists). Also, *compound* (word-level compounding) implies that *dress* is dependent on *code* and the these two words are noun compounds in the given sentence.

The use of syntactic dependencies assists in clearing up the ambiguity regarding the relationships between words and the overall structure of a given sentence. Consider a given English sentence: *Clever boys and girls go to school.* Sentence ambiguity plays a crucial role in translation scenarios, such as when translating it into French. There can be two different translations: *Les garçons habiles et les filles vont à l'école* or *Les garçons et les filles habiles vont à l'école.* What is confusing is whether the word *Clever* in the sentence refers to only *boys* or to both *boys* and *girls*. In Figure 2.7, the syntactic dependencies of this sentence reveal that *Clever* and *boys* are connected by the dependencies relation called *amod* (adjectival modifier), while *boys* and *girls* are linked by *conj* (conjunct). According to the dependency relation *amod*, *Clever* is used as an adjective to modify the noun part that follows. However, it does not disambiguate the scope of the modifier *Clever* since it is directly connected to *boys* with the relation called *amod*

(adjectival modifier), which implies that *Clever* is modifying *boys*. The scope of the adjective is not explicitly spread to both *boys* and *girls*. The conjunction *and* is treated as a *cc* (coordinator) that links *boys* to *girls* with the *conj* (conjunct) relation, but without a mechanism to distribute the modifier *Clever* to both conjuncts. To accurately reflect that *Clever* modifies both *boys* and *girls*, an additional dependency relation or a more complex representation would be needed. Some dependency frameworks might use a shared modifier construction or employ a secondary link from *Clever* to *girls* to indicate that the adjective applies to both nouns in the coordination. However, in different annotations, especially those that produce simpler trees, such nuances may not be captured, and the interpretation of the scope of modifier would rely on semantic understanding beyond the syntactic representation.

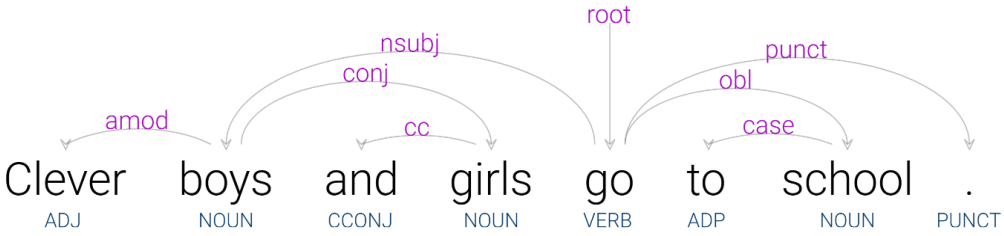


Figure 2.7: Syntactic dependencies of an English sentence.

Original: 深圳市交警以前曾选用非常规处罚。
NMT: Shenzhen traffic police used to choose very rules of punishment.
Reference: Shenzhen's traffic police have opted for unconventional penalties before.

Figure 2.8: Chinese source sentence and translations from different sources. One is from gold translation, and another is from the NMT engine.

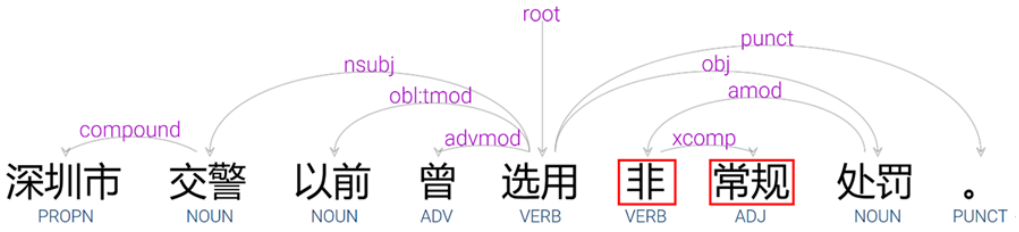


Figure 2.9: The gold syntactic dependency structure of the given Chinese sentence. The red boxes contain two Chinese characters demonstrating the syntactic components and their instructions based on their dependency relations.

Another example is a translation scenario where the source language is Chinese, and the target language is English, as shown in Figure 2.8, from the NMT engine and the reference transla-

tion, respectively. Figure 2.9 shows the dependencies between characters determine the syntactic structure of the source language sentence, the Chinese character in the first red box (starting from the left) signifies the opposite state of something, similar to the English words *non*, *un* as a prefix, or *not* as an adverb. In the second red box (starting from the left), the Chinese characters represent a frequently applied rule or regulation. Unfortunately, NMT fails to identify this scenario and misinterprets the structure of the Chinese sentence, resulting in an inaccurate English translation, which is illustrated in Figure 2.10. In the Chinese language, sentence structure is not indicated by spaces. Therefore, people must rely on contextual clues or common sense to understand the meaning of a sentence. However, the NMT does not possess an impeccable linguistic knowledge system to analyze sentence structures accurately. It is not well aware of the dependencies between the Chinese characters, which are *amod* (adjectival modifier of a nominal) and *xcomp* (open clausal complement) for the red boxes indicating that they function differently in sentence constituents in Figure 2.9. Despite the fact that NMT is capable of automatically capturing the source language sentence features via neural networks, due to factors such as corpus size and model training, it cannot be guaranteed to always learn linguistic features that limit accurate translations.

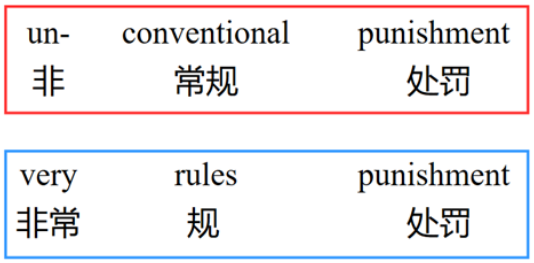


Figure 2.10: The blue box demonstrates that NMT engine misinterprets the structure of Chinese sentences, while the correct structure is shown in the red box.

2.3.2 Universal Framework and Treebank Annotation

There are two main methods for obtaining syntactic dependency information for sentences: using a manually annotated golden corpus or using an automatic dependency parser to obtain machine-annotated information. Although automatic dependency parsers can quickly analyze the syntactic structure of sentences, they do not guarantee the correctness of syntactic annotations. In order to ensure precise experimental findings, this study utilizes a corpus with gold syntactic annotations to identify the interpretability of syntactic knowledge on neural networks, where the annotations and principles of the gold-annotated corpus used in the study are shown

	Nominals	**Clauses**	**Modifier words**	**Function words**
Core arguments	nsubj obj iobj	csubj ccomp xcomp		
Non-core dependents	obl vocative expl dislocated	advcl	advmod discourse	aux cop mark
Nominal dependents	nmod appos nummod	acl	amod	det clf case
Coordination	**MWE**	**Loose**	**Special**	
conj cc	fixed flat compound	list parataxis	orphan goeswith reparandum	punct root dep

Table 2.1: A specific taxonomy of syntactic relations in UD. The rows in the upper part of the table indicate the head-related syntactic function categories, and the columns indicate the dependent structural categories. The lower part of the table shows the relations that are not dependencies in the narrow sense.

below.

UD is a cross-language treebank annotation platform developed for multilingualism to facilitate cross-lingual learning, the development of multilingual interpreters, and providing linguistic knowledge to aid parsing research. The philosophy of UD is to offer guidelines that are universal and consistent, which will make it easier to annotate linguistic knowledge in a consistent manner across different languages. UD contains over 200 treebanks in more than 100 languages, and its annotations are not limited to POS tags (part-of-speech categories), Features (additional lexical and grammatical properties of words), and Syntactic Relations (standard syntactic relations between clause constituents). UD follows its taxonomy rules for the detailed division of syntactic relations, which contain head-related syntactic function categories (Core arguments of clausal predicates, Non-core dependents of clausal predicates, and Dependents of nominals), and the corresponding dependent structural categories (Nominals, Clauses, Modifiers and Function words). It is also supplemented by relations that are not dependencies in the narrow sense (relations used to analyze coordination and Multi-Word Expression (MWE), loose joining relations, special relations), as shown in Table 2.1. In addition, more specific syntactic relations (subtypes of the universal types) in individual languages are included in UD. Subtype relations always start with a basic type followed by a colon and a subtype string, e.g., *acl:relcl* (relative adnominal clauses), *csubj:pass* (clausal subjects of passive clauses).

The gold-annotated corpus used for this study is the Parallel Universal Dependencies (PUD).

			Common		
	acl	acl:relcl	advcl	advmod	appos
	aux	aux:pass	case	cc	ccomp
	clf	compound	conj	cop	csubj
	det	discourse	discourse:sp	dislocated	flat:name
	iobj	mark	mark:adv	mark:rel	nmod
	nmod	nsubj:pass	nummod	obj	obl
Zh	obl:patient	parataxis	punct	root	vocative
	xcomp				
			PUD (exclusive)		
	case:loc	dep	fixed	flat	mark:prt
	obl:agent	obl:tmod			
			GSD (exclusive)		
	compound:ext	csubj:pass	flat:foreign	mark:adv	nmod:tmod
	orphan	reparandum			
			Common		
	acl	acl:relcl	advcl	advmod	appos
	aux	aux:pass	case	cc	ccomp
	compound	conj	cop	csubj	det
	discourse	expl	fixed	flat	flat:foreign
Ru	flat:name	iobj	mark	nmod	nsubj
	nsubj:pass	nmod	nummod	nummod:entity	nummod:gov
	obj	obl	obl:agent	orphan	parataxis
	punct	root	vocative	xcomp	
			GSD (exclusive)		
	dep	dislocated	goeswith	list	
			Common		
	acl	acl:relcl	advcl	advmod	appos
	aux	aux:pass	case	cc	ccomp
	compound	compound:prt	conj	cop	csubj
	csubj:pass	dep	det	discourse	expl
	expl:pv	fixed	flat	iobj	mark
De	nmod	nmod:poss	nsubj	nsubj:pass	nummod
	obj	obl	orphan	parataxis	punct
	root	vocative	xcomp		
			PUD (exclusive)		
	cc:preconj	flat:name	obl:tmod		
			GSD (exclusive)		
	det:poss	goeswith	obl:agent	obl:arg	reparandum

Table 2.2: Differences in syntactic relations between the PUD and GSD corpus for the three languages. *Common* represents syntactic relations included in both corpus, *exclusive* means the syntactic relations exists only in the PUD or GSD corpus.

The version of the PUD is part of the treebank in the CoNLL 2017 shared task on Multilingual Parsing from Raw Text to Universal Dependencies (avoiding ambiguity with the UD treebank, the PUD corpus refers to the corpus used in the experiments). Specifically, the Chinese PUD, Russian PUD, and German PUD corpus are used for this study. There are 1,000 pairs of parallel sentences from the news domain and Wikipedia for each language, always in the same order. As illustrated in Figure 2.11, the first 750 sentences are initially in English, and the remaining 250 are originally from German, Italian, French, or Spanish. They are then translated into English

by professional translators before being translated from English to other languages, such as English→Chinese, English→Japanese, and English→Russian. The relevant morphological and grammatical annotations are annotated and converted by Google and UD community members.

Another one is Universal Dependencies GSD (GSD), which is a multilingual treebank converted from a basic SD style treebank (avoiding ambiguity with the UD treebank, the GSD corpus refers to the corpus used in the experiments). Similar to the PUD corpus, the Chinese, Russian, and German GSD are also the gold-annotated corpus used in this study. However, the number of annotated sentences in the GSD corpus differs between the three languages, with the Chinese GSD containing 4,997 sentences, the Russian GSD having 5,030 sentences, and the German GSD 15,590 sentences. There are also some differences in dependency relations between the PUD and GSD corpus due to the number of annotated sentences and the updated version of the UD annotation rules, as shown in Table 2.2.

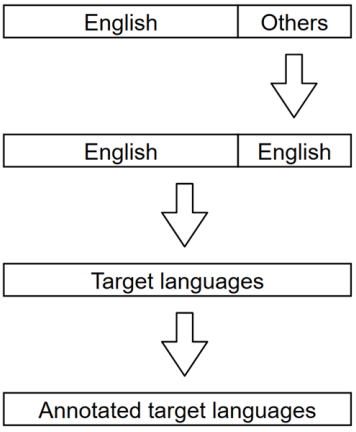


Figure 2.11: The process of PUD corpus construction.

2.3.3 Syntactic Strategies in Neural Machine Translation

The current mainstream NMT framework treats the source and target language sentences as strings for sequencing purposes. Despite making the model concise and facilitating the tokenization function, it may also cause much meaningful linguistic information to be lost. Therefore, many studies hope to improve the translation quality by modeling explicit syntactic knowledge (analyzed by parser) and integrating it with MT engines, drawing on relevant research syntactic findings based on SMT (Y. Liu et al., 2006; T. P. Nguyen et al., 2008; Xie et al., 2011).

Incorporating explicit syntactic knowledge of the source language sentence into NMT can help the encoder comprehend the meaning of the input sentences and the links between words more

precisely, offering more accurate representations to the decoder. Eriguchi et al. (2016) implemented a phrase-level attention mechanism by incorporating syntactic knowledge of the source language into an RNN-based translation model. The model encodes the source language sentence using an RNN and models the phrase structure of it using a tree-LSTM on top of the obtained hidden vectors. K. Chen et al. (2017) optimized the encoder of RNN-based NMT by adding a CNN to learn the source dependency representation vector for each word in the source language sentence, which is used as input to the decoder along with the word vector for the target word prediction, capturing the long-range dependency constraints. Bastings et al. (2017) proposed the use of a GCN to encode the dependency structure of source language sentences. The GCN takes the hidden vectors of the words in the source language sentence and uses them to generate hidden syntactic vectors for each word. These vectors are based on the dependency structure and can be given to the RNN as extra information about the source language sentence for the decoder to use.

Many studies also have been conducted to explore the incorporation of explicit syntactic knowledge into the decoder. Tu et al. (2017) suggested that the feature information of the source language sentence influences the accuracy of the translation result, while the features of the target sentence can guide the fluency of the translation. Nadejde et al. (2017) introduced combinatorial category grammar annotation to the decoder side of NMT. They also found that translation performance is further improved if syntactic knowledge is applied on the source and target side. A. N. Le et al. (2017) serialized the target language sentence through the syntax tree parsed by the SD parser to replace the original target language sentence for model training.

In recent years, the Transformer model has received much attention, and the approach of explicit syntactic knowledge incorporation also has steadily shifted from an RNN-based to a Transformer-based one. Parent-scaled self-attention (Pascal) mechanism (Bugliarello & Okazaki, 2020), a method that improves the Transformer model by integrating grammatical knowledge, which utilizes dependency information to improve translation quality through a novel, parameter-free, dependency-based self-attention mechanism. C. Ma et al. (2020) designed a Transformer model for MT that utilizes syntax-based structures on the source side. These structures are generated by the parser and incorporated into both the self-attention and positional encoding of the encoder. C. Wang et al. (2019) proposed an NMT engine based on self-supervised dependency syntax awareness in the Transformer model. By using supervised

learning to gather syntactic information from the source language sentence, the attention mechanism of the encoder is enhanced. This dependency-aware contextual feature allows the MT engine to generate improved translations.

In the past, researchers have often used RNN variants to model and represent syntactic knowledge in their studies (Eriguchi et al., 2016; K. Chen et al., 2017). However, RNN models suffer from vanishing and exploding gradients, which can increase the training cost of the MT engine. In addition, it is difficult for RNN models to process longer sequences, there may be information loss in modeling long input sequences and explicit syntactic knowledge of complex sentences. Moreover, RNN-based strategies are not efficient in parallelizing computation, which is necessary for meeting the requirements of Transformer-based MT engines. Although it is possible to incorporate explicit syntactic knowledge in the Transformer model (Bugliarello & Okazaki, 2020; Duan et al., 2020), they mostly model syntactic knowledge sequentially, which can lead to some tree-like syntactic knowledge not being explicitly included. A popular modeling strategy in the Transformer model is to use the self-attention mechanism to model and represent syntactic knowledge, which allows the model to focus on specific tokens. However, strong dependencies between tokens and explicit topological information are not explicitly modeled since the current token still performs attention weight calculation with other tokens. Also, whether attention mechanism in Transformer model can be used as explanation is still under discussion (Jain & Wallace, 2019; Serrano & Smith, 2019; Wiegreffe & Pinter, 2019). The Transformer model mostly deals with modeling and representing syntactic knowledge in terms of the proposed syntactic strategies, but its performance may restrict the best approach for utilizing this knowledge. To put it in perspective, similar to how an airplane engine cannot perform at its maximum potential when fitted into a car due to differences in architecture and materials, the speed of the car cannot match that of the airplane.

Studies have also revealed that using a pre-trained language model BERT along with the Transformer model can enhance translation accuracy (Zhu. et al., 2020; J. Yang et al., 2020). BERT can provide a more accurate representation of contextual information by utilizing contextual word embedding and general linguistic knowledge, enabling it to effectively model syntactic knowledge of input sentences for the encoder or decoder in the MT engine. Some studies have attempted to improve the performance outcomes of other downstream tasks by fusing explicit syntactic knowledge with BERT (K. Wang et al., 2020; L. Huang et al., 2020). However, such

a strategy has not been validated or discussed in the MT scenarios. Most studies only treat them (explicit syntactic knowledge or BERT) as a single data augmentation strategy in MT scenarios to provide additional knowledge under limited training sets. How the MT engine and translation quality are influenced by their fusion strategy and what interpretability in terms of syntactic knowledge can be drawn is still being determined.

2.4 Deep Learning for Graphs

2.4.1 Graph Neural Networks

Deep learning applies artificial neural networks to filter and extract features from the input data, which allows the classification and prediction of downstream tasks across various scenarios. Typically, neural networks such as Multi-layer Perceptron (MLP) (Hinton & Salakhutdinov, 2006), CNN (LeCun et al., 1989), RNN (Elman, 1990), Generative Adversarial Network (GAN) (Goodfellow et al., 2014) and Auto Encoder (AE) (Rifai et al., 2011) have become general-purpose network frameworks for problem-solving in many studies. Deep learning models, unlike shallow learning models such as statistical machine learning, utilize complex and multifunctional neural network architectures, which enables them to extract abstract and advanced information by analyzing features from shallow to deep layers. Deep learning has significantly improved CV and NLP tasks in recent years. This is due to the fact that image, text, audio, and video data have a consistent and standardized format, also referred to as Euclidean structure, where these types of data are comprised of nodes that follow specific arrangement rules and sequences, such as two-dimensional grids and one-dimensional sequences.

However, deep learning still has limitations in dealing with all situations and problems due to the diversity and complexity of data. There is a significant amount of non-Euclidean structural data in real-world applications, e.g., linguistic knowledge, knowledge graphs, social networks, and compound molecules. The non-Euclidean data structure lacks a defined alignment rule and a strict sense of sequential order between feature nodes compared with Euclidean data structure, as shown in Figure 2.12, which does not meet translation invariance. The feature matrix dimension of each block is not uniform; as illustrated in Figure 2.13, neural networks such as CNN cannot perform operations such as convolution and pooling on them directly. When it comes to understanding dependencies in syntactic structure, there is also a challenge. Many studies use linear input models like RNN and Transformer models to incorporate syntactic knowledge

into downstream tasks alongside other neural networks. Since language sentences are complex and different sentences contain different syntactic structures in terms of syntactic dependencies, where such knowledge has a non-Euclidean data structure, sequential processing and representations of them may not be able to account for all the interrelated linguistic information in MT scenarios.

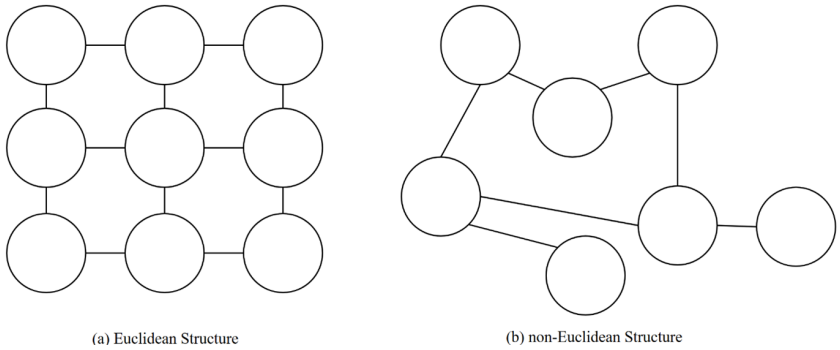


Figure 2.12: An illustration of how the two data structures differ from each other. The difference between data with Euclidean and non-Euclidean structures is that set rules of arrangement and sequence characterize the former.

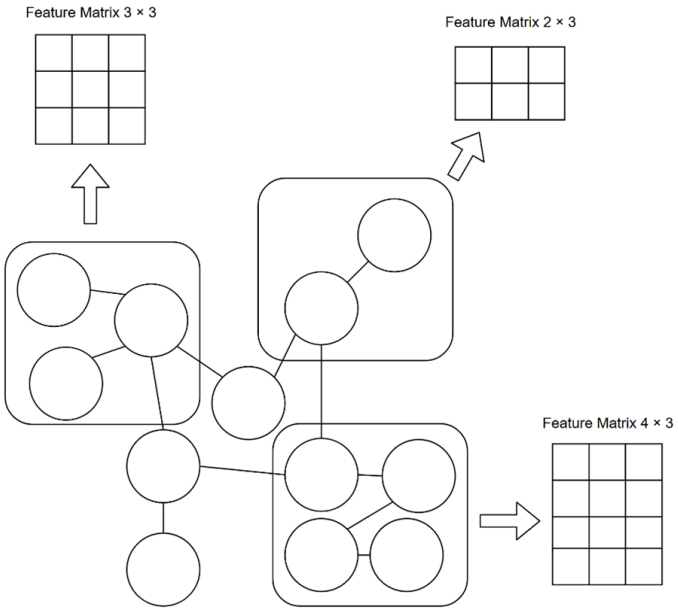


Figure 2.13: Due to irregularity, it is difficult for uniformly regularized convolutional kernel to handle non-Euclidean structured data.

A possible solution for this situation is to model non-Euclidean data as a graph structure. The graph structure can be irregular and sparse, with each node (each word in a sentence or a pixel in an image) having a flexible number of neighbors and dependencies on the graph without being constrained by its location in space. These natural structures and properties can effec-

tively capture the challenges posed by non-Euclidean data in deep learning. Gori et al. (2005) introduced the concept of Graph Neural Network (GNN), a neural network model specifically designed for processing graph-structured data. Using graphs allows for more accurate and flexible data modeling in real-world applications. Bruna et al. (2014) proposed the GCN, where CNN can be explicitly utilized to model graph-structured data. By combining the feature and label information of the central and neighboring nodes and putting it into the CNN, the GCN offers a regularised representation of each node in the graph. As a result, it can incorporate multi-scale information into higher-level representations, and their effective utilization of graph structure information provides a standard paradigm for migrating other deep learning neural network strategies to the graph.

In recent years, there have been numerous proposed approaches to redefine the concept of graph convolution, which can be categorized into two main groups. The first is the GNN based on the spectral approach (Defferrard et al., 2016; R. Li et al., 2018; Levie et al., 2018). The spectral approach introduces a signal processing method that defines the graph convolution by transforming the node features into the spectral domain to achieve convolution and then transforming them into the spatial domain. The initial GCN, based on the ChebyNet (Defferrard et al., 2016), reduced the number of parameters by characterizing the convolutional kernel in the spectral domain and approximating it using polynomial functions. This ultimately reduces the computational cost by eliminating the requirement for feature decomposition. R. Li et al. (2018) demonstrated more generalizable and flexible Adaptive Graph Convolution Neural Network (AGCNN), which can take arbitrary graph structures as input. Also, task-driven adaptive graphs are available at training time to make the model learn itself for different graph data.

The second one is the GNN based on a spatial approach (Veličković et al., 2017; Hamilton et al., 2017; Y. Li et al., 2018). The spatial domain-based graph convolutional neural network differs from the spectral domain one that starts from signal processing theory. Instead, it begins with the nodes in the graph, designs aggregation functions that collect the features of nearby nodes, and then employs a message propagation mechanism to effectively use the features of neighboring nodes to update the features of the represented central node. One of the representative works is GCN, a special case that belongs to both spatial and spectral methods. Graph Sample and Aggregate (GraphSAGE) was presented as a method for updating nodes Hamilton et al. (2017).

This is done by sampling a fixed number of neighbors for each node and then aggregating the information obtained. For every node, Graph Attention Network (GAT) (Veličković et al., 2017) calculates the hidden information for each node on the graph and uses an attention mechanism to handle data with an unknown graph structure, which can lead to improved results in node classification tasks.

More GNNs were subsequently proposed, e.g., Graph Auto-Encoder (GAE) (You et al., 2018; Z. Wu et al., 2019) and attention based spatial-temporal Graph Convolutional Network (ASTGCN) (Guo et al., 2019). GAE is a framework for unsupervised learning that encodes graphs into a latent vector space and reconstructs graph data using the encoded information. ASTGCN can learn implicit feature patterns from spatiotemporal graphs, taking into account both temporal and spatial dependencies of nodes. It has many applications, including predicting traffic speed and anticipating driver maneuvers.

2.4.2 Graph Attention Network with the Attention Mechanism

Neural network models with more parameters are more robust and can store more information, although this can lead to information overfitting. To address this, an attention mechanism can be introduced to help the neural network focus on the most critical information for the task and reduce attention to irrelevant details, thereby improving the efficiency and accuracy of task processing. The attention mechanism has unlocked new potential for RNN and Transformer models in NLP tasks and it has also sparked interest in incorporating the attention mechanism into GNN. Veličković et al. (2017) proposed GAT, which introduces the attention mechanism to the aggregation of neighboring nodes by a GNN. The attention mechanism allows the model to learn different features of each neighboring node and then selectively obtain information about neighboring nodes based on the attention weights when aggregating and updating features for the central node. Figure 2.14 and Equations 2.12, 2.13, 2.14, 2.15 summarise the working mechanism of the GAT.

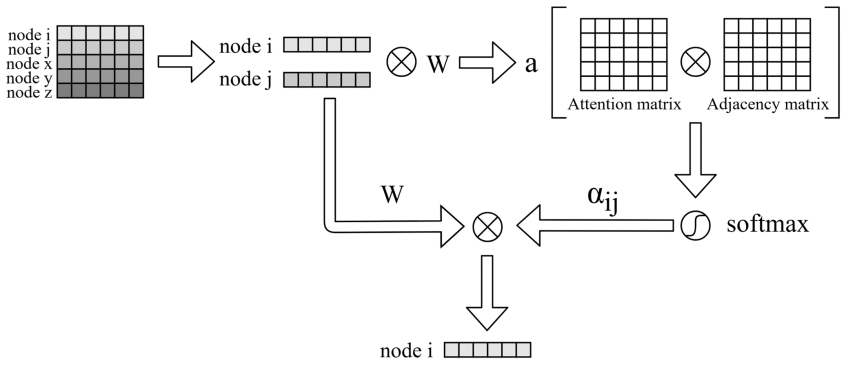


Figure 2.14: In order to update the current features of node i connected with node j, attention weights need to be computed using a learnable matrix via the feed-forward neural network, where the adjacency matrix defines the connections of nodes and the attention observation range of node i. The attention scores and the features of node j achieve the feature aggregation of node i.

$$\alpha_{ij}^{k} = \frac{exp(LeakyReLU(a^{T}[Wx_i \parallel Wx_j]))}{\sum_{v \in N_i} exp(LeakyReLU(a^{T}[Wx_i \parallel Wx_v]))} \tag{2.12}$$

$$h_i^{out} = \sigma\left(\sum_{j \in N_i} \alpha_{ij} W x_j\right) \tag{2.13}$$

$$h_i^{out} = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j \in N_i} \alpha_{ij}^{k} W^{k} x_j\right) \tag{2.14}$$

$$h_i^{out} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j \in N_i} \alpha_{ij}^{k} W^{k} x_j\right) \tag{2.15}$$

Equation 2.12 demonstrates how to determine the attention weight between the specified nodes. α_{ij} denotes the attention score of node i and node j. W is a linear transformation matrix. a^T denotes the weight vector parameter, $\parallel$ represents the concatenation of the vectors, and LeakyReLU is an activation function. Equation 2.13 indicates how node i uses an attention mechanism with one attention head to update its representation with its neighborhood.

The multi-head attention mechanism has also been applied to the GAT to make the model more robust. Based on Equation 2.12, the function that calculates the attention score using only one attention head can be changed to K attention heads. Each attention head yields a set of parameters and provides features for the subsequently weighted sum. In each layer of GAT, the K different attention heads do not affect each other and work independently. Equation

2.14 demonstrates the update mechanism for node i after applying multi-head attention, where the representation of node i is obtained by concatenating the results of each attention head. If multi-head attention is applied, the representation of node i is obtained by the averaged feature aggregation in the last layer of GAT, as shown in Equation 2.15. The overall formula for information propagation in GAT is $H_{l+1} = GAT(H_l, A; \Theta_l)$, where H_{l+1} denotes the hidden states of all input nodes at the last layer, and $A \in \mathbb{R}^{n \times n}$ is the given graph adjacency matrix, Θ_l is the model parameters.

Benefiting from the multi-head attention mechanism in GAT to capture inter-node features adaptively, syntactic knowledge can be better incorporated into neural networks for downstream tasks. K. Wang et al. (2020) applied the GAT model to encode the new dependency tree by reshaping and pruning the given dependency tree so that the aspect words are used as the root node of the dependency tree on the aspect-based sentiment analysis task. It is demonstrated that constructing syntactic trees via GAT allows the model to focus better on the connection between aspects and opinion words. N. Ma et al. (2020) proposed an ED-GAT model in comparative preference classification task that uses dependency knowledge represented by GAT and word embeddings to capture comparisons and classify preference orientations between two given entities. L. Huang et al. (2020) used the dependency tree structure via GAT and BERT to achieve syntactic awareness, which helps better to model the interactions between context and aspect words in aspect-level sentiment classification. The majority of research on GAT has centered around using it to model and represent important syntactic knowledge in downstream tasks. However, there has been a lack of exploration into what GAT knows such syntactic knowledge, even though it is possible to build it into graphs. Moreover, the impact of its model structure, which includes multi-head attention and model layers, on the acquisition of syntactic knowledge is unclear. What kind of syntactic knowledge is difficult for GAT to model and represent is still unknown and needs to be discovered. Having a clear understanding of syntactic knowledge can enhance the interpretability of GAT and enable a more efficient application of it in representing linguistic knowledge for downstream tasks.

2.4.3 Graph Neural Networks in Neural Machine Translation

Representing sentences and words as sequences in NLP tasks may result in topology information, such as tree-like syntax, being compressed or lost, which can pose challenges for downstream tasks that rely on a precise representation of source language sentence features, such as

speech recognition and MT. With its topological graph-based approach, GNN can create various linguistic graphs that convert different features in input text into nodes, edges, and graph representations. This capability is helpful for downstream tasks, as it can enable more effective pattern analysis and inference for input sentences in terms of linguistic knowledge, resulting in improved model performance.

While the encoder-decoder framework is typically used for NMT, the MT engine may not always accurately model the syntactic structure of the input sentences. Recent RNN-based models have demonstrated the possibility of integrating syntactic knowledge into MT through indirect means, which allows for the imposition of strict constraints on the relationship between syntax and translation during the modeling process (Luong et al., 2015; Eriguchi et al., 2016). However, the method of linear modeling for understanding sentence structure does not account for more dependencies between words in the sentence. In order to better incorporate useful syntactic information into MT, Bastings et al. (2017) implemented a syntactic GCN to construct the syntactic structure. To gain a clearer comprehension of the source language sentence, a CNN encodes each word and its contextual information. This helps to derive the implicit state of each word. The syntactic GCN then uses this implicit state as the input node and the syntactic dependency as an edge to iterate the GNN. As a result, the representation of each word is enriched with more relevant information. Sennrich. et al. (2016b) demonstrated that utilizing a GCN-based MT engine incorporating syntactic information on English-German and English-Czech datasets during the WMT16 task leads to a substantial enhancement in translation performance. The results based on the BLEU evaluation metric show that this approach significantly improves word order and lexical selection in MT engine. A simple and effective Recurrent Graph Syntax Encoder (RGSE) was developed for translation tasks (L. Ding et al., 2022). RGSE can be used with either RNN or transformer models. It treats RNN units as nodes and syntactic dependencies as edges, which allows the model to consider both syntactic dependencies and word sequence. Based on experiments, adding a graph syntax encoder leads to competitive translation performance and improved translation quality for longer sentences.

Besides syntactic information, the semantic representation of a sentence on the graph also enables the MT engine to concentrate on specific details of the source language sentence while creating the target language sentence, which improves the generalization ability of the neural networks. Marcheggiani et al. (2018) used semantic dependency role analysis to obtain the

predicates, arguments, and the semantic role relationship among them. The predicates and arguments are used as nodes of the graph and the edges of the semantic role relations to build the graph structure for GCN and encode the semantic structure to incorporate into the MT engine. To better use other structured semantic information in the sentence, Song et al. (2019) suggested using a semantic MT engine incorporating abstract meaning representation as a source of knowledge. This knowledge is a semantic paradigm that represents the meaning of a sentence through a directed graph. To introduce semantic information into MT, they use GNN to encode the abstract meaning representation graph structure of the sentence and input the obtained graph representation to the decoder in combination with the sentence encoding. Yin. et al. (2020) addressed the efficiency of the multimodal MT engine for utilizing semantic knowledge between pictures and text by using multimodal graphs to capture semantic relations. As a result, the translation performance is improved due to the deep semantic link between text and images through an attention mechanism provided to the decoder.

Many studies explore the feasibility of utilizing explicit linguistic knowledge on graphs in MT tasks. However, there has been limited research on how attention-based mechanisms in GAT can model syntactic knowledge in MT tasks. Furthermore, the case of fusion strategies for modeling and expressing syntactic knowledge through GAT and pre-trained language models, such as BERT, is not discussed, and it is not clear whether translation quality benefits from explicit topological linguistic knowledge on the graph and implicit linguistic knowledge in the pre-trained language model. There is still not enough exploration on how an MT engine can recognize syntactic dependencies in the source sentences if such a strategy is applied, and how the topological syntactic knowledge can be utilized to guide the knowledge inside BERT. The increased interpretability of GAT in terms of syntactic knowledge helps to better apply the fusion of linguistic knowledge on the graph and pre-trained language models in MT tasks, including but not limited to BERT.

Chapter 3

Syntactic Awareness of BERT via Universal Dependencies in MT Scenarios

3.1 Introduction

A given source language sentence in an NMT engine is converted into a target language sentence via an encoder-decoder architecture to achieve sequence transformation. By performing sequence transformation in parallel, the Transformer model, one of the most popular neural network architectures in MT tasks, improves modeling efficiency and translation performance by implementing the self-attention mechanism for feature extraction and sentence modeling. The self-attention mechanism helps the model understand more accurate contextual information in sentences and allows the model to capture different implicit linguistic knowledge, such as syntactic and semantic information (Raganato & Tiedemann, 2018; Voita et al., 2019). However, its inclusion does not allow the Transformer model to fully achieve effective modeling and learning of deep linguistic knowledge. Due to the extreme dependence of the MT engine on bilingual data and the absence of explicit prior linguistic knowledge as a guide, the neural network may overlook some necessary syntactic knowledge and produces poor-quality translations such as not fluent or violating syntax in the target language (Z. Zhang et al., 2020; X.-P. Nguyen et al., 2020).

One potential solution is to use large-scale monolingual data to place more emphasis on linguistic features in MT engines, thereby alleviating the knowledge-poor scenarios under limited bilingual data. Many researchers have tried to improve the NMT engine by utilizing the pre-trained language model BERT since it uses self-supervised learning to gain knowledge from a large corpus beforehand and employs the Transformer model framework. The BERT-based NMT engine allows the rich monolingual knowledge in BERT to be used by the NMT engine through fine-tuning (Imamura & Sumita, 2019b; Zhu. et al., 2020; Z. Zhang et al., 2021). For instance, BERT can be used for source language encoding to produce better source language sentence representations. It also can strengthen the link between the source and target languages on the target language side. Most studies focus on the performance improvements of the MT engine with the incorporation of BERT via a sentence-level evaluation metric called the BLEU score, which considers the quality of translation in terms of word string matching. Additionally, the purpose of some studies is to design probing tests to investigate BERT in order to improve interpretability from syntactic knowledge. Lin et al. (2019) discovered that BERT has hierarchical encoding and knowledge of specific syntactic trees. Tenney et al. (2019) found that parts of speech and syntax information can be encoded by BERT embeddings.

However, they mostly discuss vanilla BERT after pre-training and do not discuss MT scenarios for fine-tuning. Besides, there is a lack of discussion on the performance and impact of BERT applied to MT tasks from the linguistics perspective, such as syntactic knowledge. Although the BLEU score can quickly evaluate the performance of the MT engine, it does not consider linguistics such as rich morphology, sentence semantics, and sentence structure between the source and target languages. The impacts of the BERT-based MT engine on translation quality and the types of source language sentence structures that can pose challenges for these engines are not well understood. Despite its usefulness to MT engines, it is currently under investigation whether BERT is effective in acquiring and understanding more syntactic knowledge through fine-tuning MT tasks. There is also a need for interpretable investigations on what syntactic knowledge of source language sentences is difficult for BERT to detect in MT scenarios and whether there is a link between syntactic knowledge detected in BERT and translation quality.

In order to address the above questions of what BERT knows syntactic knowledge in MT scenarios, this research first constructs the three MT engines with corresponding BERT versions as the encoder, where the source languages are Chinese, Russian, and German, and the target

language is English. The study in this chapter also proposes two methods to evaluate the performance of syntactic knowledge in BERT after MT fine-tuning, as well as its impact on translation quality. The first experiment examines the syntactic dependency understanding of BERT fine-tuned by MT scenarios through syntactic probing experiments based on the corpus with gold syntactic annotations, in which BERT is an independent model in the experiment. Given that the BLEU score does not take into account sentence structure, morphology, and even meaningless sentences with correct phrases but in the wrong order can get a high score. In this study, the BLEU score is utilized solely as a fundamental evaluation metric to promptly verify the efficacy of the MT engines. The later experiments use a state-of-the-art Quality Estimation (QE) model and the corpus with gold syntactic annotations to detect the link between the syntactic knowledge of the source language and the translation quality of the target language on a sentence level, where the whole MT engine with BERT is considered as a target of the investigation. The main contributions of this chapter are shown below:

- The experiments present syntactic probing experiments to examine what syntactic knowledge of the source language side BERT acquires before and after fine-tuning the three MT directions. Based on the results, probes have identified syntactic performance when BERT predicts dependency relations and three syntactic patterns within BERT. When MT fine-tuning is applied to BERT, in most cases it results in decreased performance in F1-score for dependency relation prediction and does not significantly change its syntactic patterns compared to before the MT fine-tuning. The experiments also indicate that the inherent syntactic dependencies of BERT are already formed in pre-training and that MT fine-tuning is unlikely to make it reconstruct a new syntactic knowledge system.

- The experiments show a correlation between translation quality and syntactic dependencies that syntactic dependencies of the source language sentences are also one factor determining the translation quality, which the chi-square test can verify in three MT directions. Moreover, the experiment clarifies the syntactic structures defined by dependency relations of the source language sentences that MT engines fail to detect. They lead to low-quality translations, although they can be well-predicted in syntactic probing experiments conducted on BERT. The prediction scores identified by probes in syntactic probing experiments also have some associations with such dependency relations contributing to low-quality translations, e.g., the F1-score of such dependency relations in the middle

layer in BERT is usually higher than in the top layer.

3.2 Construction of Translation Engines

The architecture of the MT engine is similar to a study in MT (Imamura & Sumita, 2019a), in which the MT engine employs the BERT-base model as its encoder, and the decoder specification is identical to that used in the vanilla Transformer model, as shown in Figure 3.1. Three MT engines are built, where Chinese (Zh), Russian (Ru), and German (De) are used as the source language of each MT engine, and English (En) is the target language for them. Different BERT-base versions are applied for each source language. Specifically, Chinese uses BERT-wwm-ext[1], Russian is RuBERT[2], and German uses bert-base-german[3]. Although their model structures are identical, they have different pre-training strategies. BERT-www-ext uses a whole word masking strategy, which differs from vanilla BERT in that it masks the whole Chinese character phrase instead of an individual character. RuBERT is pre-trained for Russian, it is based on the initialization of multilingual BERT, however. Moreover, bert-base-german is the German version of the vanilla BERT. The three different languages and versions of BERT aim to investigate the phenomenon of inherent syntactic knowledge exhibited under the structure of the BERT model in the MT scenarios and the impact of a BERT-based MT engine on translation quality. Given a source language sentence $S_w = \left\{w^1, w^2, w^3, ..., w^n\right\}$ with sentence length n, it is then processed and vectorized by WordPiece tokenizer, as shown in Equation 3.1, and fed into BERT to obtain the final contextual representation h_B.

$$S_B = \left\{[CLS], w^1, w^{2\#1}, w^{2\#2}, w^3, ..., w^n, [SEP]\right\} \tag{3.1}$$

where $w^{n\#k}$ denotes the k-th subword of w^n, $[CLS]$ and $[SEP]$ are special tokens for defining a sentence in BERT.

For a target language sentence $T_w = \left\{w^1, w^2, w^3, ..., w^m\right\}$ of sentence length m, the word embedded sequence $T_D = \left\{w^1, w^2, w^3, ..., w^m\right\}$ is obtained after vectorization and positional encoding. The masked multi-head attention layer of the decoder computes the target sequence contextual features and obtains the representation h_M in Equation 3.2.

[1] https://huggingface.co/hfl/chinese-bert-wwm-ext
[2] https://huggingface.co/DeepPavlov/rubert-base-cased
[3] https://huggingface.co/bert-base-german-cased

$$h_M = MultiHead(T_D, T_D, T_D) \tag{3.2}$$

$$h_D = MultiHead(h_M, h_B, h_B) \tag{3.3}$$

In the decoder, the encoder-decoder multi-head attention layer calculates how the linguistic features of the source and target languages are related. The source language representations h_B are from BERT, which acts as the encoder, while the masked multi-head attention layer in the decoder provides the target language representations h_M, as in Equation 3.3. The final representation of the encoder and decoder h_D is fed into the feed-forward neural network for feature mapping and transformation, the probability distribution of the predicted words is then calculated by the softmax layer.

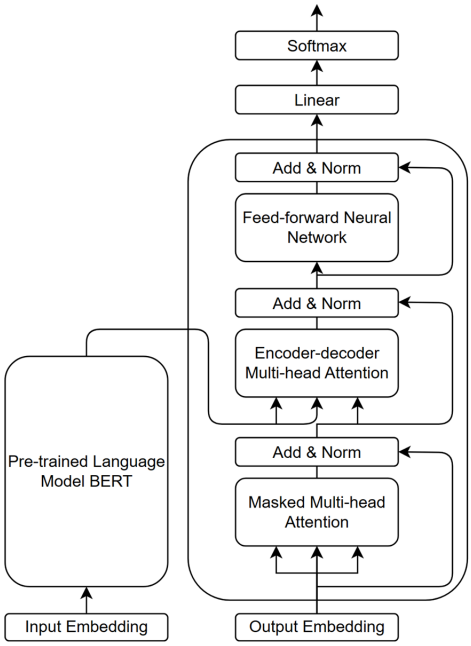


Figure 3.1: BERT-based MT engine for three MT directions where BERT is the encoder and the decoder is based on the vanilla Transformer model.

Construction of the MT engines uses the United Nations Parallel Corpus (UNPC)[4] as the data set for Chinese and Russian translation scenarios. UNPC has six parallel corpus containing Arabic, Simplified Chinese, English, French, Russian, and Spanish. These parallel corpus consist of official and other proceedings of United Nations conferences and meetings in multiple domains between 1990 and 2014. 1.2M parallel sentence pairs are randomly selected as the training set

[4] `https://opus.nlpl.eu/UNPC.php`

for Chinese to English (Zh→En) and Russian to English (Ru→En), and 6,000 parallel sentence pairs as their validation and test sets, respectively. Since UNPC does not provide a corpus for German, the European Parliament Proceedings Parallel Corpus (Europarl)[5] is another data set for the German to English (De→En) translation scenario. Europarl is a parallel corpus containing several European languages and consists mainly of the proceedings of the European Parliament. It is also divided into training, validation, and test sets. The number of their parallel sentence pairs is consistent with the Zh→En and Ru→En translation directions.

The BERT-base model for each source language has the same model structure, model layers = 12, attention heads = 12, and embedding dimension = 768. All BERT-base models in the experiments are fine-tuned by MT to update the internal parameters to achieve the best performance, where the MT engine learning rate = 2e-5 and batch size = 8, Adam as the optimizer, and cross entropy as the loss function. All translation engines are trained on RTX 3080 GPU. The BLEU score is used as a metric to evaluate the basic performance of the translation engines.

3.3 Investigation of Syntax in BERT

A syntactic probing experiment is proposed to investigate BERT acquisition of syntactic knowledge in Chinese, Russian, and German sentences before and after MT fine-tuning. In this study, the dependency relations of the source language sentences are the syntactic knowledge this experiment is concerned with, where dependency relations specify the syntactic structure of sentences and describe the dependencies between words. In this experiment, syntactic probing experiments are considered a sequence labeling task. BERT, both before and after MT fine-tuning, serves as the subject of the experiment. Its task is to predict dependency relations, which are the labels indicating relationships in a dependency connection between dependents and heads, for tokens within sentences when they function as dependents in a dependency connection, as defined by gold standard syntactic annotations. However, cues from the tokens when they serve as the head of the dependency relation would neither be provided nor required to be predicted. This experiment also differs from other early studies that use multiple algorithms or complicated probes to identify knowledge in BERT. The probes utilized in those studies are too complex to determine whether the knowledge of syntax is solely due to BERT or those

[5] `https://opus.nlpl.eu/Europarl.php`

complicated probes. Moreover, feed-forward neural networks or shallow classifiers are the functions used to receive representations from BERT in an MT engine. The complex probes do not match how BERT actually works in MT scenarios to reflect what it knows about that knowledge correctly. To ensure that the syntactic knowledge in BERT is probed as much as possible and to simulate BERT in MT scenarios inspired by Papadimitriou et al. (2021), a linear classifier as the probe is added and applied to each layer of BERT, indicating that specific layers are evaluated, as shown in Figure 3.2. Each token in a sentence has corresponding dependency relations to indicate the syntactic constituents of that token in the sentence. BERT needs to predict each of these dependency relations in the given sentence before and after fine-tuning the MT task via a shallow classifier.

In order to ensure the correctness of the syntax gathered by BERT, the PUD and GSD corpuses with gold syntactic annotations are used in this syntactic probing experiment. The PUD corpus contains 1,000 sentences with gold syntactic annotation for each source language (Chinese PUD[6], Russian PUD[7] and German PUD[8]). Though they are both universal dependencies, there are some differences in dependency relations between the GSD and the PUD corpus, as well as the number of annotated sentences in each corpus (more details are in Chapter 2.3). The Chinese GSD[9] and Russian GSD[10] corpus have about 5,000 sentences each, and the German GSD[11] corpus is much larger with about 15,000 sentences. Dependency relations with a lower number are removed from the experiments to avoid inaccurate results.

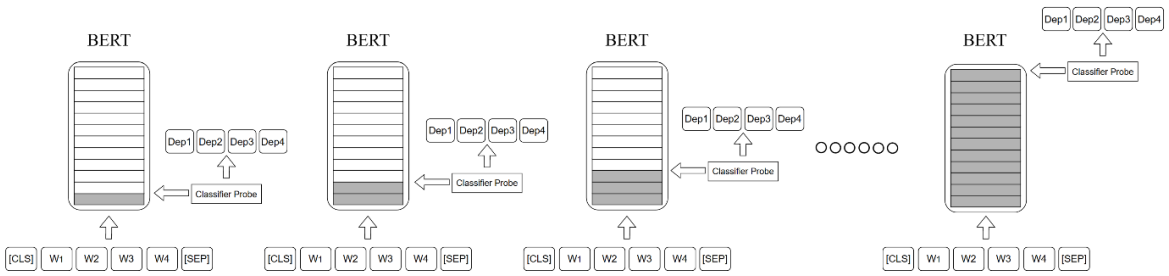


Figure 3.2: Syntactic probing experiments detect what BERT knows about the syntactic knowledge. Probes are spread across each layer of BERT, and BERT needs to use a limited number of layers to predict the corresponding dependency relation of each input token.

The predictions made by BERT for the dependency relations of each language are separated into

[6] https://github.com/UniversalDependencies/UD_Chinese-PUD
[7] https://github.com/UniversalDependencies/UD_Russian-PUD
[8] https://github.com/UniversalDependencies/UD_German-PUD
[9] https://github.com/UniversalDependencies/UD_Chinese-GSDSimp
[10] https://github.com/UniversalDependencies/UD_Russian-GSD
[11] https://github.com/UniversalDependencies/UD_German-GSD

two groups: before MT fine-tuning and after MT fine-tuning. Both groups include tests from the PUD and GSD corpus. When conducting probing experiments on either a PUD or GSD corpus, a training set, validation set, and test set are constructed. The training set contains 70% of sentences, while the validation and test sets contain 15% each to avoid model overfitting and effective result confirmation. The surface classification layer parameters are updated, while all BERT parameters are frozen without updating the syntactic knowledge from the training corpus. The evaluation metric is F1-score, and learning rate = 1e-3, the word embedding = 768, optimizer = Adam.

3.3.1 Experimental results

Language	Data set	BLEU
Zh→En	UNPC	56.34
Ru→En	UNPC	55.85
De→En	Europarl	38.06

Table 3.1: BERT-based MT engines are built for three different source languages, and the BLEU score indicates the proper functioning of the MT engines.

The three MT engines created for each source language produce English translations with different BLEU scores, as shown in Table 3.1. The BLEU score for De→En is relatively lower than for Zh→En and Ru→En, where the difference can be attributed to the different datasets or the version of BERT used. BERT is then examined separately from the MT engines for syntactic probing experiments (more details are in Appendix Sec A.1). It has been observed that BERT models understand syntactic dependencies of the source language sentences differently and can be classified into three syntactic phenomena based on the findings from the probes, as shown below.

- If BERT struggles to learn a given dependency relation, adding more layers to BERT does not result in significant improvement.
- If BERT is proficient in learning a given dependency relation, adjusting the number of BERT layers or fine-tuning the MT task can preserve its ability to make accurate predictions.
- The number of layers in BERT can affect specific dependency relations, leading to noticeable differences in prediction performance between layers.

Jawahar et al. (2019a) have shown that the middle layers of BERT contain more syntactic

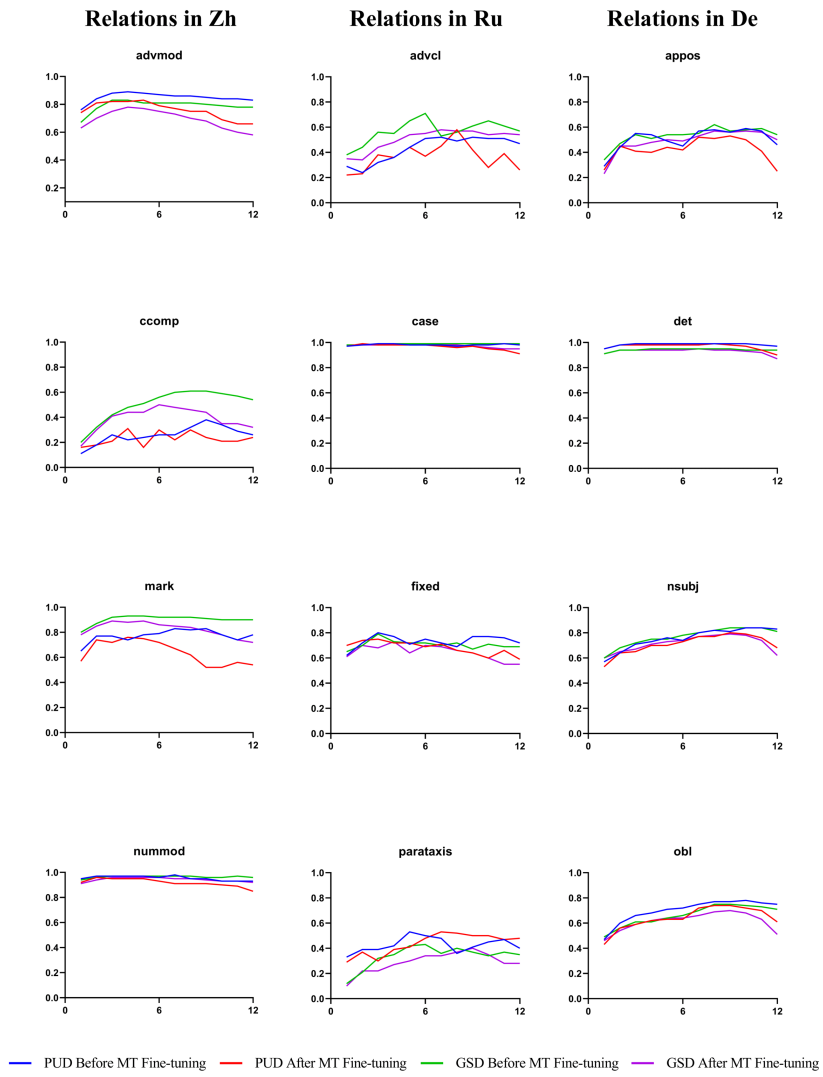


Figure 3.3: Randomly selected examples of syntactic probing experiments are presented in three languages: Chinese (Zh), Russian (Ru), and German (De). These examples serve to illustrate various experimental results and reinforce the conclusions drawn in the descriptions of this research. The x-axis denotes the number of layers of BERT, and the y-axis denotes the F1-score.

knowledge and that the performance of the middle layer in terms of syntactic knowledge is better. Syntactic probing experiments show that the performance of BERT before and after MT fine-tuning for more detailed dependency relations in syntactic knowledge depends not only on its number of layers but also on the type of dependency relations as a contributing factor. As illustrated in Figure 3.3, when BERT encounters the dependency relations it prefers, going through different layers of BERT does not significantly harm the prediction performance, e.g., *nummod* (numeric modifier) for Zh, *case* (case marking) for Ru, and *det* (determiner) for De, BERT still obtains robust prediction scores regardless of the number of layers. On the contrary, dependency relations such as *ccomp* (clausal complement) in Zh, *parataxis* (place side by side)

in Ru, and *appos* (appositional modifier) in De do not achieve improved prediction performance, although the number of layers of BERT is increased.

During the syntactic probing experiments on the three languages, the prediction results of BERT for the PUD and GSD corpus are similar before and after fine-tuning for the MT task. After fine-tuning with the MT task, the performance of BERT to predict most dependency relations has decreased, especially the prediction score in the top layer of BERT. Only a small portion of them have shown improvement or been maintained. Additionally, BERT exhibits different syntactic trends when predicting dependency relations in different languages. It is referred to as a syntactic pattern to differentiate it from the previously mentioned syntactic phenomenon. The common syntactic patterns in BERT are shown below.

- Smooth: There is no significant difference in the syntactic prediction performance between layers, as it remains relatively similar and stable.
- Climb + Decline: As the number of layers increases, the syntactic prediction performance gradually increases and finally plateaus or gradually decreases. The performance of each layer fluctuates less.
- Fluctuate: There is a significant difference in the syntactic prediction performance between the layers, although the overall trend is observable.

As shown in Figure 3.4, as the number of layers of BERT increases, the prediction performance curves, such as *nummod* (numeric modifier) for Zh, *cc* (coordinating conjunction) for Ru, and *det* (determiner) for De, are very smooth. The difference in the number of layers does not bring any significant performance difference, and BERT shows relatively high prediction performance for such dependency relations. Some dependencies, such as *conj* (determiner) for Zh, *obj* (object) for Ru, and *nmod* (nominal modifier) for De, have their prediction performance curves increase gradually with the number of layers of BERT. Their prediction performance curves peak in the middle layers of BERT and then level off or decrease in the higher layers. The results are comparable to those discovered in (Jawahar et al., 2019a), which confirmed that the middle layers of BERT are conducive to syntactic knowledge. However, the prediction performance curves for some dependencies fluctuate, and the prediction performance curves vary dramatically between layers, such as *nsubj:pass* (passive nominal subject) for Zh and Ru, and *parataxis* (place side by side) for De. There appears to be a discrepancy in the definition of abstract syntactic

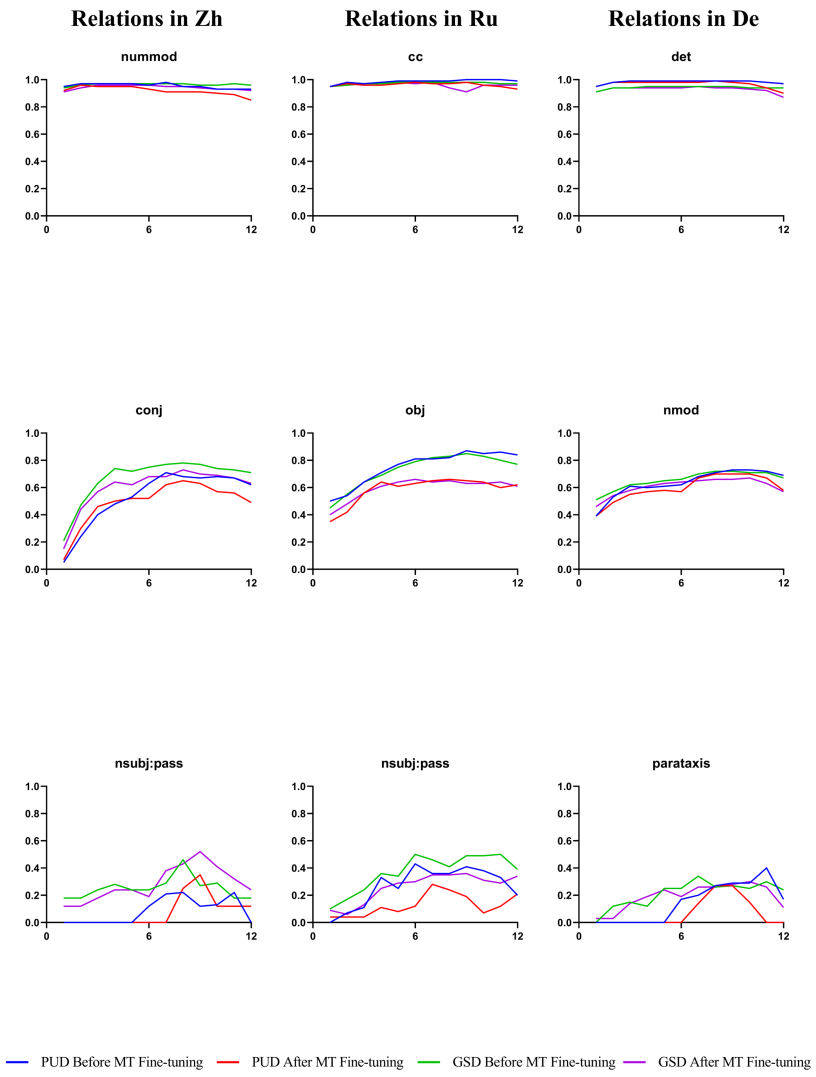


Figure 3.4: Prediction performance of BERT in identifying dependency relations is compared across different syntactic patterns in each language. The selection of these dependency relations is based on their representation of typical syntactic patterns. Arranged from top to bottom, they are categorized as *Smooth*, illustrating a consistent and even pattern, *Climb + Decline*, depicting a pattern that rises and then either plateaus or falls, and *Fluctuate*, signifying a pattern with significant difference changes. The x-axis denotes the number of layers of BERT, and the y-axis denotes the F1-score.

knowledge in BERT and golden syntactic knowledge from annotated corpus, or BERT may not have a clear concept of syntactic knowledge. Besides, these more significant fluctuations tend to occur more in the middle layers of BERT, which further confirms the greater sensitivity of the middle layers to syntactic knowledge.

Based on predictions made for the three source languages, BERT displays consistent syntactic patterns for particular dependency relations (more details are in Appendix Sec A.1). As an example, it has been observed that the syntax patterns across the three languages for *conj* (de-

terminer) in BERT are *Climb + Decline*, and syntax patterns for *cc* (coordinating conjunction) are *Smooth*, with both *conj* and *cc* belonging to coordination in UD definition. Similarly, BERT also shows consistent syntactic patterns across the three languages for *csubj* (clausal subject), *ccomp* (clausal complement) and *xcomp* (open clausal complement), which are defined as core arguments and clauses in the UD definition. The syntactic patterns for *csubj* are *Fluctuate*, and *Climb + Decline* for both *ccomp* and *xcomp*. Table 3.2 demonstrates that the dependency relations in bold have the same syntactic patterns in BERT for the syntactic probing experiment across all three languages. These dependency relations are mainly concentrated in the core arguments-based functional category, where 5 out of 6 relations appear to have the same syntactic pattern for their own experiment across different languages. The same trend can be observed in other functional and structural categories. This suggests that BERT has the ability to create a coherent learning process for dependency relations across languages, resulting in a consistent syntactic pattern, despite different pre-training strategies and pre-training on monolingual data from various languages.

	Nominals	Clauses	Modifier words	Function words
Core arguments	**nsubj** **obj** iobj	**csubj** **ccomp** **xcomp**		
Non-core dependents	**obl** vocative expl dislocated	advcl	advmod discourse	aux cop mark
Nominal dependents	**nmod** **appos** nummod	acl	amod	det clf case
Coordination	MWE	Loose	Special	
conj **cc**	fixed flat compound	list parataxis	orphan goeswith reparandum	punct **root** dep

Table 3.2: For the syntactic probing experiment, the bolded individual dependency relations follow the same syntactic patterns in all three languages in BERT.

Furthermore, the prediction scores of the MT fine-tuned BERT for dependency relations are reduced in most cases (more details are in Appendix Sec A.1), e.g., *advmod* (adverbial modifier) and *mark* (marker) in Zh, *advcl* (adverbial clause modifier) and *fixed* (fixed multiword expression) in Ru, and *nsubj* (nominal subject) and *obl* (oblique nominal) in De, as shown in Figure 3.3. However, their syntactic patterns are consistent before and after MT fine-tuning, where they do not change from one to the other. Both the PUD and GSD corpus as data sets present similar findings. The MT engine can somehow benefit from the implicit syntactic knowledge

in BERT. However, the fine-tuning of the MT task does not fundamentally alter how BERT predicts dependency relations, suggesting that BERT has developed relatively settled syntactic patterns or systems during pre-training. The MT fine-tuning does not directly increase the syntactic knowledge in BERT. If one wants to emphasize the importance of syntactic knowledge more in the MT scenarios, purposefully training BERT from scratch or using other neural networks to model and represent syntactic knowledge are feasible alternatives.

3.4 Investigation of Translation Quality

To investigate whether syntactic dependencies of source language sentences affect the translation quality of NMT, MT engines for each of the three languages translate the PUD corpus of their corresponding source languages. BLEU score is not used as an evaluation metric for MT output since it does not consider factors such as paraphrasing, synonymy, and sentence fluency, and most translation scenarios do not provide reference translations. Instead, a state-of-the-art QE model (Ranasinghe et al., 2020) for translation quality assessment is adopted. It considers the semantic relationship between the source language sentence and the translation, fluency, word order rationality, and other factors to provide a QE score that reflects the translation quality. The QE score is scored from 0 to 1, and the higher the QE score, the higher the translation quality of the target sentence. The translation quality is ranked according to the QE score from highest to lowest. Translations with QE scores in the top 20% are rated as high-quality translations, and those with the lowest scores in the bottom 20% are considered low-quality translations. The dependency relations and quantities in the source language sentences of the high-quality and low-quality translations are then recorded, and chi-square tests are applied to perform the following investigations.

(a) A chi-square test of independence is conducted to investigate whether syntactic dependencies of source sentences are related to translation quality. Different levels of translation qualities (high-quality and low-quality translations) and their corresponding dependency relations of source language sentences are treated as variables. The test statistic χ^2 reflects the association between syntactic dependencies of source language sentences and translation quality under degree of freedom and significance level.

(b) A test based on the chi-square goodness of fit test to clarify the relationship between the number of dependency relations in high-quality and low-quality translations. The observed

values in the chi-square test are the given number of dependency relations in high and low-quality translations. Low-quality translations are expected to reach the level of high-quality translations; therefore, the experiment assumes that the expected value of low-quality translations is the observed value of high-quality translations, and the expected value of high-quality translations is the observed value of low-quality translations to ensure the equality of expected and observed frequencies. The experiment only discusses the square of the difference in low-quality translations divided by the expected frequency, describing the difference and gap in the number of dependency relations between low-quality and high-quality translations based on the expected value. This is not a complete chi-square goodness of fit test since the square of the difference between high-quality translations divided by the expected frequency is not calculated and discussed. Since the experiment is based on the chi-square test approach, χ^2 is still used to refer to the test statistic.

3.4.1 Experimental results

Languages	Dependencies	df	Significance level	P-value	Critical value	Test statistic χ^2
Zh	32	31	0.05	$p < 0.05$	44.98	171.4
Ru	29	28		$p < 0.05$	41.34	154.9
De	30	29		$p < 0.05$	42.56	182

Table 3.3: The number of dependency relations in high and low-quality translations is calculated in the chi-square test of independence.

(a) Dependency relations of the three languages are counted and calculated for high-quality and low-quality translations. Table 3.3 shows that the test statistic value for Zh is significantly higher than the critical value at a significance level of 0.05, which is also the case for Ru and De. It means that the Null Hypothesis (H_0) that translation quality and syntactic dependencies in source language sentences are unrelated is disproved. The Alternative Hypothesis (H_1) that translation quality varies according to the syntactic dependencies of source language sentences and that syntactic dependency is indeed an essential factor affecting translation quality is accepted. It also provides a statistical interpretation of why the previous studies on incorporating syntactic knowledge in MT tasks can improve translation quality (Bugliarello & Okazaki, 2020; C. Wang et al., 2019).

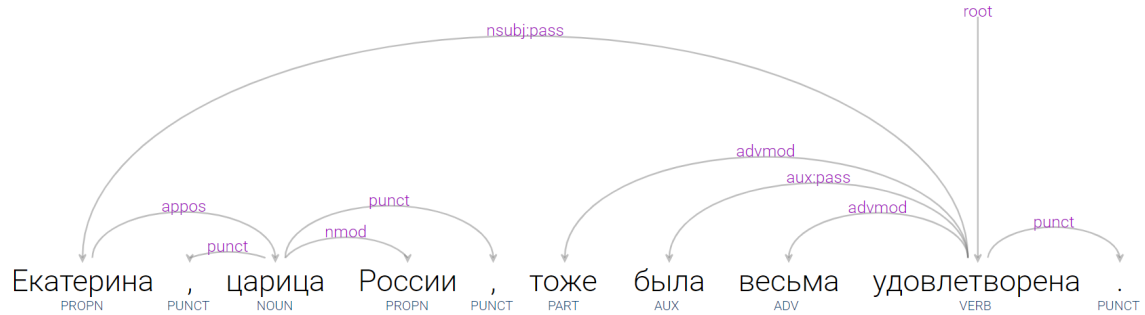
(b) The values of χ^2 are ranked from high to low for the three source languages alone with its corresponding dependency relations, as shown in Table 3.4. The higher value of χ^2 proves that the number of that dependency relation differs more between high and low-quality translations,

	Dependency	Quality			F1-score	
		High	Low	χ^2	Layer-6	Layer-12
Zh	**flat:name**	**1**	**53**	**2704**	0.68	0.78
	appos	**10**	**73**	**396.9**	0.40	0.48
	flat	**2**	**24**	**242**	0.70	0.74
	dep	42	99	77.3	0.28	0.29
	advcl	62	113	41.9	0.43	0.33
	mark	32	66	36.1	0.72	0.54
	nsubj	293	380	25.8	0.63	0.60
	obl	**86**	**132**	**24.6**	0.39	0.31
	case	**214**	**286**	**24.2**	0.82	0.76
	obl:tmod	28	54	24.1	0.68	0.46
	compound	267	333	16.3	0.70	0.60
Ru	flat:foreign	1	55	2916	0.69	0.18
	flat	**1**	**31**	**900**	0.22	0.18
	flat:name	**12**	**57**	**168.7**	0.82	0.86
	appos	**8**	**31**	**48.3**	0.36	0.47
	obl	**207**	**307**	**66.1**	0.71	0.66
	parataxis	21	50	48.3	0.48	0.48
	case	**306**	**406**	**40**	0.98	0.91
	conj	93	147	32.6	0.63	0.72
	cc	82	123	31.3	0.98	0.93
	amod	274	347	20.5	0.90	0.89
	nummod:gov	10	20	19.4	0.67	0.44
De	**flat**	**0**	**9**	**-**	0.25	0
	appos	**10**	**96**	**739.6**	0.42	0.25
	flat:name	**6**	**61**	**504.1**	0.52	0.39
	compound	25	72	88.3	0.47	0.51
	obl	**212**	**327**	**62.3**	0.63	0.61
	compound:prt	10	34	57.6	0.92	0.48
	case	**324**	**459**	**56.25**	0.97	0.84
	obl:tmod	14	34	28.5	0.55	0.62
	nmod:poss	39	67	20.1	0.96	0.85
	nsubj	241	308	18.6	0.73	0.68
	advcl	27	47	14.8	0.34	0.36

Table 3.4: The top-11 dependency relations for each language are demonstrated according to the value of χ^2. The bolded ones are the relations common to all three languages frequently appearing in low-quality translations. High-quality translations of De do not contain "flat", and thus the value of χ^2 cannot be calculated. F1-score is derived from a probing experiment where BERT fine-tuned for the MT task and PUD corpus as the data set.

and low-quality translations have to be significantly improved to reach the level of high-quality translations. For all three languages, five common dependency relations occur more frequently in low-quality translations: *appos* (appositional modifier), *case* (case marking), *flat* (flat multiword expression), *flat:name* (names), and *obl* (oblique modifier). Consider the scenario where a sentence in Ru needs to be translated into En, as shown in Figure 3.5. The MT engine translates the second half of the Ru sentence acceptably, the noun phrases at the beginning of the sentence containing the dependency relation called *appos* are not translated correctly, however. It is also found that BERT does not yield low prediction scores for these five dependency relations in

syntactic probing experiments. However, the MT engine loses detection of them, resulting in meaningless and non-fluent translation outputs. One conjecture is that BERT learns syntactic dependencies differently when it is a standalone model and when part of the MT engine, where multiple neural networks may also be filtering different syntactic knowledge, even though it may be important for BERT. More importantly, when BERT is used as an encoder in the MT engine, most dependency relations are more pronounced in the number of low-quality translations compared to high-quality ones. It reveals two potential problems contributing to low-quality translations: certain dependency relations, such as *appos* and *flat*, can hinder the understanding of an MT engine to source language sentences, and source language sentences with longer lengths may also pose a challenge to an MT engine due to their higher number and more complex syntactic dependencies.



MT herat , her country ' s russian federation , was also very pleased .
Reference Catherine of Russia was also very satisfied.

Figure 3.5: Dependencies for Russian example. BERT fails to interpret the nouns linked by *appos* in the translation *Catherine of Russia* (appos) *was also very satisfied.*

The dependency relations in low-quality translation can also be partially reflected by the probes in the syntactic probing experiment as shown in Table 3.4, although this correlation is not absolute. *appos* is an appositional modifier that modifies or defines a noun. *flat:name* is a relation used to clarify proper nouns such as names. They all share the characteristic of establishing relationships between nouns in the sentence. In both the Zh and Ru syntactic probing experiments, the F1-score of the top layer (layer-12) of these dependency relation is higher than those of the middle layer (layer-6). In contrast, the results are opposite when conducting similar experiments in De. The reason for the differences may be due to the BERT version used. German BERT is developed from vanilla BERT, while Chinese and Russian BERT are not. The Zh and Ru MT engines utilize UNPC as their training set, while the German MT engine uses Europarl. This may suggest that the syntactic knowledge in the

BERT architecture can be modified by the syntactic difference within the training set, where the training set UNPC may contain more syntactic features, such as noun information, which makes the Chinese and Russian BERT pay more attention to mentioned knowledge. However, more frequent dependency relations, such as *obl* and *case*, in the low-quality translations of all three languages tend to show generally higher F1-score in the middle layer after MT fine-tuning (more details are in Appendix Sec A.2), revealing that those syntactic dependencies affecting translation quality tend to have higher probe performance in the middle layer than in the top layer in BERT. Furthermore, when there is a smaller difference in the number of dependency relations for high and low-quality translations, there is also a smaller drop in F1-score performance between the middle and top layers. For example, an observation can be made regarding the dependency relations called *flat:foreign* (foreign words) and *flat:name* in Ru. While *flat:foreign* has a significant gap between the middle and top layers of BERT with an χ^2 of 2916, that of *flat:name* is smoother under an χ^2 of 168.7.

3.5 Conclusions

This chapter discusses what BERT knows about the syntactic dependencies in source language sentences before and after MT fine-tuning, as well as how the dependency relations to the source language sentences impact translation quality when BERT is used as the encoder of the MT engine in three MT directions. Through syntactic probing experiments, it has been proposed that three specific syntactic phenomena and patterns can help to clarify the syntactic knowledge in BERT before and after MT fine-tuning. It can be confirmed that the F1-score of BERT for detecting most dependency relations is decreased compared to that before fine-tuning, which means that it loses detections of some syntactic knowledge in the MT directions. In addition, syntactic patterns for most dependency relations in BERT do not change even after MT fine-tuning, implying that pre-training is crucial for BERT to form syntactic knowledge and MT fine-tuning is a strategy that is applying syntactic knowledge from BERT to enhance or assist other neural networks in an MT engine.

By examining the translation quality of the three MT scenarios using the QE model and the chi-square test, the experiments show that dependency relations are one factor determining high and low-quality translations. When BERT is used as an encoder in the MT engine, the MT engine experiences low-quality translations when confronted with some source language

sentences. It has been found that certain dependency relations in source language sentences are frequently not identified by the MT engine via the chi-square test principle, which can be the cause of low-quality translations. Additionally, syntactic probing experiments can provide a certain degree of relationships and interpretability to translation quality, although they are not absolute.

In conclusion, the lack of clarity and identification of partial syntactic dependencies by BERT can impact translation quality. It becomes possible to improve translation quality by incorporating explicit syntactic knowledge with limited bilingual data. Later chapters will bring more interpretability from the perspective of syntactic knowledge in other neural networks, discussing the representation of syntactic dependencies and their applications in MT scenarios to improve translation quality.

Chapter 4

Syntactic Interpretability of GAT and its Potential in MT Scenarios

4.1 Introduction

One of the strategies for helping neural networks in the MT engine understand sentence structure better can be explicit syntactic knowledge incorporation, which involves using external syntactic annotations to label the sentence structure. In previous studies, RNN variants were frequently utilized to create explicit syntactic knowledge and showcase the benefits of implementing these strategies in MT engines (S. Wu et al., 2017b; Currey & Heafield, 2018; M. Zhang et al., 2019). Although the performance of the MT engine is improved, the RNN model still has certain inevitable flaws, such as long-range dependencies, exploding and vanishing gradients, which severely consume computational resources and increases the training difficulty of the MT engine, limiting the use of explicit syntactic knowledge incorporation strategies in MT scenarios.

The Transformer model introduced subsequently to the introduction of RNNs does not depend on recurrent or convolutional units. Instead, it uses a self-attention network structure to model and represent input sequences. The self-attention mechanism helps the Transformer model capture more contextual information and accurately distinguish the impact of individual input tokens. As a result, explicit syntactic knowledge strategies shift from RNN-based to Transformer-based, where such knowledge is commonly modeled and represented sequentially and linearly via a self-attention network (Z. Zhang et al., 2020; McDonald & Chiang, 2021). However, syntactic dependency is usually denoted as a syntactic tree-like representation. The

strong dependencies between words are not explicitly modeled in the self-attention network, and the MT engine still observes all the input tokens when calculating the attention score after a sequential positional embedding. Additionally, there is an ongoing debate about the effectiveness of using the attention mechanism as a reliable explanation (Jain & Wallace, 2019; Serrano & Smith, 2019; Brunner et al., 2020). It remains uncertain whether the effectiveness of this strategy is solely due to the attention mechanism. More importantly, a car with an airplane engine never reaches the speed of an airplane. Incorporating explicit syntactic knowledge strategies often rely on the Transformer model, causing a bottleneck in its syntactic performance since all functions depend on such a neural network framework.

GAT is a type of graph neural network that can assign weights to neighbor nodes based on their importance via a multi-head attention mechanism and aggregate information from neighbor nodes. Its graph-based topological representation of syntactic knowledge and preservation of information about strong dependencies between words allows linear sentences to be linked with topological linguistic knowledge through graphs. Therefore, it has gained significant attention in syntactic knowledge modeling and representation in downstream tasks (B. Huang & Carley, 2019; L. Huang et al., 2020; Lyu et al., 2021). However, most studies focus on the applications of GAT in downstream tasks. It remains to be investigated what syntactic knowledge is easy or difficult for GAT to learn, although syntactic knowledge modeling and representation via GAT is practical. Besides, GAT includes multi-head attention and the option for additional layers similar to the Transformer model. It is still being determined whether the performance of the model in representing syntactic knowledge improves by increasing the number of multi-head attention heads and layers. There is insufficient research and discussion regarding the explanations of GAT when considering syntactic knowledge. More research and discussion are needed on how GAT model and represent syntactic knowledge to make decisions and why biases may occur. Such an understanding can enhance the effectiveness of the GAT by providing insight into its decision-making when applying it to MT scenarios.

Another research limitation is as the pre-trained language model is becoming more prevalent, BERT is being increasingly utilized in Transformer-based MT engines. So far, the potential and possibilities of using explicit syntactic knowledge and BERT together in MT scenarios have yet to be discussed, despite some studies exploring their incorporation into other downstream tasks (L. Huang et al., 2020; Li. et al., 2021; N. Ma et al., 2020). GAT only models and

represents the syntactic knowledge the parser provides, while BERT requires learning more complicated knowledge, such as surface information of the sentence, syntactic structure, and semantic knowledge, in MT tasks. Whether GAT yields advantages in syntactic knowledge learning relative to BERT in MT scenarios and whether explicit syntactic knowledge on the graph via GAT working with BERT improves translation quality still need to be investigated.

To address the above research gaps, this research proposes a dependency relation prediction experiment for GAT, where Chinese (Zh), German (De), and Russian (Ru) PUD corpuses with gold syntactic dependencies are used as data sets. The words and syntactic dependencies in the sentence from the PUD corpus are considered as nodes and edges in a graph, where GAT needs to predict the dependency relations between nodes based on the nodes (words) and edges (dependency arcs) information. The number of attention heads and layers of GAT in the dependency relation prediction experiment is increased sequentially to verify how GAT learns syntactic knowledge and which model structure yields the most optimal prediction performance. Moreover, another dependency relation prediction experiment is proposed and applied to BERT to investigate the possibility of fusing GAT and BERT in the MT scenarios. The experiment verifies the difference in prediction performance between GAT and BERT on the PUD corpus via three languages (Zh, Ru, De) for dependency relation prediction through paired t-test and F1-score. The main contributions of this chapter are shown below:

- The study investigates how GAT learns syntactic dependencies of three languages and how attention heads and model layers contribute to dependency relation learning through the dependency relation prediction tasks. GAT prefers to use at least 4 attention heads for syntactic dependency feature extraction to achieve optimal results in predicting dependency relations. It has also been observed that GAT has the most accurate predictions for dependency relations in all languages when the model has 2 layers. However, increasing the number of layers impairs such learning on GAT in different languages, where the prediction scores decrease or even reach a score of 0. Although a small number of dependency relations can withstand this negative effect, most experience a significant decrease in prediction performance. Increasing the number of attention heads does not improve the results.
- The study examines the differences between GAT and BERT in the prediction of dependency relations and the potential of fusing them in terms of syntactic knowledge in the

MT scenarios. Based on the paired t-test, the study finds that the F1-score prediction performance of GAT statistically differs from that of BERT, which is fine-tuned for the MT scenarios. This difference has been observed across three MT scenarios. According to the F1-score of each dependency relation, GAT, which is not being pre-trained and does have a complicated model structure, is able to achieve better prediction of most dependency relations without sacrificing training speed compared to BERT fine-tuned for MT. However, detecting certain dependency relation subtypes can be challenging for GAT, and the number of dependency relations might also limit its detection.

4.2 Investigation of Syntax in GAT

The study proposes dependency relation prediction experiments for GAT (Veličković et al., 2017), aiming to obtain more interpretable information about what it learns about syntactic knowledge through prediction experiments and to explore how the number of attention heads and layers affect its prediction of such knowledge.

Consider an input sentence $S = [w_1, w_2, \dots w_i, w_{i+1}]$, where w denotes a token in the given sentence, and the last token of the sentence is w_{i+1}. The sentence S is then fed into the graph space in GAT, and the tokens in the sentence are transformed into nodes on the graph. The node features provided to a GAT layer are $X = [x_1, x_2, \dots x_i, x_{i+1}], x_i \in \mathbb{R}^F$, where x denotes tokens in the given sentence S, F is the hidden feature size of each node on the graph. Equations 4.1 and 4.2 summarise the working mechanism of GAT.

$$h_i^{out} = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j \in N_i} \alpha_{ij}^k W^k x_j\right) \tag{4.1}$$

$$\alpha_{ij}^k = \frac{exp(LeakyReLU(a^T[Wx_i \parallel Wx_j]))}{\sum_{v \in N_i} exp(LeakyReLU(a^T[Wx_i \parallel Wx_v]))} \tag{4.2}$$

1-hop neighbors $j \in N_i$ for node i, $\Big\Vert_{k=1}^{K}$ denotes the number of K multi-head attention representations are concatenated to aggregate information, W^k is a learnable weight matrix in k-th attention head, σ is a sigmoid function, h_i^{out} is the new hidden feature of the node i. α_{ij}^k is an attention score between node i and j in the k-th attention head, W is the learnable weight matrix, a is a learnable context vector during training, and $LeakyReLU$ is as activation function. For simplicity, feature propagation in GAT can be written as $H_{l+1} = GAT(H_l, A; \Theta_l)$,

where H_{l+1} is the stacked feature states of all input nodes at layer $l+1$, $A \in \mathbb{R}^{n \times n}$ is the graph adjacency matrix in GAT, which is used to describe dependency connections between words. Θ_l is the total model parameters at that layer l (more details are in Chapter 2.4.2).

The dependency relation prediction experiments include tests on three languages, which are Zh, Ru, and De. All sentences with gold syntactic dependency annotations are taken from the PUD corpus[1][2][3] of each language. Therefore, the experiments do not rely on external parser annotation of linguistic knowledge. Each sentence in the PUD corpus is considered a training sample. Consider an English sentence: *A witness told police that the victim had attacked the suspect in April.* Each token in the sentence, as defined by the PUD corpus, is treated as a node in the graph. For example, *A* is a node, and *witness* is another node in the graph. Additionally, the syntactic dependencies in the PUD corpus define the edges between the nodes. For example, the dependency relation between *witness* and *told* is *nsubj* (nominal subject), which means that these two nodes are connected and have an edge named *nsubj*. In other words, the graph completely incorporates the sentence and its syntactic details. In order to predict the dependency relations between nodes, GAT needs to acquire knowledge based on the node features and their connections. For instance, if an edge interconnects *witness* and *told*, after obtaining the representations of those two nodes, GAT must determine the appropriate dependency relation called *nsubj* for this particular edge, as shown in Figure 4.1.

Syntactic dependencies in a PUD corpus can be considered as either a unidirectional or bidirectional graph, as shown in Figure 4.1. In the dependency relation prediction experiments, these syntactic dependencies are considered bidirectional in GAT, where the parent node points to the child node and the child node points to the parent node, respectively. This is because neighboring nodes should have different features when a node is a parent or a child node. To enhance its ability to determine dependency relations between nodes, GAT should take into account and learn the dependency information and the significance of neighboring nodes. During GAT testing, new words not seen in training are initialized with random node embeddings.

Experiments are also conducted to investigate the effects of attention heads and model layers on GAT in dependency relation prediction tasks. The number of attention heads for GAT is sequentially set to 2, 4, 6, and 8. Also, the number of layers is set to 2, 3, 4, 5, and 6. The aim is

[1] `https://github.com/UniversalDependencies/UD_Chinese-PUD`
[2] `https://github.com/UniversalDependencies/UD_Russian-PUD`
[3] `https://github.com/UniversalDependencies/UD_German-PUD`

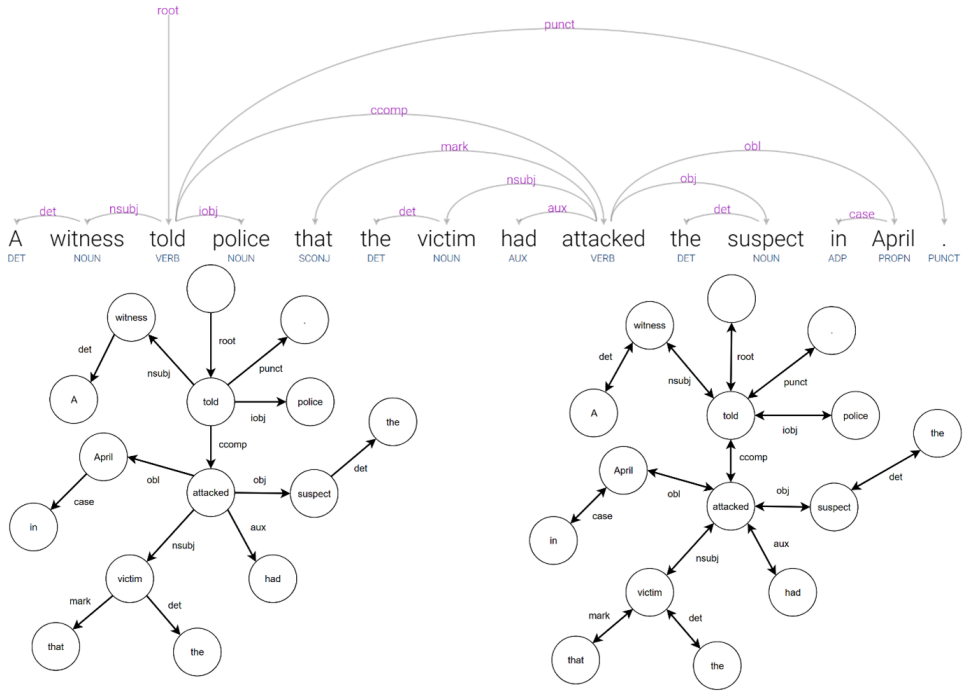


Figure 4.1: The upper figure shows an English sentence and its syntactic information. The lower two figures illustrate how this sentence is transferred to the graph in GAT. Lower left figure represents the unidirectional graph, and lower right figure stands for bidirectional graph.

to investigate the effects of pairing multi-head attentions and model layers in syntactic dependency learning to give more interpretability about syntactic knowledge and model construction. The PUD corpus for all languages is divided into a randomly selected training set, validation set, and test set with the number of sentences 800, 100, and 100, respectively. Dividing data into training, validation, and test sets facilitates the creation of models that are accurate, generalizable to new data, and robust against overfitting and underfitting. Word embedding = 768, node embedding = 768, dropout = 0.2, optimizer = Adam, learning rate = 2e-5, and F1-score are used as evaluation metrics.

4.2.1 Experimental results

According to Table 4.1, the experiments on predicting dependency relations suggest that GAT performs the best overall prediction performance when there are 2 model layers with a minimum of 4 attention heads, e.g., the best overall prediction performance is achieved with 2 layers and 4 attention heads for Ru and De. Zh is more biased toward 6 or 8 attention heads in its prediction results. When it turns to detailed prediction results of each dependency relation (more details are in Appendix Sec B.1), the increase in the number of attention heads can be more beneficial to certain dependency relations. As the number of attention heads increases,

their F1-score gradually rises, e.g., *cop* (copula) for Zh, *acl* (clausal modifier of noun) for Ru, and *conj* (conjunct) for De at a model layer with 2. However, the continued increase in attention heads may not result in more significant prediction gains in F1-score. The number of attention heads above 4 does not lead to considerable performance gains for Ru and De when GAT is 2 layers, either in terms of overall prediction performance or detailed individual prediction results. This phenomenon also occurs in Zh. When comparing the prediction performance, there is no significant difference between using 6 or 8 attention heads. However, it is worth noting that increasing the number of attention heads to 8 may lead to a decrease in prediction scores for some dependency relations. For example, *ccomp* (clausal complement) in Zh has a prediction score of 0.53 with 2 model layers and 6 attention heads. However, this score decreases to 0.30 when the number of attention heads increases to 8.

This observation differs from what has been seen with the Transformer and BERT models (Vaswani et al., 2017; Devlin et al., 2019), as they tend to improve their feature extraction and modeling abilities with the addition of more attention heads. One conjecture is that it is related to the model structure of the neural network. Sequential models such as the Transformer model and BERT discuss the contextual links and contributions of all input tokens to the current token when performing attention calculations. Increasing the number of attention heads allows the model to learn more feature representations and interconnections in several subspaces via different attention heads. However, the observation range of all nodes is already limited in GAT due to syntactic dependency constraints. It is unnecessary to discuss the features of all nodes in more subspaces to select the most valuable neighboring nodes. Therefore, the benefit of the continued increase in the number of attention heads is not significant and may even cause redundancy of feature information and thus impair syntactic knowledge learning.

The experiments also show that as the number of model layers continues to increase exceeding 2, the prediction performance decreases dramatically, and GAT gradually loses its ability to learn and predict dependency relations. Typically, the F1-score of the dependency relation prediction decreases gradually compared to that with 2 layers or even drops to 0, as shown in Table 4.2. The number of dependency relations with an F1-score of 0 is also recorded for each language at each layer with a different number of attention heads, as shown in Figure 4.2. The F1-score of 0 occurs more frequently when there are more than 3 model layers, increasing the number of attention heads does not address this issue. The increase of the number of model layers does

		Zh		
	2 Heads	**4 Heads**	**6 Heads**	**8 Heads**
2 Layers	0.63	0.62	**0.64**	**0.64**
3 Layers	0.64	0.61	0.62	0.63
4 Layers	0.56	0.58	0.64	0.49
5 Layers	0.49	0.50	0.51	0.50
6 Layers	0.37	0.40	0.33	0.33
		Ru		
	2 Heads	**4 Heads**	**6 Heads**	**8 Heads**
2 Layers	0.58	**0.61**	0.47	0.56
3 Layers	0.45	0.55	0.54	0.53
4 Layers	0.44	0.47	0.56	0.57
5 Layers	0.42	0.52	0.46	0.49
6 Layers	0.41	0.36	0.31	0.33
		De		
	2 Heads	**4 Heads**	**6 Heads**	**8 Heads**
2 Layers	0.64	**0.67**	0.64	0.56
3 Layers	0.60	0.56	0.56	0.57
4 Layers	0.56	0.50	0.53	0.53
5 Layers	0.58	0.61	0.50	0.47
6 Layers	0.48	0.49	0.48	0.42

Table 4.1: Overall GAT predictions of dependency relations for three languages with different numbers of attention heads and model layers.

GAT		Zh			Ru			De		
Layers	Heads	advmod	clf	dep	case	flat	mark	acl:relcl	cc	naubj
2	2	0.90	0.87	0.64	0.99	0.85	0.97	0.71	0.97	0.75
	4	0.90	0.82	0.63	0.99	0.86	0.94	0.75	0.99	0.72
	6	0.91	0.89	0.66	0.98	0.87	0.96	0.75	0.96	0.72
	8	0.90	0.83	0.62	0.98	0.86	0.90	0.41	0.97	0.69
3	2	0.90	0.88	0.64	0.98	0	0.93	0.60	0.96	0.78
	4	0.91	0.86	0.64	0.98	0.86	0.94	0.45	0.96	0.71
	6	0.90	0.88	0.66	0.98	0.77	0.93	0.41	0.96	0.72
	8	0.91	0.9	0.66	0.99	0.86	0.93	0.46	0.96	0.74
4	2	0.89	0.68	0.64	0.97	0	0.94	0.52	0.84	0.74
	4	0.90	0.66	0.65	0.99	0.77	0.94	0.45	0.85	0.73
	6	0.91	0.69	0.68	0.99	0.67	0.97	0.40	0.85	0.77
	8	0.90	0	0.64	0.99	0.8	0.94	0.45	0.96	0.74
5	2	0.90	0	0	0.97	0.55	0.93	0.42	0.85	0.78
	4	0.90	0	0	0.98	0.77	0.96	0.68	0.82	0.79
	6	0.90	0	0	0.97	0.67	0.93	0.44	0.81	0.72
	8	0.89	0	0	0.99	0.48	0.96	0.43	0.86	0.73
6	2	0.83	0	0	0.94	0	0.91	0	0.83	0.65
	4	0.86	0	0	0.95	0	0.97	0	0.78	0.65
	6	0.84	0	0	0.94	0	0.93	0	0.79	0.67
	8	0.86	0	0	0.96	0	0.93	0.37	0.85	0.63

Table 4.2: The predictions of some dependency relations in three different languages are shown. As the number of layers increases, GAT gradually loses the learning of syntactic dependencies, and the F1-score even drops to 0. Some dependency relations are unaffected and continue to have relatively high prediction scores.

not bring any performance improvement, which may be due to the loss of inherent properties of the nodes or the generation of more redundant information leading to the degradation of the

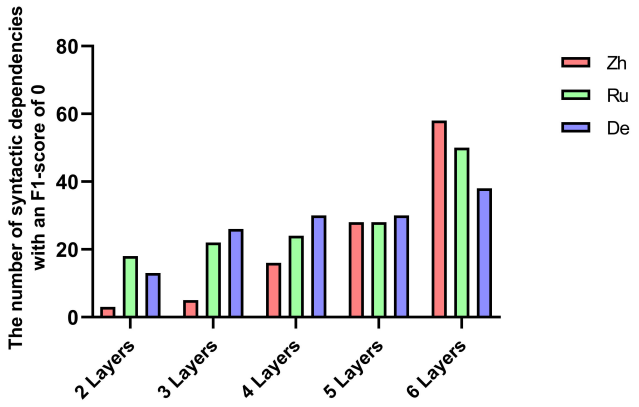


Figure 4.2: The number of F1-score decreases to 0 when the GAT is used in various model layers with varying numbers of attention heads. When the number of layers increases, there are more prediction failures in learning syntactic dependencies, even though each layer has 2, 4, 6, and 8 attention heads, respectively.

model performance. However, GAT still performs well in predicting some dependency relations, e.g., *flat* (flat multiword expression), *compound* (compound), *nmod* (nominal modifier) in Zh. *cop* (copula), *flat:name* (names), *nummod* (numeric modifier) in Ru, *nmod* (nominal modifier), *obl* (object) and *det* (determiner) in De. The F1-score for the predictions of these dependency relations do not drop to 0 as the number of layers increases, and they still maintain relatively valid prediction scores when the depth of the model reaches 6 layers. Although the GAT learns each language differently, some common dependency relations across these three languages relations still have the similar feature that F1-score never reduces to 0: *advmod* (adverbial modifier), *case* (case marking), *cc* (coordinating conjunction), *mark* (marker), *nsubj* (nominal subject), *punct* (punctuation). GAT can easily detect specific dependency relations, regardless of the number of attention heads and model layers being changed, even when different languages are applied. Besides, there is ambiguity in some dependency relations regarding the correlation between the number of attention heads and model layers. An example of this is the F1-score for *advcl* (adverbial clause modifier) in Zh, the F1-score of it is only around 0.3 with 5 model layers, but it becomes almost 0 with 4 or 6 model layers.

Despite this, GAT shows strong prediction capability for the majority of dependency relations in three languages, provided it has the proper number of attention heads and model layers. Specifically, this is the case when there are two layers present. The strong learning of syntactic dependencies explains why incorporating explicit syntactic knowledge through GAT is practical and useful for downstream tasks in other studies. In addition, a deeper GAT can

produce relatively high prediction scores for certain common dependency relationships in different languages, implying that deeper graph neural networks with more model layers, like deep pre-trained language models, are also possible in future research.

4.3 Investigation of GAT and BERT in Syntax

The explicit syntactic knowledge of GAT and the implicit knowledge of BERT can be helpful for downstream tasks. However, the application of MT scenarios is unclear. Therefore, the study described in this section compares the prediction differences between GAT and BERT on the dependency relation prediction experiment to obtain more information about the interpretability and fusion possibilities of both models in terms of syntactic dependencies in the MT scenarios.

Since this study aims to propose a fusion strategy for syntactic knowledge incorporation and BERT for three MT directions (Zh→En, Ru→En, De→En), the experiment for predicting dependency relations involves each of the three source languages (Zh, Ru, and De) along with their corresponding BERT versions[4][5][6]. BERT is fine-tuned for the PUD corpus via the following proposed dependency relation prediction experiment first and then fine-tuned for the MT called MT-B to ensure that BERT learns the syntactic knowledge from the MT task. Although the pre-training strategies for BERT are different for each source language, the model structure is the same (12 layers and 12 attention heads). UNPC[7] is the training set for Chinese and Russian MT engines, whereas Europarl[8] is the training set for German MT engine. In each MT engine (Zh→En, De→En, and Ru→En), the training set size is 1.2M sentence pairs, the validation set is 6K, and the test set is 6K for the MT engines, which use BERT as the encoder and vanilla transformer decoder (the MT engines are built the same as in Chapter 3.2).

After BERT has been fine-tuned for the MT task, the experiment applies that dependency relation prediction experiment again to examine what BERT knows syntactic dependencies after MT fine-tuning via the PUD corpus. A simple fully-connected layer is added to the last layer of BERT as a classification layer, as shown in Figure 4.3. In the experiment, BERT needs to predict the dependency relation corresponding to each input token in the given sentence. All its parameters are frozen to prevent learning new syntactic knowledge from the PUD corpus,

[4] https://huggingface.co/hfl/chinese-bert-wwm-ext
[5] https://huggingface.co/DeepPavlov/rubert-base-cased
[6] https://huggingface.co/bert-base-german-cased
[7] https://opus.nlpl.eu/UNPC.php
[8] https://opus.nlpl.eu/Europarl.php

except for the last fully-connected layer. However, BERT and GAT differ in the way they predict dependency relations. GAT is a graph-based prediction to learn syntactic knowledge, and thus the parent and child nodes are given in the prediction experiments as the case with the construction of Chapter 4.2. Conversely, the dependency relation prediction experiment for BERT provides the current child node, which is each token in the input sentence considered as a dependent in terms of syntactic dependency. However, it does not provide the parent node, which is another token linked to the current token as the head according the dependency connection. Since it is a sequential model that needs to consider the information of all input tokens, this approach models how it considers the syntactic knowledge in MT tasks as much as possible. Besides, BERT already has some syntactic knowledge in pre-training. Setting up a complex prediction experiment (e.g., specifying both parent and child nodes) would neither fit the scenario of BERT application in MT tasks nor determine whether the syntactic knowledge comes from BERT or a complex detection model. Unlike GAT, which always focuses on syntactic knowledge, which is only a part of what BERT needs to learn in the MT task, the dependency relation prediction experiment can reveal how BERT knows syntactic knowledge after fine-tuning the MT task.

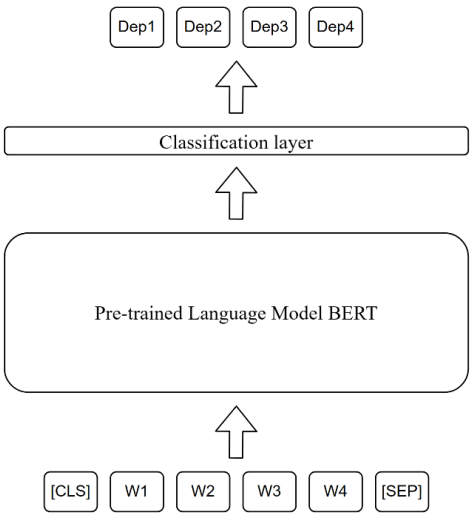


Figure 4.3: Dependency relation prediction experiment for BERT, where BERT needs to predict dependency relations via a classification layer.

Another BERT is added to the prediction task for each language, and its parameters are fine-tuned and updated according to the PUD corpus called UD-B in the dependency relation prediction experiment. It is considered the best model and performance of the BERT for learning syntactic knowledge from the PUD corpus. It would further show the robustness and

competitiveness of GAT in learning syntactic knowledge if it can beat UD-B in the prediction of dependency relations.

Experiments were conducted to evaluate the performance differences between GAT and BERT (MT-B and UD-B) in terms of overall prediction and individual prediction. First, the experiments use paired t-tests to compare whether there is a significant overall prediction difference between GAT and MT-B regarding dependency relations prediction. Second, the experiments discuss the prediction performance of the three models (GAT, MT-B, and UD-B) for individual dependencies by the prediction scores of each dependency relation to discuss their learning differences and the possibility and potential of a fusion of GAT and BERT in the MT scenarios.

The relevant settings for the dependency relation prediction experiments of GAT are the same as in Chapter 4.2. GAT with 2 layers for all language prediction tasks, with 6 attention heads for the Zh and 4 for the Ru and De. To ensure the consistency of GAT and BERT on the dependency relation prediction experiments, the experiments use the PUD corpus as the dataset and apply K-fold cross-validation, where the number of training and test sets are 850 and 150, respectively. GAT and BERT with hidden size = 768, K-fold = 5, learning rate = 2e-5, and optimizer = Adam. F1-score is used as the evaluation metric for the experiments.

4.3.1 Experimental results

As shown in Table 4.3, according to paired t-tests, the p-value of Zh prediction task is less than the significance level (α) revealing that Null Hypothesis (H_0) that GAT and MT-B do not exhibit statistical differences in the F1-score of predictions of dependency relations is rejected. Instead, the Alternative Hypothesis (H_1) is accepted, indicating a statistically significant difference in the F1-score for the dependency relation prediction between GAT and MT-B. Similar conclusions also apply to the predicted results for Ru and De, with p-values less than the significance level (α). Based on the experiments conducted, it has been observed that GAT and BERT have significant differences in predicting dependency relations in three languages (excluding outliers), which difference has been confirmed through paired t-tests.

According to the F1-score shown in Table 4.4, even though MT-B is fine-tuned by the PUD corpus and the MT task, it is not very precise when it comes to predicting dependency relations. Instead, GAT performs better in predicting dependency relations for all languages, and only a few dependency relations had an F1-score that is lower than that of MT-B. Fine-tuning of

Languages	Observations	Sample size	α	Mean	STDev	T-value	P-value
Zh	MT-B	31	0.05	0.57	0.21	3.450	0.001
	GAT			0.74	0.26		
Ru	MT-B	28		0.65	0.22	2.283	0.030
	GAT			0.74	0.24		
De	MT-B	27		0.63	0.20	2.062	0.049
	GAT			0.70	0.25		

Table 4.3: The results of GAT and MT-B on the prediction of dependency relations of the three languages are compared using paired t-tests.

BERT in other downstream tasks may yield similar results, as many studies show that explicit syntactic incorporation strategies via GAT and BERT in downstream tasks can improve model performance. If BERT can demonstrate strong knowledge of syntax after fine-tuning, explicit syntactic incorporation strategies built on GAT would be difficult to have a significant positive performance gain for downstream tasks (L. Huang et al., 2020; M. Chen et al., 2021; Zhou et al., 2022).

Based on the findings in Chapter 3, some dependency relations in the source language that frequently appear in low-quality translations are considered major contributing causes: *appos* (appositional modifier), *case* (case marking), *flat* (flat multiword expression), *flat:name* (names), and *obl* (oblique nominal). The experiments show that GAT is commonly more competitive than MT-B in predicting these dependency relations, which can be a justification for the explicit syntactic knowledge incorporation strategy via GAT in the MT scenarios. In addition, GAT has an advantage over MT-B in predicting other dependency relations, e.g., *nmod* (nominal modifier), *conj* (conjunct) in Zh, *obl* (oblique nominal), *cop* (copula) in Ru, and *advmod* (adverbial modifier), *flat:name* (names) in De. The dependency relation called *root* (root) is the sentence main predicate[1] and indicates the main substance in a sentence. Although GAT and BERT predict dependency relations differently, the relation cannot be associated with decreased translation quality since it is present in every sentence. However, the detection of GAT is still better than that of MT-B, which means BERT fine-tuned for the PUD corpus and MT tasks still lacks the detection to specify such a relation. Not only *root*, but GAT has a more robust prediction performance for most dependency relations, accounting for 25 of the 37 relations in Zh, 20 of the 33 relations in Ru, and 20 of the 32 relations in De. The primary responsibility of GAT in the MT tasks is to model and represent the syntactic dependencies supplied by the external parser. Assuming that GAT can represent such syntactic knowledge as correctly as

[1]One of the orphaned dependents gets promoted to the root position if the main predicate is absent.

	Zh				Ru				De			
	#	MT-B	GAT	UD-B	#	MT-B	GAT	UD-B	#	MT-B	GAT	UD-B
acl	15	0	0	*0*	210	**0.523**	0.392	*0.854*	16	0	0	*0*
acl:relcl	391	0.420	**0.913**	0.836	141	**0.451**	0.405	*0.960*	230	**0.659**	0.605	*0.912*
advcl	454	0.279	**0.376**	*0.728*	170	0.330	**0.334**	*0.842*	183	0.414	**0.495**	*0.832*
advmod	1039	0.668	**0.909**	*0.946*	789	0.843	**0.902**	*0.964*	962	0.622	**0.984**	0.958
amod	356	0.400	**0.919**	0.874	1510	0.872	**0.979**	*0.982*	904	0.658	**0.935**	*0.976*
appos	236	**0.480**	0.423	*0.740*	103	0.428	**0.436**	*0.570*	215	0.350	**0.561**	*0.786*
aux	569	0.758	**0.875**	*0.966*	38	**0.878**	0.836	*0.932*	318	0.818	**0.862**	*0.972*
aux:pass	63	**0.862**	0	*0.970*	96	0.958	**0.988**	0.968	188	0.835	**0.934**	*0.965*
case	1114	0.734	**0.963**	0.928	1804	0.931	**0.983**	0.981	1736	0.840	**0.994**	0.986
case:loc	301	0.670	**0.779**	*0.954*	-	-	-	-	-	-	-	-
cc	234	0.851	**0.990**	0.938	516	0.954	**0.969**	*0.988*	622	0.829	**0.981**	0.972
ccomp	354	0.148	**0.277**	*0.656*	122	0.469	**0.536**	*0.752*	158	**0.289**	0.296	*0.704*
clf	303	**0.816**	0.737	*0.980*	-	-	-	-	-	-	-	-
compound	1493	0.619	**0.881**	*0.886*	6	0	0	0	231	0.465	**0.496**	*0.850*
conj	318	0.481	**0.976**	0.842	593	0.732	**0.862**	*0.920*	716	0.591	**0.673**	*0.912*
cop	170	0.588	**0.962**	0.842	75	0.756	**0.983**	0.830	238	**0.782**	0.755	*0.954*
dep	343	0.251	**0.556**	*0.742*	-	-	-	-	-	-	-	-
det	315	0.712	**0.963**	0.956	399	0.870	**0.997**	0.974	2295	0.914	**0.996**	0.980
expl	-	-	-	-	4	0	0	*0.890*	75	**0.711**	0.319	*0.982*
fixed	-	-	-	-	189	**0.600**	0.577	*0.846*	5	0	0	*0*
flat	67	0.724	**0.867**	*0.965*	55	0.220	**0.583**	*0.538*	2	0.080	**0.371**	*0.344*
flat:foreign	-	-	-	-	78	0.330	**0.903**	0.892	-	-	-	-
flat:name	120	0.791	**0.897**	*0.936*	173	**0.910**	0.888	*0.986*	131	0.486	**0.844**	0.762
iobj	11	0	0	*0.134*	161	**0.510**	0	*0.730*	81	**0.494**	0	*0.874*
mark	242	0.512	**0.980**	0.905	253	0.780	**0.867**	0.854	394	0.817	**0.992**	0.980
mark:adv	19	**0.992**	0.400	*0.970*	-	-	-	-	-	-	-	-
mark:prt	303	**0.438**	0.237	*0.838*	-	-	-	-	-	-	-	-
mark:relcl	540	**0.869**	0.756	*0.944*	-	-	-	-	-	-	-	-
nmod	601	0.386	**0.919**	0.826	1634	0.667	**0.870**	*0.920*	933	0.590	**0.749**	*0.888*
nsubj	1529	0.598	**0.612**	*0.906*	1180	**0.719**	0.666	*0.936*	1285	0.659	**0.678**	*0.950*
nsubj:pass	57	**0.127**	0	*0.766*	147	**0.280**	0	*0.904*	165	**0.391**	0	*0.974*
nummod	666	0.848	**0.993**	0.988	162	0.529	**0.690**	*0.732*	196	0.736	**0.808**	*0.926*
obj	1306	0.459	**0.558**	*0.858*	640	**0.558**	0.518	*0.928*	767	**0.599**	0.485	*0.960*
obl	595	0.204	**0.846**	0.738	1243	0.672	**0.911**	*0.914*	1125	0.584	**0.821**	*0.918*
obl:agent	16	**0.364**	0	*0.888*	9	0	0	*0.520*	-	-	-	-
obl:patient	33	0	0	*0.986*	-	-	-	-	-	-	-	-
obl:tmod	174	**0.534**	0.104	*0.816*	-	-	-	-	101	**0.623**	0.216	*0.832*
parataxis	-	-	-	-	176	**0.525**	0.200	*0.706*	59	0.160	0	*0.524*
punct	2503	0.754	**0.990**	0.990	2589	0.960	**0.990**	0.990	2415	0.932	**0.999**	0.981
root	850	0.493	**0.968**	0.894	850	0.886	**0.994**	0.982	850	0.711	**0.932**	*0.982*
xcomp	467	0.292	**0.437**	*0.804*	306	0.591	**0.634**	*0.880*	158	**0.430**	0.291	*0.820*

Table 4.4: GAT, MT-B, and UD-B present the results of predicting dependency relations in the PUD corpus with F1-score. GAT performs better than MT-B in predicting syntactic dependencies, as indicated by the bold format. Additionally, some syntactic dependencies can exceed those of UD-B, as shown in the non-italic format in the UD-B column.

possible through the graph structure and provide it to the MT engines, the translation quality may be further improved to reduce the generation of translations that do not conform to the syntactic constraints.

The frequency of most dependency relations is less than 500, and the robust prediction of them by GAT indicates that the training cost of GAT is not expensive compared to BERT pre-trained by a large corpus. With the same number of training samples, GAT outperforms MT-B in most dependency relations and can even outperform UD-B in some. However, when the number of

training samples is less than 100, predicting dependency relations can be challenging for both GAT and BERT. The powerful model structure and pre-training based on a large corpus can somewhat alleviate this problem in BERT. Conversely, GAT does not have such functions and thus has worse prediction results. There are 8 dependency relations with a number less than 100 in Zh, and 6 of them are undetectable with an F1-score of 0 in GAT: *acl* (clausal modifier of noun), *aux:pass* (passive auxiliary), *iobj* (indirect object), *nsubj:pass* (passive nominal subject), *obl:agent* (agent modifier), *obl:patient* (object in disposal construction). While Ru and De both have 7 dependency relations with a number less than 100, among which 3 and 4 are undetectable by GAT, respectively. They are *compound* (compound), *expl* (expletive), *obl:agent* (agent modifier) in Ru, and *acl* (clausal modifier of noun), *fixed* (fixed multiword expression), *iobj* (indirect object), *parataxis* (place side by side) in De. Besides, GAT is struggling with the predictions of some subtypes of dependency relations. As an illustration, it can correctly predict *aux* (auxiliary) in Zh with a score of 0.875 but fails to identify its subtype *aux:pass* (passive auxiliary). Although its prediction score for *mark* (marker) can reach 0.980, the prediction scores for its three other subtypes (*mark:adv* (manner adverbializer), *mark:prt* (particle), *mark:relcl* (relative clause)) are not as high as those of MT-B. The same holds true for *nsubj* (nominal subject) and *obl* (object). Some common dependency relations in the three languages that are even difficult to predict for GAT are *iobj* (indirect object) and *nsubj:pass* (nominal subjects of passive clauses). They are consistent in syntactic knowledge classification, with core arguments as functional categories and nominals as structural categories. GAT may not have learned the syntactic subjects of indirect objects and passive clauses well enough. However, robust learning of most of the dependency relations is achieved in the prediction task for all three languages using only a concise model structure and parameters without sacrificing training speed and without requiring large amounts of data support, as shown in Table 4.5. A lightweight and inexpensive GAT shows enough competitiveness in modeling and representing explicit syntactic dependencies compared to BERT.

UD-B achieves a better F1-score in the prediction of dependency relations, while the predictions are as expected, considering that it is pre-trained with large-scale corpus and has more attention heads and complex model structure than GAT. However, it does not have the highest F1-score in all predictions of dependency relations, some relations are lost to GAT, such as *conj* (conjunct) in Zh, *det* (determiner) in Ru, and *advmod* (adverbial modifier) in De. In Zh, GAT outperforms UD-B in a total of 8 dependency relations, 6 of which have sample sizes greater than 300. There

	GAT	MT-B		UD-B	
Batch size	1	16	1	16	1
Speed (sec per epoch)	8	1.5	7.5	3.5	28
Parameters for Zh	5,439,021	102,303,022			
Parameters for Ru	7,345,296	177,884,969			
Parameters for De	6,401,324	109,115,949			

Table 4.5: Model parameters and training speed comparison between GAT, MT-B, and UD-B. The experiments follow the Veličković et al. (2017), where the batch size of GAT is 1. The batch size of BERT is not only 16 but also set to 1 to fairly compare the differences between GAT and BERT.

are 7 of them in Ru, and 3 of which are more than 300. Likewise, there are another 8 in De, 6 of which are over 300. Additionally, the experiments record common dependency relations that predicted better results than UD-B in all three languages: *case* (case marking), *mark* (marker), *det* (determiner), and *cc* (coordinating conjunction). Three of these dependency relations (*case*, *cc*, and *mark*) are also common relations that do not appear to have an F1-score of 0 due to the increase in the number of model layers when it goes through different languages. It is more likely GAT has some cross-linguistic knowledge of syntactic dependencies that are not affected by the model layers and have higher prediction scores than BERT, which can be reflected in three different languages.

4.4 Conclusions

This chapter explores syntactic dependency learning in GAT as well as the impact of the number of attention heads and model layers. Better model optimization is achieved when the number of attention heads in GAT is at least 4. As the number of model layers in GAT increases beyond two, it struggles to identify dependency relations between nodes. This often leads to a significant drop in the F1-score or even a score of 0. Adding more attention heads does not improve the situation.

Additionally, statistically significant differences in the prediction of syntactic knowledge between the GAT and MT-B are shown by paired t-tests and F1-score. Compared to MT-B, GAT retains competitive learning of dependency relations without compromising training speed. It even outperforms UD-B in some dependency predictions. However, some predictions of dependency relations are still challenging for GAT. The accuracy of predictions also can be influenced by the number of dependency relations utilized in the prediction experiments.

Overall, the dependency relations prediction experiments bring more interpretability regarding

syntactic dependencies, revealing how GAT learns them through the number of attention heads and layers, as well as discussing the possibility of explicit syntactic knowledge incorporation via GAT and BERT in MT scenarios. The next chapter will propose the application of syntactic dependencies via GAT and BERT in MT scenarios to improve translation quality based on these findings.

Chapter 5

Syntactic Knowledge via Graph Attention with BERT

5.1 Introduction

BERT is a pre-trained language model that takes inspiration from the Transformer model, which not only retains the advantages of the Transformer model but also has the benefit of pre-training. It uses two pre-training objectives (MLM and NSP) to learn the linguistic features of the given corpus. Such objectives aim to provide word-level and sentence-level features for BERT to perform self-supervised learning to acquire implicit knowledge from the large corpus. The rich implicit knowledge and robust model structure of BERT provide a better framework for initializing and modeling downstream tasks via fine-tuning. Therefore, many studies have attempted to apply BERT as an encoder or decoder component in NMT to aid MT engines in sentence modeling for enhanced model performance (Imamura & Sumita, 2019b; Zhu. et al., 2020).

Syntactic dependency plays a crucial role in the MT tasks, which aims to dissect the syntactic structure of sentences and transform it into an easily understandable tree-like structure and knowledge. MT engines can reduce sentence ambiguity by providing a better understanding of the context of the sentence through such explicit structural information, which helps to improve the accuracy of sentence-specific modeling. Several studies have confirmed the advantages of including explicit syntactic knowledge in the NMT engines (Currey & Heafield, 2019; Z. Zhang et al., 2020; McDonald & Chiang, 2021). The representation of syntactic knowledge

is typically linearly modeled, but there is less discussion about its graph-based topological manner. Although most studies focus on using syntactic knowledge in the vanilla Transformer model, there has yet to be an investigation into the scenarios when BERT is also applied in MT engines. It is still being determined if incorporating both external explicit syntactic knowledge and implicit knowledge from BERT would improve model performance and translation quality. Moreover, there is a lack of linguistic interpretability to investigate the feasibility of such a strategy rather than being limited by BLEU scores.

In linguistics, syntactic dependencies are not sequential and linear in a sentence. Instead, they are commonly depicted through syntactic trees or graphs. Although the Transformer model can process and represent them, the linear representation still does not clearly represent the overall syntactic information and inter-word dependencies. As discussed before, whether the attention mechanism in the Transformer model can be used as an explanation is still under discussion. It is also possible that the modeling and representation of syntactic dependencies in the Transformer model can cause a performance bottleneck for the syntactic strategy. A GNN can be regarded as a method of graph-based topological feature integration where the nodes represent the words in the sentence, and the edges describe the connections between the words, allowing the hierarchical relationship of each node to be clearly represented and more information about the syntactic structure to be retained. In designing a GNN, it is essential to establish the graph structure and node connection for a sentence beforehand. Such structure and connection usually do not change during training, where it is a combination of prior knowledge and explicit features represented on the graph, therefore. The recently proposed GAT has attracted widespread attention in NLP for its ability to more clearly represent syntactic dependencies in sentences through topology, considering neighboring and global features between nodes via the attention mechanism. As a result, some studies have attempted to fuse explicit syntactic knowledge via GAT with BERT in downstream tasks to achieve better performance breakthroughs (L. Huang et al., 2020; M. Chen et al., 2021; Zhou et al., 2022).

What remains to be clarified is whether the fusion of explicit syntactic knowledge incorporation via GAT in a graph-based topology manner and BERT can improve translation quality in MT scenarios. There needs to be more linguistic interpretability to discuss the changes in translation quality, e.g. if translation quality is improved, what source language dependencies can be better recognized by the MT engine to produce better translations? How the explicit

syntactic knowledge on the graphs relates to translation quality and how it affects BERT has yet to be discussed.

In response to the above questions, this chapter introduces the concept of Syntactic knowledge via Graph attention with BERT (SGB) engines. The aim is to explore potential improvements in translation quality by adapting the MT engine, based on the Transformer model, with the incorporation of explicit syntactic knowledge from a graph structure and using BERT as an encoder. Through experiments utilizing the QE model and paired t-tests, it has been confirmed that the proposed engines significantly enhance translation quality. Furthermore, the study discusses comprehensible factors that contribute to improving translation quality and the influence of syntactic knowledge via graphs on BERT regarding syntactic knowledge. Experiments include Chinese (Zh), Russian (Ru), and German (De) to English (En) translation tasks and aim to demonstrate the effectiveness of the proposed approach. The main contributions are shown below:

- SGB engines are the first attempt to demonstrate the significant effectiveness of combining explicit syntactic knowledge via graph attention with BERT in three MT directions, where the training of the MT engine can be fine-tuned to improve translation quality without the need for BERT pre-training from scratch.
- The study in this chapter uses the BLEU score to evaluate basic translation performance, as well as the QE model to score the translations, and explores the before- and after-change in translation quality regarding syntactic knowledge. The proposed SGB engines improve the translation quality of three MT directions without sacrificing the BLEU score. The QE score indicates that SGB engines can improve translation quality under different lengths of source language sentences. The experiments also identify which dependency relations in the source language sentences benefit the most and can be more effectively specified by the SGB engines to produce better translations in three MT directions. Besides, the experiments confirm the effectiveness of the syntactic knowledge on the graphs via GAT by disrupting the order of words in the source language sentences, where such effectiveness relies on the modeling of the sentences by BERT in the first place.
- The experiments show that the robust learning of syntactic dependencies by GAT can be reflected in translation quality. Specifically, if a dependency relation in a source language sentence can be accurately predicted by GAT, it can also be better recognized by the SGB

engines, resulting in a better translation. This study also finds that even though GAT is separate from BERT when working together as an encoder in the SGB engines, it can still influence BERT to consider the syntactic structure of the source language sentences and create a different representation change. This significant representation change usually occurs in the bottom and middle layers of BERT. Furthermore, this study tests another large pre-trained language model called XLM-R-large and syntactic knowledge on graphs via GAT in three MT directions. The experiments show that XLM-R-large and on-graph syntactic knowledge via GAT are still beneficial for translation quality, as confirmed by BLEU and QE scores. It reveals the potential of the proposed approach, which is not limited to BERT.

5.2 Construction of the SGB Engines

In this chapter, the details of the SGB engines are described. Figure 5.1 demonstrates the overall model architecture of the proposed SGB engines, which comprises four layers: encoding layer, graph attention layer, fusion, and output layer.

Encoding: Translations from three source languages into English are included in the experiments: Chinese to English (Zh→En), Russian to English (Ru→En), and German to English (De→En). Given a source language sentence $S = [w_1, w_2, w_3, \dots w_i]$, where i denotes the number of tokens in the sentence, and S is then divided into subword tokens and supplied into BERT, which are transformed into $\tilde{S} = [[CLS], w_1^1, w_1^{1\#1}, w_2, w_3^3, w_3^{3\#3}, \dots w_n, [SEP]]$, Where $w^{n\#n}$ denotes subwords of w_n, [CLS] and [SEP] are special tokens in BERT.

As an encoder for each MT engine, three different BERT variants are used, where chinese-bert-wwm-ext[1] for Chinese MT scenario (Zh→En), rubert-base[2] for Russian MT scenario (Ru→En), and bert-base-german-cased[3] for German MT scenario (De→En). Although BERT variants have the same model structure, their strategies used in pre-training differ. Chinese BERT employs whole word masking, Russian BERT begins with the multilingual version of BERT-base as its initialization, while German has the same strategy as vanilla BERT. The experiments focus on suggesting fusion strategies that can be applied to the structure of BERT models in general with different pre-training strategies rather than just for a specific pre-training approach-based

[1] `https://huggingface.co/hfl/chinese-bert-wwm-ext`
[2] `https://huggingface.co/DeepPavlov/rubert-base-cased`
[3] `https://huggingface.co/bert-base-german-cased`

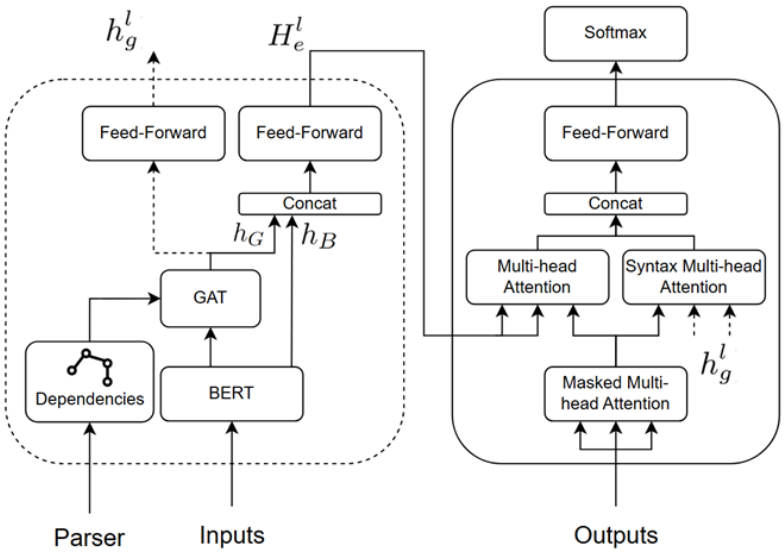


Figure 5.1: The architecture of the SGB engines. The encoder based on BERT and GAT is on the right, and the decoder from the Transformer model is on the left. The dashed lines indicate the optional strategy that the representations from GAT can also guide the decoder.

BERT model.

BERT captures the representation of each subword token, and the final layer of it outputs the final sentence representation, $h_B = BERT(\tilde{S})$. To obtain the syntactic dependency information of the source language sentence, tokenizing and syntactic dependency parsing on source language sentences $\tilde{S}$ are also carried out using a Universal Dependencies-based parser[4]. Following the parsing results, the node adjacency matrix for each input source language sentence is constructed for node connections, and each token corresponds to a node in the graph. Since word embedding representations from BERT contain rich linguistic information, the initial node embedding on the graph is encoded by BERT. Taking into account the subword segmentation of BERT, the subword token representations are merged in an average manner.

Graph Attention: Given the characteristics of syntactic dependencies, there is always a dependency describing the connection between two words. Therefore, words in a sentence are considered graph nodes, while dependencies are edges between nodes. To model and represent the node properties and graph-structured information, GAT (Veličković et al., 2017) serves as the primary component. Tokens in the source language sentence are transformed into nodes in the GAT layer, which are $\tilde{G} = [x_1, x_2, \ldots x_i, \ldots x_n], x_i \in \mathbb{R}^F$, where n denotes the total number of nodes, F is the feature size of each node. The message passing and attention mechanism of GAT is summarised in Equation 5.1 and 5.2.

[4] `https://github.com/hankcs/HanLP`

$$h_i^{out} = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j \in N_i} \alpha_{ij}^k W^k x_j\right) \tag{5.1}$$

$$\alpha_{ij}^k = \frac{exp(LeakyReLU(a^T[Wx_i \parallel Wx_j]))}{\sum_{v \in N_i} exp(LeakyReLU(a^T[Wx_i \parallel Wx_v]))} \tag{5.2}$$

where the node i attends its 1-hop neighbors $j \in N_i$, the concatenation of K multi-head attention output is represented by $\Big\Vert_{k=1}^{K}$. The representation of node i at the given layer is h_i^{out}. α_{ij}^k denotes attention weight between node i and j. W^k is a learnable weight matrix, a stands for the learnable weight vector, an activation function called $LeakyReLU$ is applied at the end. The feature calculation of one-layer GAT can be summarised as $h_G = GAT(X, A; \Theta_l)$. The graph inputs are $X \in \mathbb{R}^{n \times F}$, and the outputs in a certain layer are $h_G \in \mathbb{R}^{n \times F'}$ where n denotes the total number of nodes, F denotes the feature size, F' represents the hidden size in GAT, the graph adjacency matrix $A \in \mathbb{R}^{n \times n}$ indicates node connectivity, Θ_l is the model parameters.

Fusion and Output: The SGB engines contain two strategies for applying explicit syntactic knowledge. The first strategy, Syntactic knowledge via Graph attention with BERT Concatenation (SGBC), combines GAT-based syntactic knowledge and BERT working on the encoder, as shown in Equations 5.3 and 5.4.

$$H_e^l = concat(h_B, h_G) \tag{5.3}$$

$$\tilde{h}_d^l = attn_D(h_d^l, H_e^l, H_e^l) \tag{5.4}$$

where $attn_D$ denotes encoder-decoder attention in the decoder, the output of the l-th layer is denoted by l, d is the representation of the current sentence in the decoder. H_e^l is obtained by concatenating representations from BERT (h_B) and GAT (h_G) in the encoder part. The feed-forward neural network subsequently processes the representations with these attention features alone with residual connection, similar to the case of vanilla transformer.

In the second strategy, Syntactic knowledge via Graph attention with BERT and Decoder (SGBD), the syntactic knowledge on the graph is applied to the encoder and also guides the decoder through syntactic-decoder attention, as shown in Equations 5.5, 5.6 and 5.7.

$$\tilde{h}_d^l = attn_D(h_d^l, H_e^l, H_e^l) \tag{5.5}$$

$$\tilde{h}_s^l = attn_S(h_d^l, h_g^l, h_g^l) \tag{5.6}$$

$$\tilde{h}_t^l = concat(\tilde{h}_d^l, \tilde{h}_s^l) \tag{5.7}$$

where $attn_D$ denotes encoder-decoder attention in the decoder, and $attn_S$ stands for syntax-decoder attention, where the decoder can review the syntactic knowledge via GAT of the source language sentences. h_g^l is the output of the GAT processed by another feed-forward neural network. Representations from the encoder-decoder attention ($attn_D$) and the syntax-decoder attention ($attn_S$) are concatenated to obtain a final representation $\tilde{h}_t^l$ in the decoder. Predicted words are then formed using a linear transformation and a mapping of softmax alone with a feed-forward neural network and residual connection, similar to the vanilla Transformer model as it performs in Transformer model.

5.3 Investigation of Model Performance

The experiments start by training two SGB engines (SGBC and SGBD) with the UNPC and Europarl datasets and evaluating the effectiveness of the proposed strategies through the BLEU score, where the UNPC corpus[5] is for Chinese-English (Zh→En) and Russian-English (Ru→En), Europarl corpus[6] is for German-English (De→En) translation direction. To ensure that the MT engine is adequately trained while avoiding overfitting of the model, 1M sentence pairs for each MT scenario are chosen as the training set in the experiments, and 6K and 5K sentence pairs are used as the validation and test sets. The number of training set sentences is reduced to allow it to simulate the impact of SGB engines on other low-resource languages and performance on a more limited training set size.

The MT engine is regarded as the base model (Baseline) when the encoder is only a BERT. The proposed SGB engines are consistent with the Baseline engines in model training to allow for a fair comparison. The BERT variant for all source languages is the base version, where layers = 12, hidden size = 768, and attention head = 12. The decoder comes from the vanilla Transformer model, with 6 layers and 8 attention heads, and the rest of the modules and parameters remain the same. The parameters of the GAT are 2 layers with 6 attention heads when dealing with syntactic dependencies in Chinese and 2 layers with 4 attention heads for Russian and German. The Adam optimizer trains all MT engines with parameters $\beta_1 = 0.9$

[5] `https://opus.nlpl.eu/UNPC.php`
[6] `https://opus.nlpl.eu/Europarl.php`

and $\beta_2 = 0.98$, the learning rate (excluding GAT) = 2e-5, the learning rate of GAT = 5e-5, dropout of GAT = 0.1, the word embedding = 768, and the cross entropy as loss function. All experiments are conducted on RTX 3080 and 3090 GPUs.

Experimental results

As shown in Table 5.1,the proposed SGB engines perform better in all three translation directions, achieving BLEU scores comparable to or even higher than the Baseline engines in every case. With smaller training samples, the SGB engines are competitive with the Baseline engines in terms of BLEU scores, implying that explicit syntactic knowledge via GAT and implicit knowledge in BERT with a limited bilingual training set is still beneficial for the MT engine to learn the syntactic structure of the source language sentences, and other low-resource language scenarios may also be applicable. Inspired by Kocmi et al. (2021) that the BLEU score is not an accurate indication of the MT engine regarding translation performance, a QE model from their study called COMET is added to the experiment to re-evaluate the translation results of all MT engines. COMET evaluates the correlation among sentences in the source language, MT translations, and references. It assigns QE scores ranging from 0 to 100 to each translation, with higher scores indicating higher translation quality. As illustrated in Table 5.2, the SGB engines also receive a higher QE score, and the MT engines are not as insignificant as the differences presented by BLEU scores. Based on the results, it can be confirmed that the SGB engines outperform the Baseline engines in terms of both BLEU and QE scores. The QE model is a better indicator of translation quality differences since it considers the information, such as the semantic diversity of sentences and syntactic structure, compared to a single reference-based BLEU score. Additionally, the actual translation scenarios always lack provided references, and the source language sentences may also be in or out-of-domain of the training set. The QE model can be a more reliable evaluation metric of the translation quality and generalization ability of the MT engine compared to the BLEU score.

5.4 Investigation of Translation Quality

The performance of the SGB engines in the MT scenarios can be confirmed via the BLEU score. The BLEU score does not take into account the linguistic structure or morphology between the source language sentence and translated sentences (Novikova et al., 2017; Kocmi

	Size	Baseline	SGBC	SGBD
Zh→En	0.1M	24.26	**24.89**	24.72
	0.5M	38.48	**38.71**	38.53
	1M	47.15	**47.23**	47.17
Ru→En	0.1M	21.12	**21.45**	21.33
	0.5M	37.69	**37.74**	37.68
	1M	47.22	**47.36**	47.27
De→En	0.1M	15.41	**15.79**	15.50
	0.5M	26.89	**27.13**	26.92
	1M	37.59	**37.67**	37.63

Table 5.1: BLEU scores for three MT scenarios with different training set sizes. Despite the smaller data set size, the SGB engines are still more competitive than the Baseline engines in terms of BLEU score.

Data set size	Zh→En	Baseline	SGBC	SGBD
1M	BLEU	47.15	**47.23**	47.17
	COMET	82.20	83.69	**84.78**
	Ru→En	Baseline	SGBC	SGBD
	BLEU	47.22	**47.36**	47.27
	COMET	80.93	81.34	**82.56**
	De→En	Baseline	SGBC	SGBD
	BLEU	37.59	**37.67**	37.63
	COMET	78.02	78.66	**79.37**

Table 5.2: Performance of BLEU and QE scores for Baseline and SGB engines on three MT scenarios at 1M training set size.

et al., 2021), however. In order to address this, the following experiments use the gold syntactic knowledge annotated corpus and a QE model. The annotated corpus provides correct syntactic knowledge of the source language sentences, while the QE model focuses on factors such as semantics, syntactic coherence, and word order rationality in the translations. This approach helps investigate changes in translation quality and provides interpretability in terms of syntactic knowledge.

5.4.1 Overall Translation Quality

The Baseline and SGB engines are used to translate sentences from the PUD corpus with gold syntactic annotations for each source language sentence, which are Chinese PUD[7], Russian PUD[8], and German PUD[9]. Each PUD corpus contains 1,000 annotated syntactic source language sentences (different source languages but identical semantics of sentences). The experiments then apply a state-of-the-art QE model[10] considering both the source language sentences and machine outputs to score translations. The QE scores range from 0 to 1, with higher scores

[7] `https://github.com/UniversalDependencies/UD_Chinese-PUD`
[8] `https://github.com/UniversalDependencies/UD_Russian-PUD`
[9] `https://github.com/UniversalDependencies/UD_German-PUD`
[10] `https://github.com/TharinduDR/TransQuest`

representing better-quality translations. The experiments also used paired t-tests and box plots to indicate the differences in translation quality and the distribution of QE scores before and after the proposed strategies, where the paired t-test has a significance level of 0.05.

Experimental results

From Table 5.3, for the comparison between the Baseline and the SGBC engine in Zh (Zh→En), they exhibit average differences, denoted as $\bar{x}_d$, of 0.024 and a standard deviation of differences S_d of 0.109, with a test statistic of 7.18, corresponding to a p-value < 0.001. Similarly, the p-value in comparison between the Baseline and SGBD engines in Zh (Zh→En) is less than 0.001. It means that at a significance level of 0.05, both rejected Null Hypothesis (H_0), where H_0 refers to the QE scores of translation difference between the averages of after (SGBC or SGBD) and before (Baseline) is not big enough to be statistically significant. Instead, Alternative Hypothesis (H_1) is accepted that the translation quality of SGBC and SGBD engines have statistically significant differences with the Baseline engine. Similar results are found for Ru (Ru→En) and De (De→En) translations, indicating that the proposed strategies result in noticeable variations in translation quality compared to the Baseline engines, and such variations are statistically significant in three MT scenarios. Figure 5.2 also shows the distribution characteristics of the Baseline and the two SGB engines in terms of QE scores for three MT scenarios via the box plot. Even though the BLEU scores favor SGBC engines as shown in Table 5.1, the box plots indicate that SGBD engines have a higher distribution of QE scores than SGBC engines. Specifically, the upper quartile, lower quartile, and median values in the box plot for SGBD engines are higher than those for the Baseline and SGBC engines in three MT directions. This finding is consistent with Table 5.2, which also suggests that the translation quality of SGBD engines is commonly slightly superior to SGBC engines.

Language	Sample size	Models		$\bar{x}_d$	S_d	Test statistic	P-value
Zh	1000	Baseline	SGBC	0.024	0.109	7.18	$p < 0.001$
			SGBD	0.032	0.111	9.12	$p < 0.001$
Ru	1000	Baseline	SGBC	0.024	0.042	18.38	$p < 0.001$
			SGBD	0.034	0.045	23.67	$p < 0.001$
De	1000	Baseline	SGBC	0.007	0.113	2.16	$p = 0.030$
			SGBD	0.012	0.110	3.61	$p < 0.001$

Table 5.3: Comparison of Baseline and SGB engines using paired t-test for the PUD corpus translations of three source languages.

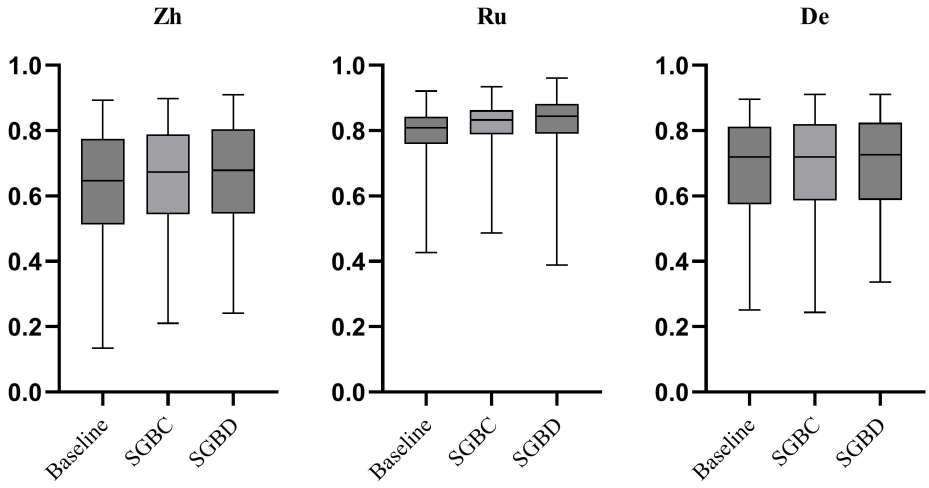


Figure 5.2: The distribution of the QE scores for the translations in the three MT directions is shown in the box plots.

5.4.2 Sentence Length

The experiment also investigates which Sentence Length (Sen L) of the source language sentence benefits most from the proposed strategies for better translation quality. After translating sentences from the PUD corpus in the three languages using the Baseline, SGBC, and SGBD engines and scoring them using the QE model, the experiment ranks the translations from highest to lowest quality according to their QE scores and considers the bottom 30% as low-quality translations (300 translations). In order to make a classification of the source language sentences with different sentence lengths, the experiment again divides the low-quality translations based on the length of their source language sentences. It is important to acknowledge that no explicit rules dictate the distribution requirements for sentence length. Consequently, the subsequent definitions and categorizations of sentence lengths aim to more thoroughly examine how the proposed strategies effectively identify and distribute QE scores based on the length of sentences in the source language. Given an x-length source language sentence corresponding to a low-quality translation, if $x \leq 25$, it is considered a Short-length (S) sentence. If $25 < x \leq 45$, it is considered a Medium-length (M) sentence. If $45 < x$, it is a Long-length (L) sentence. Given the differences in characters and words between the three languages (Chinese does not have space between the characters), Russian and German both follow another classification rule. Given a source langauge sentence of length x, if $x \leq 14$, it is a Short-length (S) sentence. If $14 < x \leq 24$, it is a Medium-length (M) sentence. If $24 < x$, it is a Long-length (L) sentence.

Subsequently, the average QE score of low-quality translations according to sentence length is calculated for each language to investigate the difference in the quality improvement of the proposed SGB engines on low-quality translations.

Experimental results

		Zh		
Sen L	Samples	Baseline	SGBC	SGBD
L	93	0.425	**0.512**	0.508
M	142	0.423	0.500	**0.517**
S	65	0.434	0.543	**0.560**
		Ru		
Sen L	Samples	Baseline	SGBC	SGBD
L	32	0.719	**0.751**	0.745
M	155	0.698	0.746	**0.750**
S	113	0.686	**0.752**	0.747
		De		
Sen L	Samples	Baseline	SGBC	SGBD
L	57	0.513	**0.554**	0.549
M	150	0.512	0.561	**0.586**
S	93	0.482	0.574	**0.578**

Table 5.4: Average QE scores based on the length of the source language sentences for the three MT engines for low-quality translations.

The translation quality of the proposed SGB engines is significantly improved compared with the Baseline engines in all three MT scenarios, as shown in Table 5.4. When it comes to translating source language sentences, SGBD engines tend to excel with medium and short-length sentences, while SGBC engines perform better with longer sentences. In general, translations are most significantly improved when the source language sentences are medium or short in length. However, this phenomenon is not reflected in the previous BLEU scores, where those two SGB engines only have a slight performance advantage. The BLEU score is one of the evaluations that can reflect the performance of the MT engine, while the translation performance provided is based on one standard reference translation in the domain of the training set. The ability of the MT engines to generalize and clarify sentence structure is put into higher demand, as language and syntactic knowledge are diverse, where the PUD corpus also contains out-of-domain sentences (not only news but also wikis). The SGB engines provide more accurate and fluent translations of the PUD corpus containing out-of-domain sentences without sacrificing BLEU scores, reflecting that syntactic knowledge on the graph via GAT can enrich the representations from BERT and guide the decoder in an MT engine learning of syntactic knowledge of the source language sentences.

5.4.3 Syntactic Relations

A sentence contains different dependency relations to indicate the syntactic structure of the sentence. What dependency relations in the source language sentence can be better clarified and learned by the SGB engines to improve the translation quality? In other words, which dependency relations in the source language sentence benefit the most and thus contribute to the quality of the translation? The experiment retains the low-quality translations and groups the source language sentences of the low-quality translations according to their dependency relations. Given that the current dependency relation is d, source language sentences containing d in the low-quality translations are grouped together. The experiment investigates the average QE scores of low-quality translations corresponding to the groups of dependency relation for source language sentences before and after applying the proposed strategies.

Experimental results

Table 5.5 to Table 5.7 lists how dependency relations in source language sentences change and benefit from the proposed SGB engines in each MT direction. Source language sentences containing such dependency relations are recognized better to varying degrees and produce better translations by the SGB engines, where their translations have higher average QE scores than the Baseline engines. Although both the SGBC and SGBD engines apply syntactic knowledge on graphs via GAT, their focus on dependency relations can be different and generate different qualities of translation, as shown in Table 5.8. For example, in the SGBC engine, the dependency relation called *flat* (flat multiword expression) has a significant average QE score compared to that of the Baseline engine. However, such a dependency relation does not have the same effect in the SGBD engine. The difference between the QE score of the Baseline and SGBD engines is less significant than the SGBC engine results. The dependency relation called *flat:name* (names) has an average QE score of 0.761 in Ru in the SGBC engine. When compared to the average QE score of such a relation in the Baseline engine, this relation ranks fourth in terms of score difference between the Baseline and SGBC engines. However, *flat:name* does not appear in the top five average QE score differences between the Baseline and SGBD engines. *cop* (copula) in De also has a similar case.

Nevertheless, some dependency relations in source language sentences are still treated similarly in SGBC and SGBD engines, resulting in a similar average QE score difference between them

Bad Translations (Zh)	Sentences	Baseline	SGBC	SGBD
acl:relcl	112	0.435	0.515	**0.505**
advcl	118	0.430	0.512	**0.518**
advmod	197	0.433	0.512	**0.522**
amod	82	0.435	**0.528**	0.520
appos	109	0.404	0.482	**0.518**
aux	147	0.421	0.514	**0.532**
aux:pass	147	0.436	0.477	**0.525**
case	222	0.428	0.511	**0.526**
case:loc	90	0.429	0.523	**0.531**
cc	49	0.436	**0.513**	0.512
ccomp	92	0.441	0.513	**0.524**
clf	109	0.437	0.527	**0.533**
compound	216	0.427	0.512	**0.524**
conj	55	0.435	**0.521**	0.518
cop	79	0.426	**0.520**	0.511
csubj	19	0.410	0.483	**0.509**
dep	123	0.429	**0.514**	0.513
det	70	0.438	**0.530**	0.528
discourse:sp	30	0.388	**0.502**	0.501
flat	41	0.387	**0.494**	0.473
flat:name	57	0.415	**0.518**	0.506
iobj	6	0.387	0.422	**0.511**
mark	63	0.424	0.510	**0.529**
mark:adv	3	0.365	**0.427**	0.386
mark:prt	80	0.435	**0.532**	0.517
mark:relcl	137	0.431	**0.518**	0.513
nmod	154	0.429	0.509	**0.523**
nsubj	283	0.426	0.510	**0.523**
nsubj:pass	21	0.423	0.512	**0.545**
nummod	162	0.429	0.514	**0.522**
obj	238	0.428	0.514	**0.522**
obl	140	0.432	0.511	**0.534**
obl:agent	8	0.379	0.576	**0.597**
obl:patient	8	0.365	**0.460**	0.434
obl:tmod	60	0.417	**0.509**	0.495
xcomp	114	0.438	0.522	**0.528**
root	300	0.426	0.514	**0.523**

Table 5.5: Translation quality of Chinese sentences under different MT engines according to dependency relations.

Bad Translations (Ru)	Sentences	Baseline	SGBC	SGBD
acl	57	0.708	**0.758**	0.749
acl:relcl	53	0.706	**0.756**	0.747
advcl	58	0.695	0.739	**0.752**
advmod	165	0.704	**0.750**	0.747
amod	228	0.707	**0.753**	0.752
appos	43	0.695	**0.742**	0.740
aux	10	0.700	0.764	**0.777**
aux:pass	36	0.718	0.749	**0.760**
case	265	0.702	**0.748**	**0.748**
cc	135	0.698	**0.751**	0.748
ccomp	32	0.681	0.745	**0.747**
compound	3	0.758	0.802	**0.811**
conj	137	0.699	**0.749**	0.748
cop	30	0.720	0.774	**0.781**
csubj	14	0.699	0.748	**0.757**
det	112	0.697	0.747	**0.746**
fixed	50	0.688	0.742	**0.750**
flat	34	0.696	0.730	**0.738**
flat:foreign	31	0.678	0.701	**0.727**
flat:name	64	0.703	**0.761**	0.751
iobj	61	0.700	**0.746**	**0.746**
mark	65	0.703	0.745	**0.750**
nmod	225	0.705	0.750	**0.751**
nsubj	265	0.701	**0.749**	0.748
nsubj:pass	49	0.708	0.750	**0.754**
nummod	38	0.707	0.748	**0.762**
nummod:gov	25	0.716	**0.769**	0.759
obj	152	0.705	0.755	**0.756**
obl	244	0.701	**0.749**	**0.749**
obl:agent	4	0.716	**0.748**	0.734
orphan	6	0.608	**0.768**	0.719
paratax	54	0.693	**0.725**	0.724
xcomp	78	0.712	**0.760**	0.757
root	300	0.700	0.748	**0.750**

Table 5.6: Translation quality of Russian sentences under different MT engines according to dependency relations.

Bad Translations (De)	Sentences	Baseline	SGBC	SGBD
acl:relcl	83	0.506	0.578	**0.582**
advcl	56	0.514	**0.570**	0.556
advmod	181	0.506	0.573	**0.582**
amod	187	0.507	0.567	**0.571**
appos	92	0.500	0.556	**0.565**
aux	77	0.520	0.586	**0.597**
aux:pass	62	0.498	**0.576**	0.556
case	276	0.504	0.568	**0.574**
cc	140	0.509	**0.565**	0.561
cc:preconj	5	0.539	0.591	**0.597**
ccomp	43	0.514	0.575	**0.579**
compound	65	0.495	**0.577**	0.565
compound:prt	46	0.493	0.579	**0.595**
conj	146	0.510	**0.565**	0.561
cop	77	0.502	0.577	**0.586**
csubj	6	0.449	**0.566**	0.554
csubj:pass	4	0.491	0.464	**0.504**
det	277	0.504	0.565	**0.571**
expl	19	0.486	0.573	**0.589**
flat	5	0.442	0.553	**0.625**
flat:name	71	0.505	0.551	**0.565**
iobj	20	0.546	**0.590**	0.589
mark	87	0.511	0.561	**0.570**
nmod	176	0.517	0.570	**0.574**
nmod:poss	73	0.508	**0.572**	0.556
nsubj	271	0.504	0.571	**0.574**
nsubj:pass	54	0.504	**0.580**	0.575
nummod	47	0.507	**0.581**	0.562
obj	178	0.506	0.576	**0.577**
obl	249	0.502	0.544	**0.574**
obl:tmod	47	0.501	0.531	**0.557**
parataxis	19	0.512	**0.573**	0.546
xcomp	49	0.513	**0.565**	0.553
root	300	0.503	0.570	**0.574**

Table 5.7: Translation quality of German sentences under different MT engines according to dependency relations.

			Zh		
	Baseline	SGBC		Baseline	SGBD
obl:agent	0.379	**0.576**	obl:agent	0.379	**0.597**
discourse:sp	0.388	**0.502**	iobj	0.387	**0.511**
flat	0.387	**0.494**	nsubj:pass	0.423	**0.545**
flat:name	0.415	**0.518**	appos	0.404	**0.518**
mark:prt	0.435	**0.532**	discourse:sp	0.388	**0.501**
			Ru		
	Baseline	SGBC		Baseline	SGBD
orphan	0.608	**0.768**	orphan	0.608	**0.719**
aux	0.700	**0.764**	aux	0.700	**0.777**
ccomp	0.681	**0.745**	ccomp	0.681	**0.747**
flat:name	0.703	**0.761**	discourse	0.614	**0.676**
fixed	0.688	**0.742**	fixed	0.688	**0.750**
			De		
	Baseline	SGBC		Baseline	SGBD
csubj	0.449	**0.566**	flat	0.442	**0.625**
flat	0.442	**0.553**	csubj	0.449	**0.554**
expl	0.486	**0.573**	expl	0.486	**0.589**
compound:prt	0.493	**0.579**	compound:prt	0.493	**0.595**
compound	0.495	**0.577**	cop	0.502	**0.586**

Table 5.8: For each source language, the top 5 dependency relations in source language sentences are listed where there is the most significant difference in average QE scores between the Baseline and SGB engines.

and the Baseline engines. For example, when computing the average QE score difference with the Baseline engine, *obl:agent* (agent modifier) is always in first place in Zh for SGBC and SGBD engines. Similarly, *orphan* (orphan), aux (auxiliary) and *ccomp* (clausal complement) in Ru show the ranking of the average QE score differences between the Baseline engine and the SGB engines, where they are still in the top three of the maximum score differences. Also, when comparing the average QE scores difference between the Baseline engine and the SGB engines separately, the top four positions in De are occupied by *csubj* (clausal subject), *flat*, *expl* (expletive), and *compound:prt* (phrasal verb particle).

The difference between SGBC and SGBD engines is whether there are GAT representations that directly guide the decoder in the feature extraction of source language sentences in an MT engine. The fact that they each selectively process some syntactic structures in the source language sentences to improve the translation quality means that the decoder also learns knowledge directly from GAT, which is responsible for these differences. However, incorporating GAT representations directly into the decoder does not guarantee that it would be helpful for the MT engine. For instance, in the SGBC engines, *discourse:sp* in Zh, *flat:name* in Ru, and *advcl* in De have higher average QE scores than those of the SGBD engines, as shown in Table 5.5 to Table 5.7. This may be because the SGBD engine overemphasizes the syntactic knowledge of

the source language sentences, which leads to knowledge redundancy and a negative impact on the quality of the translation.

As discussed, the SGBC and SGBD engines are similar in their efforts to handle some dependency relations in source language sentences and thus improve translation quality so that those dependency relations would have a similar top ranking in average QE score difference compared to the Baseline engines. The syntactic tree information, as encoded by the GAT, exhibits a level of universality. Consequently, even with the structural variations between the SGBC and SGBD engines, the treatment of syntactic structures in the source language sentences is maintained in a similar fashion.

5.4.4 Disruption of Sentence Order

Informal sentences mostly allow for some intentional bending of rules and conventions. They prevent the MT engines from efficiently recognizing the structure and semantics of the source language sentences since these sentences do not follow the strict syntactic rules that academic writing does. The proposed SGB engines benefit translation quality if the source sentences follow strict syntactic rules. Whether they can assist the MT engines in recognizing the source sentences if they do not follow the syntactic rules or even violate the syntax still needs to be discussed. Therefore, the experiments disrupt the order of the source language sentences in their words in the PUD corpus of each language to verify how topological syntactic knowledge via GAT and BERT determines the translation quality. Given a source language sentence A B C D E F G, the sentence is randomly disordered into C B A G F E D. These new disordered source language sentences are then translated by the Baseline and the SGB engines, where the automatic parser inside the MT engines also performs syntactic structure analysis on these scrambled sequences, which means that the syntactic representations from the parser fed into the model could also be potentially damaged. The QE model then scores the translation quality between unmodified source language sentences and translations. In order to confirm the effectiveness of incorporating syntactic knowledge via GAT into the MT engines, the experiment also randomly selected the same 50 translations from the Baseline, SGBC, and SGBD engines and compared their translation quality, which is done in such a poor-case scenario.

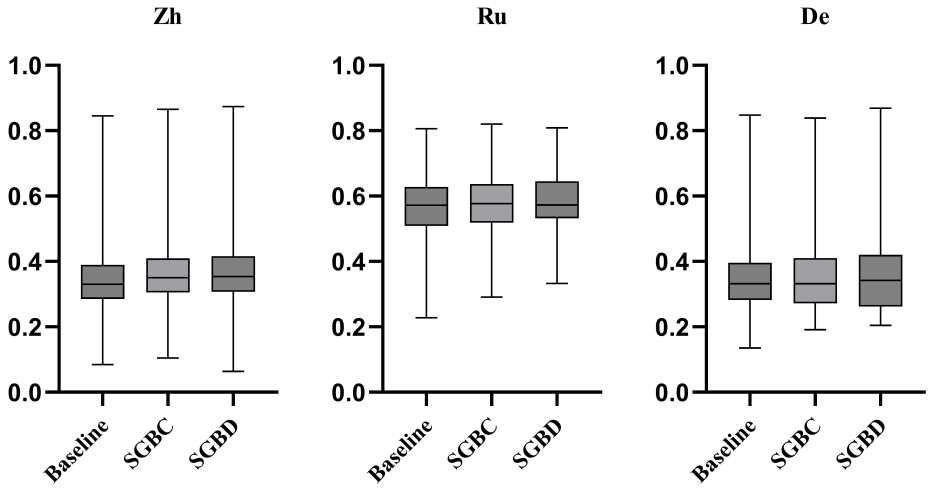


Figure 5.3: When the words in the source language sentences are disordered, the box plots display the distribution of QE scores for the translations in the three MT directions.

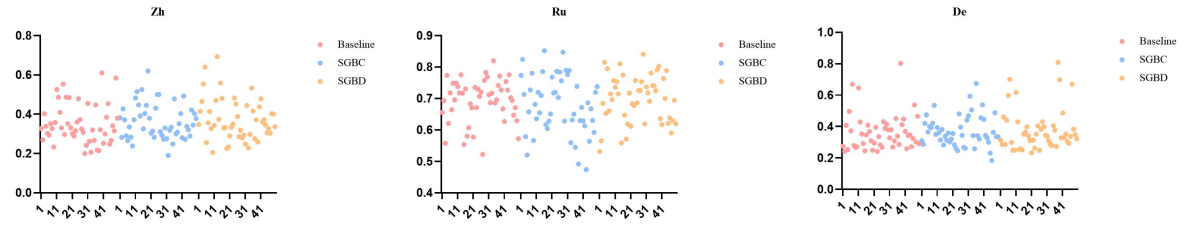


Figure 5.4: Translation quality distribution of the 50 source language sentences translated by different MT engines in three MT directions.

Experimental results

According to Figure 5.3, the Baseline and SGB engines experience a significant decrease in translation quality in the three MT directions when the words in the source language sentence are not in the correct order, compared with what is observed in Figure 5.2. The QE scores distribution of the SGB engines is slightly higher than that of the Baseline engines, which indicates that the improvement in translation quality is a result of the explicit syntactic knowledge via GAT, providing further evidence of the effectiveness of the SGB engines. However, this improvement is limited. The median of each MT engine in the box plots remains below 0.4 for both Zh (Zh→En) and De (De→En) and below 0.6 for Ru (Ru→En), where the translation quality remains unacceptable for all MT directions. There is no guarantee that using GAT in the encoder or providing explicit syntactic knowledge to the decoder can result in a significant improvement in translation quality. For example, it is unlikely that the median of the box plots would increase sharply from below 0.4 to 0.6 as a result.

Figure 5.4 shows the translation quality of 50 randomly selected sentences from the PUD corpus that are translated by different MT engines. It further shows the range of translation quality scores, with the distribution of translation quality for the SGB engines being very similar to that of the Baseline engines in general, where Zh and De translations mainly concentrate around 0.3-0.4, and Ru in the range of 0.6-0.7. While the SGB engines may produce slightly better translations than the Baseline engines in some instances, the increase in translation quality is minimal, such as an increase from 0.35 to 0.38.

It suggests that BERT plays a more significant role than GAT in determining the representation of the source language sentence and translation quality. Although it is pre-trained on a large corpus, it cannot recover and guess the source language sentence meaning as humans do if the sentence order is disrupted in the MT scenarios. Graph-based syntactic knowledge via GAT helps the encoder or the decoder better understand the structure of the source language sentences, but it cannot directly and fundamentally improve translation quality.

5.5 Investigation of Syntactic Knowledge

Although the translation quality benefits from the syntactic knowledge on the graphs, how does GAT relate to the translations and affect BERT? Could the dependency relations in the source language sentences that GAT learns be a factor contributing to improved translation quality? How does BERT deal with the knowledge from GAT, even though BERT knows the syntactic knowledge in pre-training? Therefore, the experiments involving syntactic predictions in GAT and representational similarity analysis in BERT aim to investigate the interpretability of the SGB engines in terms of syntactic knowledge learning.

5.5.1 Syntactic Predictions in GAT

Whether GAT knows syntactic knowledge is one of the clues for translation quality improvement. If GAT can learn about syntactic knowledge, what dependency relations are easy for GAT to go through? Therefore, a syntactic dependency prediction task for GAT is proposed. The task involves transferring source language sentences from each PUD corpus (Chinese PUD, Russian PUD, and German PUD) to a bidirectional graph, as discussed in Chapter 4.2. The current node must consider all its neighbors information, where the nodes on the graph are the words in the sentence, and the edges between the nodes are the dependency connections between the

words. GAT needs to predict the relationships (dependency relations) between nodes based on the node information (words) and the edges (dependency connections).

The GAT not trained for the MT scenarios is the test subject in the experiments to reflect its true mastery of syntactic knowledge. It is because the syntactic knowledge learned by the GAT trained for the MT scenarios comes from the parser. The syntactic annotations generated by the parser may not be correct, however, it is still possible for GAT to remain capable of effectively modeling and representing them. Neither gold dependency detection nor parser dependency detection for GAT trained on the MT scenarios can imply robust or failure learning for syntactic knowledge. E.g., given a dependency relation called *appos* (appositional modifier), it may be defined differently in gold PUD and parser annotations. If the experiment uses gold annotation to detect GAT trained in the MT scenarios in terms of syntactic knowledge, the experiment can not conclude that GAT does not know *appos* (if GAT cannot learn *appos* well), where the truth is that GAT robustly models and represents parser annotations in the MT scenarios. Similarly, if the experiment uses syntactic knowledge from the parser annotation to test the GAT trained in the MT scenarios, the GAT shows strong learning ability for the *appos* from the parser, whereas given that the parser annotation is wrong, the experiment can not conclude that the GAT can master the dependency relation called *appos*.

The number of GAT layers in the experiments is all 2 layers, while the number of attention heads is 6 for the Chinese (Zh) test and 4 for the Russian (Ru) and German (De) tests. Word embedding = 768, optimizer = Adam, learning rate = 5e-5, dropout = 0.1, and F1-score is the evaluation metric. The parameters of the GAT are the same as those in the SGB engines in three MT directions. The training and test sets for each language are divided into 850 and 150 sentences in the experiment.

Experimental results

As shown in Table 5.9, GAT can learn some dependency relations better, and most have F1-scores of 0.9 or higher. Section 5.4.3 shows which dependency relations in the source language sentences are better recognized by SGB engines, improving translation. This experiment reveals a link between prediction performance for dependency relations in GAT and translation quality: When GAT has good prediction performance for some dependency relations, the translation quality of source language sentences containing such relations can be improved in the

Zh				
	GAT Prediction	Baseline	SGBC	SGBD
mark	0.986	0.424	0.510	0.529
cc	0.984	0.436	0.513	0.512
conj	0.970	0.435	0.521	0.518
nummod	0.965	0.429	0.514	0.522
root	0.955	0.426	0.514	0.523
cop	0.945	0.426	0.520	0.511
det	0.935	0.438	0.530	0.528
case	0.934	0.428	0.511	0.526
nmod	0.933	0.429	0.509	0.523
amod	0.927	0.435	0.528	0.520
Ru				
	GAT Prediction	Baseline	SGBC	SGBD
det	0.990	0.697	0.747	0.746
root	0.987	0.700	0.748	0.750
amod	0.982	0.707	0.753	0.752
case	0.978	0.702	0.748	0.760
aux:pass	0.974	0.718	0.749	0.760
cop	0.971	0.720	0.774	0.781
advmod	0.934	0.704	0.750	0.747
cc	0.930	0.698	0.751	0.748
flat:foreign	0.921	0.678	0.701	0.727
obl	0.900	0.701	0.749	0.749
De				
	GAT Prediction	Baseline	SGBC	SGBD
case	0.992	0.504	0.568	0.574
cc	0.987	0.509	0.565	0.561
det	0.987	0.504	0.565	0.571
mark	0.981	0.511	0.561	0.570
advmod	0.932	0.506	0.573	0.582
root	0.931	0.503	0.570	0.574
aux:pass	0.927	0.498	0.576	0.556
amod	0.913	0.507	0.567	0.571
flat:name	0.876	0.505	0.551	0.565
aux	0.868	0.520	0.586	0.597

Table 5.9: The top 10 highest prediction scores of GAT for dependency relations in different source language sentences and the translation quality changes from different MT engines for these sentences.

MT scenarios. For example, when considering the dependency relation called *conj* (conjunct) in Zh, with a strong prediction score of 0.970, the translation quality of source language sentences that contain this relation is enhanced from 0.435 (Baseline) to 0.521 (SGBC) and 0.518 (SGBD). Similarly, *det* (determiner) in Ru obtains the highest prediction score of 0.990, and the translation quality of the source language sentences containing this relation is also improved from 0.697 (Baseline) to 0.747 (SGBC) and 0.746 (SGBD). Such a phenomenon is also observed in De in terms of syntactic prediction and translation quality. It reveals that the prediction of syntactic knowledge in GAT can be one of the factors of translation quality improvement.

However, some dependency relations with low prediction scores also lead to an increase in

translation quality (more details are in Appendix Sec C.1). For example, the prediction scores for *xcomp* in Zh, Ru, and De are 0.423, 0.623, and 0.224, respectively. Despite this, SGB engines can still recognize these relations in the source language sentences more accurately, leading to better translations. One possible reason for the low prediction score of GAT could be due to the insufficient number of dependency relations present in the PUD corpus, which can lead to ineffective learning of such a syntactic feature. Alternatively, it could be because the encoder or decoder needs a more explicit representation of source language sentence structure in the MT engines rather than solely relying on whether the relation annotation itself is correct (the parser can produce different and wrong syntactic dependency annotations).

5.5.2 Representational Similarity Analysis

A method called Representational Similarity Analysis (RSA) is used to compare the similarities between various neural network representation spaces. Inspired by Merchant et al. (2020), n examples are used to compare differences in the representations of two neural networks based on RSA. The representations are converted into a similarity matrix, the Pearson correlation is then calculated by comparing the flat upper triangles of the similarity matrix to determine the similarity score between the representation spaces. The experiment aims to investigate whether syntactic knowledge on the graphs via GAT also affects the representation space of BERT to improve the modeling of source language sentences. Source language sentences corresponding to the 300 low-quality translations are grouped as stimuli according to the type of dependency relations. Given that the current dependency relation is x, any source language sentences corresponding to low-quality translations would be grouped as the stimulus. The experiments extract BERT from the Baseline engine and the SGB engines to compare their representations (e.g., Baseline vs SGBC), and the kernel for all tests is cosine similarity. The RSA score has a range of -1 to 1. A score of -1 means a complete negative correlation, 0 means no correlation, and +1 means a complete positive correlation.

Experimental results

The outcomes of the RSA experiment, which compares the BERT in the Baseline and SGB engines based on the syntactic prediction scores of GAT, are presented in Table 5.10. Based on the experimental results, it can be observed that the comparison of representation changes in each layer of BERT (Baseline vs SGBC and Baseline vs SGBD) reveals that the lowest RSA

			Zh		
	GAT	RSA (Baseline vs SGBC)	Layer	RSA (Baseline vs SGBD)	Layer
mark	0.986	0.418	5	0.407	3
cc	0.984	0.274	4	0.354	5
conj	0.970	0.380	5	0.340	4
nummod	0.965	0.274	4	0.237	3
root	0.955	0.216	4	0.390	4
			Ru		
	GAT	RSA (Baseline vs SGBC)	Layer	RSA (Baseline vs SGBD)	Layer
det	0.990	0.426	4	0.408	3
root	0.987	0.466	3	0.504	3
amod	0.982	0.444	3	0.391	4
case	0.978	0.462	4	0.413	4
aux:pass	0.974	0.357	3	0.327	3
			De		
	GAT	RSA (Baseline vs SGBC)	Layer	RSA (Baseline vs SGBD)	Layer
case	0.992	0.686	5	0.759	2
cc	0.987	0.591	6	0.741	6
det	0.987	0.584	8	0.817	6
mark	0.981	0.676	6	0.769	6
advmod	0.932	0.733	6	0.774	8

Table 5.10: Top-5 highest F1-score of syntactic knowledge learning on the graph and its BERT layer with the lowest similarity in RSA analysis for each language.

scores are typically found in the lower and middle layers of BERT. Specifically, for Zh and Ru, the lowest RSA scores are concentrated in layers 3-5, while for German, they are concentrated in layers 5-8 (more details are in Appendix Sec C.2).

The representation of one of the lower and middle layers of BERT undergoes significant changes, regardless of whether GAT effectively predicts such a dependency relation in source language sentences. For example, the *nummod* (numeric modifier) prediction score by GAT is 0.965 in Zh. When comparing the lowest RSA scores of Baseline and SGBC engines by BERT, it shows that the lowest RSA score is in layer 4 in BERT with only 0.274. Similarly, the lowest RSA score of Baseline and SGBC engines by BERT occurs in layer 3 with a value of 0.237. The predicted score for *ccomp* (clausal complement) by GAT is only 0.337. However, in the above comparisons, the lowest RSA scores are discovered in layer 4 of BERT, with values of 0.403 and 0.249, respectively.

The MT engine uses BERT and GAT as its encoder, which works independently. However, when comparing the RSA for BERT in different MT engines, the lowest RSA scores are consistently observed in the lower and middle layers of BERT. It means that using GAT to understand the structures of source language sentences can enable BERT to re-evaluate the input sentence in a manner that produces more advantageous features for sentence representation by the decoder

via fine-tuning in an MT engine. Typically, the lower and middle layers of BERT concentrate on the surface and syntactic knowledge of the sentence, while the higher layers are more concerned with understanding the high-level semantic information. Even though the lower and middle layers of BERT tend to have the lowest RSA scores, there is still a noticeable variation in the RSA scores of its higher layers. This also suggests that representation changes in the lower or middle layers of BERT have an impact on the representation of advanced linguistic knowledge within the model.

5.6 Investigation of Pre-trained Language Model

This chapter demonstrates that modeling explicit syntactic knowledge via GAT and BERT somewhat improves translation quality. To further verify the feasibility of the proposed strategy on other large pre-trained language models, the experiments replace BERT with XLM-R (Conneau et al., 2020) in the MT engines. XLM-R is trained on 2.5 TB of text data on the Common Crawl large corpus and supports 100 languages. It is still based on the Transformer encoder architecture (12 or 24 layers), which is compatible with the Transformer model. XLM-R differs from BERT in three aspects. Its input is a text stream of any number of sentences (the same language) instead of a text pair of two sentences in BERT. Moreover, XLM-R combines different languages for pre-training, allowing the model to grasp more cross-linguistic information. The NSP strategy in BERT is also removed, and model parameters are tuned to generate a larger shared vocabulary.

BERT is replaced with XLM-R-large[11] (16 attention heads, 24 layers) for the Baseline, SGBC and SGBD engines in all three MT scenarios. The MT engines containing XLM-R are called Baseline-X, SGBC-X, and SGBD-X to differentiate from the previous MT engines (Baseline, SGBC, and SGBD). The Chinese and Russian MT engines (Zh→En and Ru→En) are still trained based on the UNPC[12] corpus, and German MT engines (De→En) are Europarl[13]. The training set size is 0.1M sentence pairs, and the validation and test sets are 6,000 parallel sentence pairs. Experiment sets embedding dimension = 1024, the learning rate (excluding GAT) = 2e-5, the learning rate of GAT = 5e-5, dropout of GAT = 0.1, batch size = 8, optimizer = Adam, cross entropy as the loss function. All MT engines are trained on RTX 3090 GPU.

[11] `https://huggingface.co/xlm-roberta-large`
[12] `https://opus.nlpl.eu/UNPC.php`
[13] `https://opus.nlpl.eu/Europarl.php`

Experimental results

	Size	Baseline-X	SGBC-X	SGBD-X
Zh→En	0.1M	26.28	26.59	27.13
Ru→En		23.62	23.86	24.01
De→En		22.93	23.28	23.46

Table 5.11: BLEU scores in three MT directions for the MT engine replacing BERT with XLM-R-large.

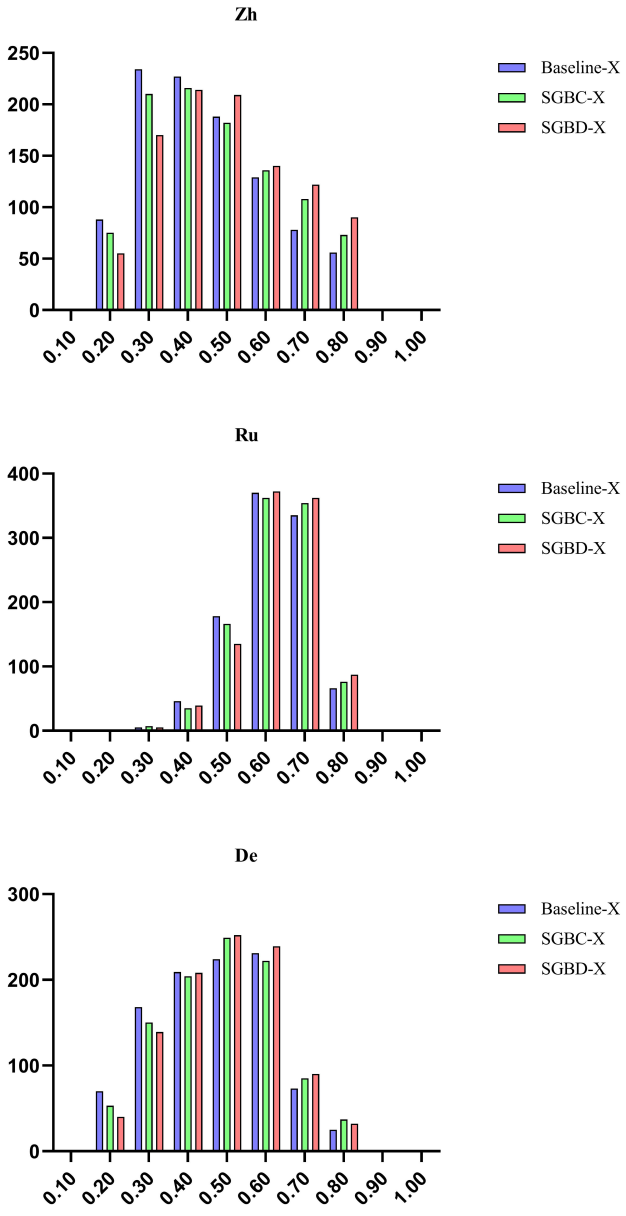


Figure 5.5: QE score intervals and numbers for translations from the Baseline, SGBC-X, and SGBD-X engines in three different MT directions.

As shown in Table 5.11, the SGBC-X and SGBD-X engines continue to have higher overall BLEU scores than the Baseline engines in the three translation directions. In particular, the SGBD-X

engine has a higher BLEU score than the SGBC-X engine in every translation direction. Figure 5.5 illustrates the QE scores of the translations of the source language sentences in the PUD corpus for each engine. Y-axis denotes the number of sentences, and the x-axis denotes the range of scores for the QE scores of the translations. E.g., the x-axis is 0.6 representing the set of translations with QE scores greater than or equal to 0.6 and less than 0.7.

According to Figure 5.5, it can be seen that the Baseline engines have the highest number of sentences in the intervals with x-axis values of 0.2, 0.3, and 0.4 in Zh (Zh→En) and De (De→En). In these intervals, SGBD-X engines provide better translation quality than SGBC-X engines. This is because SGBD-X engines have fewer translations in these intervals, which means that more translations are present in the higher-scoring intervals. A similar trend can be observed in Ru (Ru→En) when the x-axis is 0.4 and 0.5. The Baseline engine still concentrates more number of translations in the interval where the x-axis is 0.4 and 0.5 than SGBC-X and SGBD-X engines. Once the x-axis hits 0.5, the translations of Zh and De experience a change in their number, with the SGBC-X and SGBD-X engines generally starting to outperform the Baseline engines, and this trend continues until the x-axis is 0.8. A similar situation can also be observed in Ru, where the SGBC-X and SGBD-X engines contain more number of translations than the Baseline engine when the x-axis is 0.7 and 0.8. It means that the modeling of syntactic knowledge on the graphs via GAT still benefits the improvement of translation quality in three MT directions for another large pre-trained language model other than BERT. It improves the translations with low QE scores reducing their number as much as possible and includes more translations into the interval with high QE scores.

XLM-R has a deeper model layer and accepts a larger corpus in pre-training relative to BERT. One conjecture is that although it has theoretically more implicit knowledge than BERT, such knowledge is only partially exploited in MT scenarios similar to Chapter 3. BERT knows syntactic knowledge according to syntactic probing experiments, while the MT engines still fail to detect some of them, resulting in low-quality translations. Another conjecture is that the MT engine also has other neural network modules in the encoder or decoder part, these neural networks are more require explicit syntactic knowledge on the graphs via GAT to enrich their sentence representations than XLM-R. Overall this experiment reflects the feasibility of syntactic knowledge on the graphs with XLM-R, implying that the proposed strategies can also work for another large pre-trained language model not limited to BERT.

5.7 Conclusions

This chapter proposes two SGB engines, which fuse syntactic knowledge on graphs via GAT and BERT, to improve translation quality. Experiments show that the SGB engines improve the quality of translations, confirmed by the QE score in three MT directions, without sacrificing the BLEU scores. The experiments also provide clarity on the source language sentence lengths that benefit the most from the SGB engines, resulting in low-quality translation improvements. Additionally, the investigation has shown how SGB engines better recognize dependency relations in source language sentences to improve translation quality. However, improving the translation quality relies on using BERT in the SGB engines. If BERT fails to accurately identify and model source language sentences, syntactic knowledge on the graphs via GAT would not significantly enhance translation quality.

Further experiments are conducted to determine the reason for the change in translation quality. These experiments show that the predictions of dependency relations of the source language sentences by GAT can have a positive relationship with translation quality. And such modeling via GAT can make the middle and bottom layers of BERT reconsider the syntactic structure of source language sentences despite GAT not being integrated into BERT and even be helpful when it comes to another pre-trained language model as the encoder in the MT engines.

In conclusion, this is the first study to present BERT and syntactic knowledge on graphs via GAT in the MT engine, validating the possibilities and potential of GNN and pre-trained language models working together based on the Transformer model in MT scenarios. Future research can be aimed at using other GNNs to model more critical linguistic knowledge and fusing it with larger pre-trained language models in MT scenarios to further improve translation quality.

Chapter 6

Conclusions

NMT is currently the most widely used approach in both research and practical applications of MT. However, current NMT engines have limitations in fully understanding the syntactic knowledge of the source language sentences during the training process, leading to potential syntactic errors or oversights when translating into the target language. Therefore, this thesis centers on the feasibility of fusing the pre-trained language model BERT with the graph-based syntactic knowledge via GAT to guide the MT engines to concentrate on syntactic dependencies of the source language sentences and thus improve low-quality translations in the target language.

This study first investigates what BERT understands about syntactic dependencies of the source language sentences from the PUD corpus before and after MT fine-tuning (Zh→En, Ru→En, De→En) via syntactic probing experiments and clarifies the correlation between dependency relations of the source language sentences and translation quality in the target language through the QE model and chi-square test. The experiments have revealed that there are three syntactic phenomena in BERT when predicting dependency relations in source language sentences, giving insight into its overall learning criteria in terms of syntactic dependencies. During the investigation of each dependency relation predicted by BERT, typical prediction trends called syntactic patterns are also observed between each dependency relation. These syntactic patterns include *Smooth*, *Climb + Decline*, and *Fluctuate*, which can be identified by example dependency relations such as *nummod* (numeric modifier) in Zh for *Smooth*, *obj* (object) in Ru for *Climb + Decline*, and *parataxis* (parataxis) in De for *Fluctuate*. BERT can also construct a coherent learning process for some dependency relations (mainly centered on the functional

category called core arguments in relation to the head in the UD taxonomy) across different source languages, resulting in a similar syntactic pattern even though pre-training strategies and source languages are different. After being fine-tuned for the MT directions, the ability of BERT to comprehend most dependency relations decreased overall. Their prediction curve in each BERT layer would be lower than before fine-tuning. However, its syntactic patterns would not shift in most cases: where one dependency relation has the syntactic patterns called *Smooth* before the MT fine-tuning, it is not likely to change to *Fluctuate* after the MT fine-tuning. This means that MT fine-tuning is not an effective way to strengthen the syntactic dependencies in BERT. Instead, BERT has established a set manner of considering syntactic dependencies during pre-training. In order to prioritize the significance of syntactic dependencies in MT directions, there are two feasible options: training BERT from scratch for syntactic dependencies or utilizing other neural networks to model and represent syntactic knowledge working with BERT in an MT engine.

In addition, QE models and chi-square tests confirm that dependency relations in source language sentences are one of the factors affecting the translation quality of target language sentences. The Baseline engines (BERT is the encoder) fail to detect some sentence structures defined by dependency relations in source language sentences, resulting in low-quality translations in the target language, e.g., *dep* (unspecified dependency) and *advcl* (adverbial clause modifier) in Zh→En, *conj* (conjunct) and *cc* (coordinating conjunction) in Ru→En, *compound* (compound) and *nsubj* (nominal subject) in De→En. In these translation directions, five typical dependency relations across all the source language sentences frequently fail to be detected by the MT engines resulting in low-quality translations in the target language, namely *appos* (appositional modifier), *case* (case marking), *flat* (flat multiword expression), *flat:name* (names), and *obl* (oblique nominal), which can be a common weakness of BERT working with Transformer model in MT directions. Moreover, such dependency relations from the source language sentences affecting low-quality translation can be linked to the predictions given by probes in syntactic probing experiments for BERT, where a higher prediction score tends to be in the middle layer (layer-6) than in the top layer (layer-12) in BERT after MT fine-tuning, such as the F1-score for *obl* (oblique nominal) in BERT fine-tuned for Zh→En direction is 0.39 in layer-6 and 0.31 in layer-12, 0.98 in layer-6 and 0.91 in layer-12 for *case* (case marking) in Ru→En direction, 0.73 in layer-6 and 0.68 in layer-12 for *nsubj* (nominal subject) in De→En direction.

To summarise, the results of syntactic probing experiments have provided a better understanding of what syntactic dependencies of the source language sentences that BERT knows both before and after MT fine-tuning. The QE model and chi-square tests have demonstrated evidence of how and which dependency relations from the source language sentences impact translation quality, which highlights the importance of recognizing syntactic dependencies in BERT-based MT engines. One practical approach to improve translation quality is making the MT engine more accurately identify the syntactic structure of source language sentences.

In order to present syntactic dependencies of the source language sentences using a graph structure rather than sequential modeling in earlier studies, this study then conducts experiments on predicting dependency relations in three languages (Zh, Ru, De) to examine how GAT learns syntactic dependencies via the PUD corpus. This study also gradually increases the number of attention heads and model layers in GAT and pairs them to examine their impact on learning dependency relations. Based on the results of the experiments, adding more attention heads to GAT has been shown to improve its optimal for learning syntactic dependencies. GAT tends to utilize 4 attention heads for more accurate prediction of dependency relations in Ru and De. Similarly, it prefers to use 6 or 8 attention heads in Zh. Their predicted scores would be impaired if the number of model layers is more than 2, where the F1-score decreases or even becomes a score of 0. This phenomenon can be observed in Zh, Ru, and De languages. However, the prediction scores for specific dependency relations do not experience a significant decline as the number of model layers increases. The following dependency relations are common to all three languages tested: *advmod* (adverbial modifier), *case* (case marking), *cc* (coordinating conjunction), *mark* (marker), *nsubj* (nominal subject), and *punct* (punctuation). It appears that certain dependency relations exist that GAT can easily identify, regardless of the number of attention heads and model layers being modified. This is true across different languages.

Additionally, in order to explore the potential of applying explicit syntactic knowledge on graphs via GAT and BERT in MT directions, experiments are conducted to compare their ability to predict syntactic dependencies of the source languages (Zh, Ru, De) through dependency prediction experiments, allowing for a comparison between GAT, MT-B (BERT fine-tuned for MT directions) and UD-B (BERT fine-tuned for PUD corpus). Paired t-tests confirm statistical differences between GAT and MT-B in terms of overall prediction scores in three languages (Zh, Ru, De). The experiments also indicate that GAT achieves better prediction scores for most

dependency relations without sacrificing training speed compared to MT-B. Specifically, GAT outperforms MT-B in predicting 25 out of 37 dependency relations in Zh, 20 out of 33 in Ru, and 20 out of 32 in De. Although GAT prediction is effective, it has limitations when dealing with dependency relations of fewer than 100 examples, such as *nsubj:pass* (passive nominal subject) and another comparison called *nsubj* (nominal subject) in Zh. In particular, there are 57 instances of *nsubj:pass* and 1529 instances of *nsubj*, where the GAT prediction scores for these relations are 0 and 0.612, respectively. Also, when predicting the subtype of a dependency relation, such as *mark* (marker) and its subtypes called *mark:adv* (manner adverbializer), *mark:prt* (particle), and *mark:relcl* (relative clause) in Zh, GAT may experience a lower prediction score and may even encounter difficulties in making accurate predictions. The prediction score for *mark* is 0.980, while the scores for the its subtypes are 0.400, 0.237, and 0.756, respectively. Surprisingly, GAT can even perform better than UD-B in some dependency relations, particularly when the number of dependency relations is over 300. For instance, the dependency relation called *nmod* (nominal modifier), which has 707 examples in Zh, scores 0.919 for GAT while only achieving a score of 0.826 for UD-B. These show that syntactic dependencies on graphs via GAT have the ability to function as a complement to syntactic knowledge working with BERT in MT scenarios.

In summary, these experiments provide substantial evidence to validate the manner in which the GAT acquires syntactic dependencies through its attention heads and layers. Moreover, the syntactic differences between GAT and BERT are investigated in terms of dependency relations to support the possibility of a further application for explicit syntactic knowledge via GAT working with BERT in MT scenarios, which have yet to be explored in early studies.

Based on experiments and investigations for BERT and GAT in terms of syntactic knowledge, this study proposes the SGB engines (SGBC and SGBD) in three translation directions (Zh→En, Ru→En, De→En) where syntactic knowledge on graphs via GAT not only enriches the representation of the BERT-based encoder but also guides the decoder understand how the source language sentences are structured. According to the experiments, the SGB engines outperform the Baseline engines in both BLEU and QE scores, indicating an improvement in translation quality. The experiments also apply the PUD corpus and the QE model to determine that SGBD engines better recognize short and medium-length source language sentences for better translations in the target language, while SGBC engines better recognize long-length source

language sentences. In addition, the experiments clarified what sentence structures defined by dependency relations in the source language sentences are more effectively detected by the SGB engines to achieve better translation quality when compared to the Baseline engines. For example, when the Chinese source language sentences contain *flat* (flat multiword expression), the average QE score of the Baseline engine for their translations is only 0.387, compared to 0.494 and 0.473 for SGBC and SGBD engines, respectively. The typical five dependency relations in the source language sentences affecting translation quality in the three MT directions called *appos* (appositional modifier), *case* (case marking), *flat* (flat multiword expression), *flat:name* (names), and *obl* (oblique nominal) are also better identified by the SGB engines to improve translation quality. As the experiments on disrupting the order of words in the source language sentences show, if BERT fails to identify the source language sentences accurately, then the syntactic knowledge on the graphs via GAT can improve translation quality, but the impact is insignificant. This highlights the importance of having BERT correctly model the source language sentences in SGB engines.

Moreover, the experiment of predicting dependency relations shows that the ability of GAT to predict dependency relations in source language sentences can directly impact the quality of translation. For instance, when predicting the relation for *mark* (marker) in Zh, the prediction score of GAT is 0.986. In source language sentences that contain this dependency relation, the average QE scores for translations with Baseline, SGBC, and SGBD engines are 0.424, 0.510, and 0.529, respectively. The predicted score for *det* (determiner) in Ru is 0.986, and the average QE scores for Baseline, SGBC, and SGBD engines are 0.697, 0.747, and 0.746. RSA experiments also show that despite GAT not being an integral component of BERT, syntactic knowledge on the graphs via GAT can make specific layers of BERT reconsider the syntactic structure of the source language sentences via MT fine-tuning, which is beneficial for translation quality. When the source language sentences are Zh and Ru, the lowest RSA score of BERT in the SGB engines is more biased towards appearing in layers 3 to 5, while De is in layers 5 to 8. There is a discussion of the potential of XLM-R as a replacement for BERT and the use of syntactic knowledge on graphs to improve translation quality. The results indicate that syntactic knowledge on graphs are still beneficial for all three MT directions (Zh→En, Ru→En, De→En). Despite the small size of the datasets, it still shows the applicability of the proposed strategy to another pre-trained language model and highlights the need for syntactic knowledge on graphs via GAT to guide other neural networks (excluding pre-trained language models) in

an MT engine.

In conclusion, the SGB engines (SGBC and SGBD) are the first MT engines with explicit syntactic knowledge on graphs via GAT and the pre-trained language model BERT. They demonstrate that graph-structured syntactic knowledge of the source language sentences and BERT as an encoder can improve translation quality in different MT directions. Besides, the experiments provide more insight into how and why translation quality improves from a syntactic knowledge perspective, which earlier studies have yet to discuss. This thesis motivates more future research into the linguistic application of graph structure knowledge and pre-trained language model in MT scenarios. It also indicates that scoring translations using the QE model can better reveal some linguistic phenomena in MT scenarios, providing more empirical knowledge on how translation quality is improved.

6.1 Major Contributions

- This study investigates what the pre-trained language model BERT learns syntactic knowledge before and after MT fine-tuning (Zh→En, Ru→En, De→En) via syntactic probing experiments, clarifying its syntactic phenomena and patterns on dependency relations, which provides better insight into the syntactic knowledge of BERT in the Transformer model for MT scenarios.

- Distinct from the traditional BLEU metric, this study evaluates the translation quality of the MT engines via the QE model and PUD corpus with gold syntactic annotation to clarify the correlation between syntactic dependencies and translation quality in three translation directions (Zh→En, Ru→En, De→En). This study also identifies the dependency relations of the source language sentences that have the most impact on producing low-quality translations in such translation directions and highlights the correlation between probes in syntactic probing experiments and such relations.

- This study conducts experiments on predicting dependency relations to examine how GAT learns syntactic knowledge in three languages (Zh, Ru, De) under different numbers of attention heads and model layers, clarifying learning bias for dependency relations in GAT. It compensates for the fact that previous studies have only focused on its application of syntactic knowledge and have lacked interpretability of how it models this knowledge.

- This study examines the differences between GAT and BERT in prediction experiments for dependency relations in three languages (Zh, Ru, De). GAT achieves higher prediction scores for most dependency relations without sacrificing training speed showing its potential as a construct of explicit syntactic knowledge working with BERT in the MT directions, which is yet to discuss in early studies.

- This is the first study that proposes SGB engines (SGBC and SGBD), which aim to incorporate explicit syntactic knowledge via graph attention with BERT to help the encoder and the decoder better understander the structure of source language sentences in three MT directions (Zh→En, Ru→En, De→En). The syntactic knowledge on the graphs can be applied to the encoder and decoder in a Transformer-based manner, and it only requires BERT fine-tuning to work in the MT engine without the need to train it from scratch.

- This study demonstrates that the proposed SGB engines (SGBC and SGBD) statistically differ from the Baseline engines regarding translation quality via paired t-test and QE model in three MT directions (Zh→En, Ru→En, De→En). This study also demonstrates that the SGBC engine translates better for longer source language sentences, while the SGBD engine performs better with short to medium-length source language sentences. Additionally, this study identifies how SGB engines recognize those dependency relations in source language sentences and what dependency relations in source language sentences benefit most from the SGB engines to detect and produce better translations in the target language. Furthermore, it is clarified that the most gains in translation quality from syntactic knowledge on the graphs depend on the BERT by disordering words within the source language sentences.

- This study investigates the explanations for why translation quality can be improved regarding syntactic knowledge via the syntactic dependency prediction task and RSA. Based on the experiments, it is discovered that GAT is effective in studying some dependency relations of source language sentences, which can impact the quality of translation in the target language. Also, the lowest RSA scores are usually found in the bottom or middle layers of BERT in SGB engines (SGBC and SGBD), which is in line with previous studies that suggest BERT is more inclined to use middle layers to handle syntax. Besides, it confirms that the manner in which BERT processes the syntactic knowledge of the source language sentence can be impacted by the syntactic dependencies on the graphs, even

though GAT is not integrated into BERT.

- This study also discusses the performance gains of the proposed strategies on another large pre-trained language model called XLM-R in MT directions (Zh→En, Ru→En, De→En). The experiments show that for both BLEU scores and QE scores, the syntactic knowledge on the graphs still helps the MT engines achieve better translation quality, which reflects the extensibility of syntactic knowledge on the graphs working with pre-trained language models in the MT directions.

6.2 Limitations

The PUD corpus for each language is limited to only 1,000 sentences that have been annotated with gold syntax. The GSD corpus contains a higher number of annotated sentences for each language, but it does not have corresponding reference translations. This lack of information makes it difficult to evaluate MT engines and their translations if the GSD corpus is used for the translation tasks. Moreover, the number of annotated sentences in the GSD corpus is inconsistent for each language, which may introduce a bias in the number of dependency relations resulting in an unfair comparison between the experimental languages compared to the experiments using the same number of sentences provided in the PUD corpus. If the PUD corpus provided more annotated sentences, both BERT and GAT would have obtained more accurate results for syntactic dependencies detection and reflected their learning of more dependency relations, avoiding insufficient numbers to prevent the experiment from investigating more details.

Incorporating syntactic knowledge on graphs proposed in this study relies on an external syntactic parser to obtain the syntactic dependencies of the source language sentences. Adding gold syntactic knowledge to text annotations is expensive and not commonly used in MT scenarios. As a result, experiments need to rely on an external parser typically trained on a limited amount of domain-specific annotated data. However, the annotations from the parser may have errors in syntactic annotation, which can limit the efficiency of the syntactic dependencies on the graph to guide BERT and for the decoder to learn syntactic knowledge in the MT engine.

The SGB engines do not discuss the possibility of other large pre-trained language models, though the study has tested the scenario of XLM-R-large in MT engines on a small data

set. Large pre-trained Language models like RoBERTa-large and GPT series, which are pre-trained, have a massive number of model parameters, which means they can potentially achieve better performance. However, since they have several times more parameters than BERT-base, more pre-training corpus, deeper model layers, and more complex pre-training strategies make training and inference geometrically more expensive and computationally intensive. The high number and performance requirements they impose on GPUs limit the application and discussion of such state-of-the-art large-scale pre-trained language models in this study.

6.3 Future Work

Using syntactic information, such as syntactic dependencies, is highly effective for MT engines to model input sentences in MT scenarios since the syntax is derived from a high-level abstraction of human use of language, which can be used as a priori knowledge. Incorporating syntactic structures into the NMT engine is also practical, which typically uses word strings for modeling. There are two methods for incorporating such knowledge into NMT. One involves supplementing the encoder with syntactic information to better represent the source language sentences. The other entails adding syntactic information to the decoder, which enables the translation model to generate more grammatically accurate translations. Incorporating syntactic information in the decoder is more complicated than in the encoder. When generating words and syntactic tree structures interactively, it is crucial to ensure that the tree structure is correct first before obtaining translated word strings. Otherwise, incorrect tree structure could result in problematic translation outcomes. Most studies would therefore favor the application of syntactic information in the encoder. In NMT, it remains to be confirmed how to introduce syntactic knowledge more effectively in the encoder and decoder side, as well as how to properly balance the significance of both syntactic structure information and word strings. Further investigation is also needed to determine if human-annotated syntactic knowledge can enhance the translation model by providing syntactic information that the MT engine cannot learn on its own. If this is feasible, what is the benefit difference between human-annotated syntactic knowledge and knowledge provided by external parsers on MT engines still needs to be confirmed.

GNNs have proven a practical approach and framework for processing graph-structured data in theory and practice. In current research on GNN models, many studies only use the topological

information of whether there are connections between nodes in the graph. There are different numbers of neighboring nodes to the central node, and the distance of the neighboring nodes should have different degrees of influence on it. If more helpful information can be explored in the graph, the performance of the graph neural network can be improved to a higher level. In computer vision and natural language processing, the layers of neural networks can be stacked with multiple layers. Increasing the number of layers of a neural network in the ensemble range can better extract feature information from the data. However, as the number of layers increases in a GNN, the number of hops of the neighboring nodes expands, causing a larger number of central node aggregation features. It results in a dramatic increase in computational complexity during training and can lead to over-fitting. The number of network layers can be increased by limiting the number of nodes per layer, but this also results in a smaller amount of feature aggregation, reducing the information passing between nodes. How to solve this paradoxical problem can be one of the priorities for future research in GNNs.

Deep neural network-based pre-trained language models have proven highly effective for various NLP tasks. However, their purpose is mainly to offer necessary operations and implicit linguistic knowledge via fine-tuning in downstream tasks. It is often necessary to either train a new model from scratch with a new dataset or redesign the inner workings of the pre-trained language model to emphasize specific linguistic knowledge inside. Unfortunately, these methods require significant time and effort, making them impractical for many downstream tasks. According to this study, fine-tuning with syntactic knowledge on graphs can help BERT reconsider the structure of the source language sentences and make new representations that can enhance the quality of translation. Can fine-tuning be used to highlight particular linguistic knowledge in pre-trained language models? If yes, what is the maximum performance and benefit can fine-tuning provide? Additionally, does depending on fine-tuning to supplement specific linguistic knowledge negatively affect the generalization ability of pre-trained language models? A future potential research study is how fine-tuning can be used to guide and adjust the pre-trained language model with finer-grained linguistic knowledge resulting in mechanisms that align more closely with human cognition without sacrificing its ability to generalize.

References

Bahdanau, Dzmitry., Brakel, Philemon., Xu, Kelvin., Goyal, Anirudh., Lowe, Ryan., Pineau, Joelle., Courville, Aaron., and Bengio, Yoshua (2016). "An actor-critic algorithm for sequence prediction". *arXiv preprint arXiv:1607.07086*.

Bahdanau, Dzmitry., Cho, Kyunghyun., and Bengio, Yoshua (2014). "Neural Machine Translation by Jointly Learning to Align and Translate". *CoRR* abs/1409.0473. URL: `https://api.semanticscholar.org/CorpusID:11212020`.

Bastings, Jasmijn., Titov, Ivan., Aziz, Wilker., Marcheggiani, Diego., and Sima'an, Khalil (2017). "Graph Convolutional Encoders for Syntax-aware Neural Machine Translation". *Conference on Empirical Methods in Natural Language Processing*. URL: `https://api.semanticscholar.org/CorpusID:6206777`.

Bhandare, Aishwarya., Sripathi, Vamsi., Karkada, Deepthi., Menon, Vivek., Choi, Sun., Datta, Kushal., and Saletore, Vikram (2019). "Efficient 8-bit quantization of transformer neural machine language translation model". *arXiv preprint arXiv:1906.00532*.

Brown, Peter F., Della Pietra, Stephen A., Della Pietra, Vincent J., Mercer, Robert L., et al. (1993). "The Mathematics of Statistical Machine Translation: Parameter Estimation". *Comput. Linguistics* 19, pp. 263–311. URL: `https://api.semanticscholar.org/CorpusID:13259913`.

Bruna, Joan., Zaremba, Wojciech., Szlam, Arthur., and LeCun, Yann (2014). "Spectral networks and locally connected networks on graphs". English (US). *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*.

Brunner, Gino., Liu, Yang., Pascual, Damian., Richter, Oliver., Ciaramita, Massimiliano., and Wattenhofer, Roger (2020). "On Identifiability in Transformers". *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=BJg1f6EFDB.

Bugliarello, Emanuele and Okazaki, Naoaki (July 2020). "Enhancing Machine Translation with Dependency-Aware Self-Attention", pp. 1618–1627. DOI: 10.18653/v1/2020.acl-main.147. URL: https://aclanthology.org/2020.acl-main.147.

Chakrabarty, Abhisek., Dabre, Raj., Ding, Chenchen., Utiyama, Masao., and Sumita, Eiichiro (Dec. 2020). "Improving Low-Resource NMT through Relevance Based Linguistic Features Incorporation". *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, pp. 4263–4274. DOI: 10.18653/v1/2020.coling-main.376. URL: https://aclanthology.org/2020.coling-main.376.

Chen, Guanhua., Ma, Shuming., Chen, Yun., Dong, Li., Zhang, Dongdong., Pan, Jia., Wang, Wenping., and Wei, Furu (2021). "Zero-Shot Cross-Lingual Transfer of Neural Machine Translation with Multilingual Pretrained Encoders". *Conference on Empirical Methods in Natural Language Processing*. URL: https://api.semanticscholar.org/CorpusID:233295984.

Chen, Huadong., Huang, Shujian., Chiang, David., and Chen, Jiajun (July 2017). "Improved Neural Machine Translation with a Syntax-Aware Encoder and Decoder". *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Regina. Barzilay and Min-Yen Kan. Vancouver, Canada: Association for Computational Linguistics, pp. 1936–1945. DOI: 10.18653/v1/P17-1177. URL: https://aclanthology.org/P17-1177.

Chen, Kehai., Wang, Rui., Utiyama, Masao., Liu, Lemao., Tamura, Akihiro., Sumita, Eiichiro., and Zhao, Tiejun (2017). "Neural machine translation with source dependency representation". *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2846–2852.

Chen, Mingfei., Wu, Wencong., Zhang, Yungang., and Zhou, Ziyun (2021). "Combining Adversarial Training and Relational Graph Attention Network for Aspect-Based Sentiment Analysis

with BERT". *2021 14th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. IEEE, pp. 1–6.

Cheng, Yong., Xu, Wei., He, Zhongjun., He, Wei., Wu, Hua., Sun, Maosong., and Yang, Liu (2019). "Semi-supervised learning for neural machine translation". *Joint training for neural machine translation*, pp. 25–40.

Cho, Kyunghyun., Van Merriënboer, Bart., Gulcehre, Caglar., Bahdanau, Dzmitry., Bougares, Fethi., Schwenk, Holger., and Bengio, Yoshua (2014). "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". *Conference on Empirical Methods in Natural Language Processing*. URL: https://api.semanticscholar.org/CorpusID:5590763.

Clark, Kevin., Luong, Minh-Thang., Le, Quoc V., and Manning, Christopher D. (2020). "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators". *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=r1xMH1BtvB.

Clinchant, Stephane., Jung, Kweon Woo., and Nikoulina, Vassilina (Nov. 2019). "On the use of BERT for Neural Machine Translation". *Proceedings of the 3rd Workshop on Neural Generation and Translation*. Ed. by Alexandra. Birch, Andrew. Finch, Hiroaki. Hayashi, Ioannis. Konstas, Thang. Luong, Graham. Neubig, Yusuke. Oda, and Katsuhito Sudoh. Hong Kong: Association for Computational Linguistics, pp. 108–117. DOI: 10.18653/v1/D19-5611. URL: https://aclanthology.org/D19-5611.

Coenen, Andy., Reif, Emily., Yuan, Ann., Kim, Been., Pearce, Adam., Viégas, Fernanda., and Wattenberg, Martin (2019). "Visualizing and Measuring the Geometry of BERT". *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc.

Conneau, Alexis and Lample, Guillaume (2019). "Cross-lingual language model pretraining". *Advances in neural information processing systems* 32.

Conneau, Alexis., Khandelwal, Kartikay., Goyal, Naman., Chaudhary, Vishrav., Wenzek, Guillaume., Guzmán, Francisco., Grave, Edouard., Ott, Myle., Zettlemoyer, Luke., and Stoyanov,

Veselin (2019). "Unsupervised Cross-lingual Representation Learning at Scale". *Annual Meeting of the Association for Computational Linguistics*. URL: https://api.semanticscholar.org/CorpusID:207880568.

– (July 2020). "Unsupervised Cross-lingual Representation Learning at Scale". *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 8440–8451. DOI: 10.18653/v1/2020.acl-main.747. URL: https://aclanthology.org/2020.acl-main.747.

Cubuk, Ekin D., Zoph, Barret., Mane, Dandelion., Vasudevan, Vijay., and Le, Quoc V (2019). "Autoaugment: Learning augmentation strategies from data". *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 113–123.

Currey, Anna and Heafield, Kenneth (Oct. 2018). "Multi-Source Syntactic Neural Machine Translation". *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Ed. by Ellen. Riloff, David. Chiang, Julia. Hockenmaier, and Jun'ichi Tsujii. Brussels, Belgium: Association for Computational Linguistics, pp. 2961–2966. DOI: 10.18653/v1/D18-1327. URL: https://aclanthology.org/D18-1327.

– (Aug. 2019). "Incorporating Source Syntax into Transformer-Based Neural Machine Translation". *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*. Florence, Italy: Association for Computational Linguistics, pp. 24–33. DOI: 10.18653/v1/W19-5203. URL: https://aclanthology.org/W19-5203.

De Marneffe, Marie-Catherine., MacCartney, Bill., Manning, Christopher D., et al. (2006). "Generating typed dependency parses from phrase structure parses." *Lrec*. Vol. 6, pp. 449–454.

Defferrard, Michaël., Bresson, Xavier., and Vandergheynst, Pierre (2016). "Convolutional neural networks on graphs with fast localized spectral filtering". *Advances in neural information processing systems* 29.

Devlin, Jacob., Chang, Ming-Wei., Lee, Kenton., and Toutanova, Kristina (June 2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". *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)*. Minneapolis,

Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

Ding, Liang., Wang, Longyue., and Liu, Siyou (2022). "Recurrent graph encoder for syntax-aware neural machine translation". *International Journal of Machine Learning and Cybernetics* 14, pp. 1053–1062. URL: https://api.semanticscholar.org/CorpusID:253412237.

Ding, Yanzhuo., Liu, Yang., Luan, Huanbo., and Sun, Maosong (2017). "Visualizing and understanding neural machine translation". *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1150–1159.

Dou, Zi-Yi., Tu, Zhaopeng., Wang, Xing., Shi, Shuming., and Zhang, Tong (Oct. 2018). "Exploiting Deep Representations for Neural Machine Translation". *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Ed. by Ellen. Riloff, David. Chiang, Julia Hockenmaier, and Jun'ichi Tsujii. Brussels, Belgium: Association for Computational Linguistics, pp. 4253–4262. DOI: 10.18653/v1/D18-1457. URL: https://aclanthology.org/D18-1457.

Duan, Sufeng., Zhao, Hai., and Zhang, Dongdong (2020). "Syntax-Aware Data Augmentation for Neural Machine Translation". *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 31, pp. 2988–2999. URL: https://api.semanticscholar.org/CorpusID:216642136.

Duan, Sufeng., Zhao, Hai., Zhou, Junru., and Wang, Rui (2019). "Syntax-aware transformer encoder for neural machine translation". *2019 International Conference on Asian Language Processing (IALP)*. IEEE, pp. 396–401.

Dufter, Philipp., Schmitt, Martin., and Schütze, Hinrich (Sept. 2022). "Position Information in Transformers: An Overview". *Computational Linguistics* 48.3, pp. 733–763. DOI: 10.1162/coli_a_00445. URL: https://aclanthology.org/2022.cl-3.7.

Elman, Jeffrey L (1990). "Finding structure in time". *Cognitive science* 14.2, pp. 179–211.

Eriguchi, Akiko., Hashimoto, Kazuma., and Tsuruoka, Yoshimasa (Aug. 2016). "Tree-to-Sequence Attentional Neural Machine Translation". *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Katrin. Erk and

Noah A. Smith. Berlin, Germany: Association for Computational Linguistics, pp. 823–833. DOI: 10.18653/v1/P16-1078. URL: https://aclanthology.org/P16-1078.

Forbes, Maxwell., Holtzman, Ari., and Choi, Yejin (2019). "Do neural language representations learn physical commonsense?" *arXiv preprint arXiv:1908.02899*.

Gehring, Jonas., Auli, Michael., Grangier, David., and Dauphin, Yann N (2016). "A Convolutional Encoder Model for Neural Machine Translation". *Annual Meeting of the Association for Computational Linguistics*. URL: https://api.semanticscholar.org/CorpusID:6728280.

Goldberg, Yoav (2019). "Assessing BERT's syntactic abilities". *arXiv preprint arXiv:1901.05287*.

Goodfellow, Ian., Pouget-Abadie, Jean., Mirza, Mehdi., Xu, Bing., Warde-Farley, David., Ozair, Sherjil., Courville, Aaron., and Bengio, Yoshua (2014). "Generative Adversarial Nets". *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani., M. Welling., C. Cortes., N. Lawrence., and K.Q. Weinberger. Vol. 27. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Gori, Marco., Monfardini, Gabriele., and Scarselli, Franco (2005). "A new model for learning in graph domains". *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 2. IEEE, pp. 729–734.

Gulcehre, Caglar., Firat, Orhan., Xu, Kelvin., Cho, Kyunghyun., Barrault, Loic., Lin, Huei-Chi., Bougares, Fethi., Schwenk, Holger., and Bengio, Yoshua (2015). "On using monolingual corpora in neural machine translation". *arXiv preprint arXiv:1503.03535*.

Guo, Shengnan., Lin, Youfang., Feng, Ning., Song, Chao., and Wan, Huaiyu (2019). "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting". *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01, pp. 922–929.

Hamilton, Will., Ying, Zhitao., and Leskovec, Jure (2017). "Inductive representation learning on large graphs". *Advances in neural information processing systems* 30.

Hassan, Hany., Aue, Anthony., Chen, Chang., Chowdhary, Vishal., Clark, Jonathan., Federmann, Christian., Huang, Xuedong., Junczys-Dowmunt, Marcin., Lewis, William., Li, Mu.,

et al. (2018). “Achieving human parity on automatic Chinese to English news translation”. *arXiv preprint arXiv:1803.05567*.

He, Kaiming., Zhang, Xiangyu., Ren, Shaoqing., and Sun, Jian (2016). “Deep residual learning for image recognition”. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hewitt, John and Manning, Christopher D (2019). “A structural probe for finding syntax in word representations”. *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)*, pp. 4129–4138.

Hinton, Geoffrey E and Salakhutdinov, Ruslan R (2006). “Reducing the dimensionality of data with neural networks”. *science* 313.5786, pp. 504–507.

Huang, Binxuan and Carley, Kathleen (Nov. 2019). “Syntax-Aware Aspect Level Sentiment Classification with Graph Attention Networks”. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Ed. by Kentaro. Inui, Jing. Jiang, Vincent. Ng, and Xiaojun Wan. Hong Kong, China: Association for Computational Linguistics, pp. 5469–5477. DOI: 10.18653/v1/D19-1549. URL: https://aclanthology.org/D19-1549.

Huang, Lianzhe., Sun, Xin., Li, Sujian., Zhang, Linhao., and Wang, Houfeng (2020). “Syntax-aware graph attention network for aspect-level sentiment classification”. *Proceedings of the 28th international conference on computational linguistics*, pp. 799–810.

Hutchins, John (2007). “Machine translation: A concise history”. *Computer aided translation: Theory and practice* 13.29-70, p. 11.

Imamura, Kenji and Sumita, Eiichiro (Nov. 2019a). “Recycling a Pre-trained BERT Encoder for Neural Machine Translation”. *Proceedings of the 3rd Workshop on Neural Generation and Translation*. Hong Kong: Association for Computational Linguistics, pp. 23–31. DOI: 10.18653/v1/D19-5603. URL: https://aclanthology.org/D19-5603.

– (2019b). “Recycling a pre-trained BERT encoder for neural machine translation”. *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pp. 23–31.

Jain, Sarthak and Wallace, Byron C. (June 2019). "Attention is not Explanation". *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)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 3543–3556. DOI: 10.18653/v1/N19-1357. URL: https://aclanthology.org/N19-1357.

Jawahar, Ganesh., Sagot, Benoît., and Seddah, Djamé (2019a). "What does BERT learn about the structure of language?" *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*.

– (2019b). "What Does BERT Learn about the Structure of Language?" *Annual Meeting of the Association for Computational Linguistics*.

Jean, Sébastien., Cho, Kyunghyun., Memisevic, Roland., and Bengio, Yoshua (2014). "On Using Very Large Target Vocabulary for Neural Machine Translation". *ArXiv* abs/1412.2007. URL: https://api.semanticscholar.org/CorpusID:2863491.

Kaji, Hiroyuki (1988). "An efficient execution method for rule-based machine translation". *Coling Budapest 1988 Volume 2: International Conference on Computational Linguistics*.

Kalchbrenner, Nal and Blunsom, Phil (2013). "Recurrent continuous translation models". *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1700–1709.

Kitaev, Nikita., Kaiser, Łukasz., and Levskaya, Anselm (2020). "Reformer: The Efficient Transformer". *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=rkgNKkHtvB.

Kocmi, Tom., Federmann, Christian., Grundkiewicz, Roman., Junczys-Dowmunt, Marcin., Matsushita, Hitokazu., and Menezes, Arul (Nov. 2021). "To Ship or Not to Ship: An Extensive Evaluation of Automatic Metrics for Machine Translation". *Proceedings of the Sixth Conference on Machine Translation*. Online: Association for Computational Linguistics, pp. 478–494. URL: https://aclanthology.org/2021.wmt-1.57.

Koehn, Philipp and Knowles, Rebecca (Aug. 2017). "Six Challenges for Neural Machine Translation". *Proceedings of the First Workshop on Neural Machine Translation*. Ed. by Thang.

Luong, Alexandra. Birch, Graham. Neubig, and Andrew Finch. Vancouver: Association for Computational Linguistics, pp. 28–39. DOI: 10.18653/v1/W17-3204. URL: https://aclanthology.org/W17-3204.

Krizhevsky, Alex., Sutskever, Ilya., and Hinton, Geoffrey E (2017). "Imagenet classification with deep convolutional neural networks". *Communications of the ACM* 60.6, pp. 84–90.

Kromann, Matthias Trautner (2003). "The Danish Dependency Treebank and the DTAG Treebank Tool". English. Opstilling: 800.92 kro Løbe nr.: 0313404 Sider: 217-220; TLT 2003. The Second Workshop on Treebanksand Linguistic Theories ; Conference date: 14-11-2003 Through 15-11-2003.

Kumar, Sachin., Anastasopoulos, Antonios., Wintner, Shuly., and Tsvetkov, Yulia (Aug. 2021). "Machine Translation into Low-resource Language Varieties". *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Ed. by Chengqing. Zong, Fei. Xia, Wenjie. Li, and Roberto Navigli. Online: Association for Computational Linguistics, pp. 110–121. DOI: 10.18653/v1/2021.acl-short.16. URL: https://aclanthology.org/2021.acl-short.16.

Lan, Zhenzhong., Chen, Mingda., Goodman, Sebastian., Gimpel, Kevin., Sharma, Piyush., and Soricut, Radu (2019). "ALBERT: A lite bert for self-supervised learning of language representations". *arXiv preprint arXiv:1909.11942*.

Le, An Nguyen., Martinez, Ander., Yoshimoto, Akifumi., and Matsumoto, Yuji (2017). "Improving sequence to sequence neural machine translation by utilizing syntactic dependency information". *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 21–29.

LeCun, Yann., Bengio, Yoshua., and Hinton, Geoffrey (2015). "Deep learning". *nature* 521.7553, pp. 436–444.

LeCun, Yann., Boser, Bernhard., Denker, John S., Henderson, Donnie., Howard, Richard E., Hubbard, Wayne., and Jackel, Lawrence D (1989). "Backpropagation applied to handwritten zip code recognition". *Neural computation* 1.4, pp. 541–551.

Levie, Ron., Monti, Federico., Bresson, Xavier., and Bronstein, Michael M (2018). "Cayleynets: Graph convolutional neural networks with complex rational spectral filters". *IEEE Transactions on Signal Processing* 67.1, pp. 97–109.

Levine, Yoav., Lenz, Barak., Dagan, Or., Ram, Ori., Padnos, Dan., Sharir, Or., Shalev-Shwartz, Shai., Shashua, Amnon., and Shoham, Yoav (July 2020). "SenseBERT: Driving Some Sense into BERT". *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan. Jurafsky, Joyce. Chai, Natalie. Schluter, and Joel Tetreault. Online: Association for Computational Linguistics, pp. 4656–4667. DOI: 10.18653/v1/2020.acl-main.423. URL: https://aclanthology.org/2020.acl-main.423.

Li, Bei., Wang, Ziyang., Liu, Hui., Jiang, Yufan., Du, Quan., Xiao, Tong., Wang, Huizhen., and Zhu, Jingbo (Nov. 2020). "Shallow-to-Deep Training for Neural Machine Translation". *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie. Webber, Trevor. Cohn, Yulan. He, and Yang Liu. Online: Association for Computational Linguistics, pp. 995–1005. DOI: 10.18653/v1/2020.emnlp-main.72. URL: https://aclanthology.org/2020.emnlp-main.72.

Li, Jiaoda., Cotterell, Ryan., and Sachan, Mrinmaya (2021). "Differentiable subset pruning of transformer heads". *Transactions of the Association for Computational Linguistics* 9, pp. 1442–1459.

Li, Junhui., Xiong, Deyi., Tu, Zhaopeng., Zhu, Muhua., Zhang, Min., and Zhou, Guodong (July 2017). "Modeling Source Syntax for Neural Machine Translation". *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Regina. Barzilay and Min-Yen Kan. Vancouver, Canada: Association for Computational Linguistics, pp. 688–697. DOI: 10.18653/v1/P17-1064. URL: https://aclanthology.org/P17-1064.

Li, Peng., Liu, Yang., Sun, Maosong., Izuha, Tatsuya., and Zhang, Dakun (2014). "A neural reordering model for phrase-based translation". *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pp. 1897–1907.

Li, Ruoyu., Wang, Sheng., Zhu, Feiyun., and Huang, Junzhou (2018). "Adaptive graph convolutional neural networks". *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1.

Li, Xiaoqing., Zhang, Jiajun., and Zong, Chengqing (2016). "Towards Zero Unknown Word in Neural Machine Translation." *IJCAI*, pp. 2852–2858.

Li, Yanyang., Wang, Qiang., Xiao, Tong., Liu, Tongran., and Zhu, Jingbo (2020). "Neural machine translation with joint representation". *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 05, pp. 8285–8292.

Li, Yujia., Vinyals, Oriol., Dyer, Chris., Pascanu, Razvan., and Battaglia, Peter (2018). "Learning deep generative models of graphs". *arXiv preprint arXiv:1803.03324*.

Li., Xiangju, Gao., Wei, Feng., Shi, Wang.., Daling, and Joty, Shafiq R. (2021). "Span-Level Emotion Cause Analysis by BERT-based Graph Attention Network". *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*.

Lin, Yongjie., Tan, Yi Chern., and Frank, Robert (Aug. 2019). "Open Sesame: Getting inside BERT's Linguistic Knowledge". *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Florence, Italy: Association for Computational Linguistics, pp. 241–253. DOI: 10.18653/v1/W19-4825. URL: https://aclanthology.org/W19-4825.

Liu, Peter J., Saleh, Mohammad., Pot, Etienne., Goodrich, Ben., Sepassi, Ryan., Kaiser, Lukasz., and Shazeer, Noam (2018). "Generating wikipedia by summarizing long sequences". *arXiv preprint arXiv:1801.10198*.

Liu, Xiaodong., Duh, Kevin., Liu, Liyuan., and Gao, Jianfeng (2020a). "Very deep transformers for neural machine translation". *arXiv preprint arXiv:2008.07772*.

Liu, Xuebo., Lai, Houtim., Wong, Derek F., and Chao, Lidia S (2020b). "Norm-Based Curriculum Learning for Neural Machine Translation". *Annual Meeting of the Association for Computational Linguistics*. URL: https://api.semanticscholar.org/CorpusID:219260306.

Liu, Yang (2017). "Recent advances in neural machine translation". *Journal of Computer Research and Development* 54.6, pp. 1144–1149.

Liu, Yang., Liu, Qun., and Lin, Shouxun (2006). "Tree-to-string alignment template for statistical machine translation". *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp. 609–616.

Liu, Yinhan., Ott, Myle., Goyal, Naman., Du, Jingfei., Joshi, Mandar., Chen, Danqi., Levy, Omer., Lewis, Mike., Zettlemoyer, Luke., and Stoyanov, Veselin (2019). "RoBERTa: A robustly optimized bert pretraining approach". *arXiv preprint arXiv:1907.11692*.

Luong, Thang., Pham, Hieu., and Manning, Christopher D. (Sept. 2015). "Effective Approaches to Attention-based Neural Machine Translation". *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1412–1421. DOI: 10.18653/v1/D15-1166. URL: https://aclanthology.org/D15-1166.

Lyu, Boer., Chen, Lu., Zhu, Su., and Yu, Kai (2021). "Let: Linguistic knowledge enhanced graph transformer for chinese short text matching". *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 15, pp. 13498–13506.

Ma, Chunpeng., Tamura, Akihiro., Utiyama, Masao., Sumita, Eiichiro., and Zhao, Tiejun (2020). "Syntax-based transformer for neural machine translation". *Journal of Natural Language Processing* 27.2, pp. 445–466.

Ma, Nianzu., Mazumder, Sahisnu., Wang, Hao., and Liu, Bing (2020). "Entity-aware dependency-based deep graph attention network for comparative preference classification". *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL-2020)*.

Mallinson, Jonathan., Sennrich, Rico., and Lapata, Mirella (2017). "Paraphrasing revisited with neural machine translation". *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 881–893.

Marcheggiani, Diego., Bastings, Jasmijn., and Titov, Ivan (2018). "Exploiting Semantics in Neural Machine Translation with Graph Convolutional Networks". *North American Chapter*

of the Association for Computational Linguistics. URL: https://api.semanticscholar.org/CorpusID:5063437.

Marneffe, Marie-Catherine. de and Nivre, Joakim (2019). "Dependency Grammar". *Annual Review of Linguistics* 5.1, pp. 197–218. DOI: 10.1146/annurev-linguistics-011718-011842. eprint: https://doi.org/10.1146/annurev-linguistics-011718-011842. URL: https://doi.org/10.1146/annurev-linguistics-011718-011842.

McDonald, Colin and Chiang, David (June 2021). "Syntax-Based Attention Masking for Neural Machine Translation". *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*. Online: Association for Computational Linguistics, pp. 47–52. DOI: 10.18653/v1/2021.naacl-srw.7. URL: https://aclanthology.org/2021.naacl-srw.7.

Meng, Fandong., Lu, Zhengdong., Wang, Mingxuan., Li Hang.and Jiang, Wenbin., and Liu, Qun (July 2015). "Encoding Source Language with Convolutional Neural Network for Machine Translation". *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by Chengqing. Zong and Michael Strube. Beijing, China: Association for Computational Linguistics, pp. 20–30. DOI: 10.3115/v1/P15-1003. URL: https://aclanthology.org/P15-1003.

Merchant, Amil., Rahimtoroghi, Elahe., Pavlick, Ellie., and Tenney, Ian (2020). "What Happens To BERT Embeddings During Fine-tuning?" *BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*. URL: https://api.semanticscholar.org/CorpusID:216914339.

Mikolov, Tomas., Chen, Kai., Corrado, Greg., and Dean, Jeffrey (2013a). "Efficient Estimation of Word Representations in Vector Space". *International Conference on Learning Representations*. URL: https://api.semanticscholar.org/CorpusID:5959482.

Mikolov, Tomas., Sutskever, Ilya., Chen, Kai., Corrado, Greg S., and Dean, Jeff (2013b). "Distributed representations of words and phrases and their compositionality". *Advances in neural information processing systems* 26.

Müller, Samuel G and Hutter, Frank (2021). "Trivialaugment: Tuning-free yet state-of-the-art data augmentation". *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 774–782.

Müller, Stefan (2016). "Grammatical theory: From transformational grammar to constraint-based approaches". URL: https://api.semanticscholar.org/CorpusID:125362772.

Nadejde, Maria., Reddy, Siva., Sennrich, Rico., Dwojak, Tomasz., Junczys-Dowmunt, Marcin., Koehn, Philipp., and Birch, Alexandra (2017). "Predicting Target Language CCG Supertags Improves Neural Machine Translation". *Conference on Machine Translation*. URL: https://api.semanticscholar.org/CorpusID:215826482.

Nagao, Makoto (1984). "A framework of a mechanical translation between Japanese and English by analogy principle". *Artificial and human intelligence*, pp. 351–354.

Nguyen, Thai Phuong., Shimazu, Akira., Ho, Tu-Bao., Le Nguyen, Minh., and Van Nguyen, Vinh (2008). "A tree-to-string phrase-based model for statistical machine translation". *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pp. 143–150.

Nguyen, Xuan-Phi., Joty, Shafiq., Hoi, Steven CH., and Socher, Richard (2020). "Tree-Structured Attention with Hierarchical Accumulation". *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=HJxK5pEYvr.

Nirenburg, Sergei (1989). "Knowledge-based machine translation". *Machine Translation* 4.1, pp. 5–24.

Niven, Timothy and Kao, Hung-Yu (July 2019). "Probing Neural Network Comprehension of Natural Language Arguments". *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 4658–4664. DOI: 10.18653/v1/P19-1459. URL: https://aclanthology.org/P19-1459.

Nivre, Joakim, Marneffe, Marie-Catherine de, Ginter, Filip, Goldberg, Yoav, Hajic, Jan, Manning, Christopher D., McDonald, Ryan T., Petrov, Slav, Pyysalo, Sampo, Silveira, Natalia, Tsarfaty, Reut, and Zeman, Daniel (2016). "Universal Dependencies v1: A Multilingual Tree-

bank Collection". *International Conference on Language Resources and Evaluation*. URL: https://api.semanticscholar.org/CorpusID:17954486.

Novikova, Jekaterina., Dušek, Ondřej., Curry, Amanda Cercas., and Rieser, Verena (2017). "Why We Need New Evaluation Metrics for NLG". *Conference on Empirical Methods in Natural Language Processing*. URL: https://api.semanticscholar.org/CorpusID:1929239.

Ott, Myle., Edunov, Sergey., Grangier, David., and Auli, Michael (Oct. 2018). "Scaling Neural Machine Translation". *Proceedings of the Third Conference on Machine Translation: Research Papers*. Brussels, Belgium: Association for Computational Linguistics, pp. 1–9. DOI: 10.18653/v1/W18-6301. URL: https://aclanthology.org/W18-6301.

Papadimitriou, Isabel., Chi, Ethan A., Futrell, Richard., and Mahowald, Kyle (Apr. 2021). "Deep Subjecthood: Higher-Order Grammatical Features in Multilingual BERT". *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, pp. 2522–2532. DOI: 10.18653/v1/2021.eacl-main.215. URL: https://aclanthology.org/2021.eacl-main.215.

Pennington, Jeffrey., Socher, Richard., and Manning, Christopher D (2014). "Glove: Global vectors for word representation". *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.

Peters, Matthew E., Neumann, Mark., Iyyer, Mohit., Gardner, Matt., Clark, Christopher., Lee, Kenton., and Zettlemoyer, Luke (June 2018). "Deep Contextualized Word Representations". *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 2227–2237. DOI: 10.18653/v1/N18-1202. URL: https://aclanthology.org/N18-1202.

Pires, Telmo., Schlinger, Eva., and Garrette, Dan (July 2019). "How Multilingual is Multilingual BERT?" *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 4996–5001. DOI: 10.18653/v1/P19-1493. URL: https://aclanthology.org/P19-1493.

Poerner, Nina., Waltinger, Ulli., and Schütze, Hinrich (2019). "BERT is not a knowledge base (yet): Factual knowledge vs. name-based reasoning in unsupervised QA". *arXiv preprint arXiv:1911.03681* 3.

Radford, Alec and Narasimhan, Karthik (2018). "Improving language understanding by generative pre-training". URL: https://api.semanticscholar.org/CorpusID:49313245.

Radford, Alec., Wu, Jeffrey., Child, Rewon., Luan, David., Amodei, Dario., Sutskever, Ilya., et al. (2019). "Language models are unsupervised multitask learners". *OpenAI blog* 1.8, p. 9.

Raganato, Alessandro and Tiedemann, Jörg (Nov. 2018). "An Analysis of Encoder Representations in Transformer-Based Machine Translation". *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, pp. 287–297. DOI: 10.18653/v1/W18-5431. URL: https://aclanthology.org/W18-5431.

Ranasinghe, Tharindu., Orasan, Constantin., and Mitkov, Ruslan (2020). "TransQuest at WMT2020: Sentence-Level Direct Assessment". *Proceedings of the Fifth Conference on Machine Translation*.

Rifai, Salah., Vincent, Pascal., Muller, Xavier., Glorot, Xavier., and Bengio, Yoshua (2011). "Contractive auto-encoders: Explicit invariance during feature extraction". *Proceedings of the 28th international conference on international conference on machine learning*, pp. 833–840.

Rust, Phillip., Lotz, Jonas F., Bugliarello, Emanuele., Salesky, Elizabeth., Lhoneux, Miryam. de, and Elliott, Desmond (2022). "Language modelling with pixels". *arXiv preprint arXiv:2207.06991*.

Sennrich, Rico., Haddow, Barry., and Birch, Alexandra (Aug. 2016a). "Neural Machine Translation of Rare Words with Subword Units". *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Katrin. Erk and Noah A. Smith. Berlin, Germany: Association for Computational Linguistics, pp. 1715–1725. DOI: 10.18653/v1/P16-1162. URL: https://aclanthology.org/P16-1162.

– (2016b). "Edinburgh Neural Machine Translation Systems for WMT 16". *Conference on Machine Translation*. URL: https://api.semanticscholar.org/CorpusID:14919987.

Serrano, Sofia and Smith, Noah A. (July 2019). "Is Attention Interpretable?" *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 2931–2951. DOI: 10.18653/v1/P19-1282. URL: https://aclanthology.org/P19-1282.

Sharoff, Serge., Rapp, Reinhard., and Zweigenbaum, Pierre (2023). *Building and Using Comparable Corpora for Multilingual Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Springer Nature.

Shavarani, Hassan S. and Sarkar, Anoop (2021). "Better Neural Machine Translation by Extracting Linguistic Information from BERT". *Conference of the European Chapter of the Association for Computational Linguistics*. URL: https://api.semanticscholar.org/CorpusID:233169135.

Shaw, Peter., Uszkoreit, Jakob., and Vaswani, Ashish (June 2018). "Self-Attention with Relative Position Representations". *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Ed. by Marilyn. Walker, Heng. Ji, and Amanda Stent. New Orleans, Louisiana: Association for Computational Linguistics, pp. 464–468. DOI: 10.18653/v1/N18-2074. URL: https://aclanthology.org/N18-2074.

Shen, Shiqi., Cheng, Yong., He, Zhongjun., He, Wei., Wu, Hua., Sun, Maosong., and Liu, Yang (2015). "Minimum Risk Training for Neural Machine Translation". *ArXiv* abs/1512.02433. URL: https://api.semanticscholar.org/CorpusID:3913537.

Shi, Xing., Padhi, Inkit., and Knight, Kevin (2016). "Does string-based neural MT learn source syntax?" *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 1526–1534.

Shorten, Connor., Khoshgoftaar, Taghi M., and Furht, Borko (2021). "Text data augmentation for deep learning". *Journal of big Data* 8, pp. 1–34.

Slobodkin, Aviv., Choshen, Leshem., and Abend, Omri (July 2022). "Semantics-aware Attention Improves Neural Machine Translation". *Proceedings of the 11th Joint Conference on Lexical and Computational Semantics*. Ed. by Vivi. Nastase, Ellie. Pavlick, Mohammad Taher. Pile-

hvar, Jose. Camacho-Collados, and Alessandro Raganato. Seattle, Washington: Association for Computational Linguistics, pp. 28–43. DOI: 10.18653/v1/2022.starsem-1.3. URL: https://aclanthology.org/2022.starsem-1.3.

So, David., Le, Quoc., and Liang, Chen (2019). "The evolved transformer". *International conference on machine learning*. PMLR, pp. 5877–5886.

Socher, Richard., Bauer, John., Manning, Christopher D., and Ng, Andrew Y (2013). "Parsing with compositional vector grammars". *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 455–465.

Soky, Kak., Li, Sheng., Mimura, Masato., Chu, Chenhui., and Kawahara, Tatsuya (2022). "Leveraging Simultaneous Translation for Enhancing Transcription of Low-resource Language via Cross Attention Mechanism". *Interspeech*. URL: https://api.semanticscholar.org/CorpusID:252340636.

Song, Linfeng., Gildea, Daniel., Zhang, Yue., Wang, Zhiguo., and Su, Jinsong (2019). "Semantic neural machine translation using AMR". *Transactions of the Association for Computational Linguistics* 7, pp. 19–31.

Sun, Chi., Huang, Luyao., and Qiu, Xipeng (June 2019). "Utilizing BERT for Aspect-Based Sentiment Analysis via Constructing Auxiliary Sentence". *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)*. Ed. by Jill. Burstein, Christy. Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 380–385. DOI: 10.18653/v1/N19-1035. URL: https://aclanthology.org/N19-1035.

Sun, Yu., Wang, Shuohuan., Li, Yukun., Feng, Shikun., Chen, Xuyi., Zhang, Han., Tian, Xin., Zhu, Danxiang., Tian, Hao., and Wu, Hua (2019). "ERNIE: Enhanced representation through knowledge integration". *arXiv preprint arXiv:1904.09223*.

Sutskever, Ilya., Vinyals, Oriol., and Le, Quoc V (2014). "Sequence to sequence learning with neural networks". *Advances in neural information processing systems* 27.

Tamura, Akihiro., Watanabe, Taro., and Sumita, Eiichiro (2014). "Recurrent neural networks for word alignment model". *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1470–1480.

Tenney, Ian., Xia, Patrick., Chen, Berlin., Wang, Alex., Poliak, Adam., McCoy, R Thomas., Kim, Najoung., Van Durme, Benjamin., Bowman, Samuel R., Das, Dipanjan., and Pavlick, Ellie (2019). *What do you learn from context? Probing for sentence structure in contextualized word representations*. cite arxiv:1905.06316Comment: ICLR 2019 camera-ready version, 17 pages including appendices. URL: http://arxiv.org/abs/1905.06316.

Tu, Zhaopeng., Liu, Yang., Lu, Zhengdong., Liu, Xiaohua., and Li, Hang (2017). "Context gates for neural machine translation". *Transactions of the Association for Computational Linguistics* 5, pp. 87–99.

Üstün, Ahmet., Berard, Alexandre., Besacier, Laurent., and Gallé, Matthias (Nov. 2021). "Multilingual Unsupervised Neural Machine Translation with Denoising Adapters". *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Ed. by Marie-Francine. Moens, Xuanjing. Huang, Lucia. Specia, and Scott Wen-tau Yih. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 6650–6662. DOI: 10.18653/v1/2021.emnlp-main.533. URL: https://aclanthology.org/2021.emnlp-main.533.

Vaswani, Ashish., Shazeer, Noam., Parmar, Niki., Uszkoreit, Jakob., Jones, Llion., Gomez, Aidan N., Kaiser, Łukasz, and Polosukhin, Illia (2017). "Attention is all you need". *Advances in neural information processing systems* 30.

Veličković, Petar., Cucurull, Guillem., Casanova, Arantxa., Romero, Adriana., Lio, Pietro., and Bengio, Yoshua (2017). "Graph attention networks". *arXiv preprint arXiv:1710.10903*.

Voita, Elena., Talbot, David., Moiseev, Fedor., Sennrich, Rico., and Titov, Ivan (July 2019). "Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned". *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 5797–5808. DOI: 10.18653/v1/P19-1580. URL: https://aclanthology.org/P19-1580.

Wallace, Eric., Wang, Yizhong., Li, Sujian., Singh, Sameer., and Gardner, Matt (2019). "Do NLP Models Know Numbers? Probing Numeracy in Embeddings". *Conference on Empirical Methods in Natural Language Processing*. URL: https://api.semanticscholar.org/CorpusID:202583694.

Wang, Chengyi., Wu, Shuangzhi., and Liu, Shujie (2019). "Source Dependency-Aware Transformer with Supervised Self-Attention". *CoRR* abs/1909.02273. arXiv: 1909.02273. URL: http://arxiv.org/abs/1909.02273.

Wang, Kai., Shen, Weizhou., Yang, Yunyi., Quan, Xiaojun., and Wang, Rui (2020). "Relational Graph Attention Network for Aspect-based Sentiment Analysis". *Annual Meeting of the Association for Computational Linguistics*. URL: https://api.semanticscholar.org/CorpusID:216553425.

Wang, Qiang., Li, Fuxue., Xiao, Tong., Li, Yanyang., Li, Yinqiao., and Zhu, Jingbo (Aug. 2018). "Multi-layer Representation Fusion for Neural Machine Translation". *Proceedings of the 27th International Conference on Computational Linguistics*. Ed. by Emily M. Bender, Leon. Derczynski, and Pierre Isabelle. Santa Fe, New Mexico, USA: Association for Computational Linguistics, pp. 3015–3026. URL: https://aclanthology.org/C18-1255.

Wang, Rui., Utiyama, Masao., Liu, Lemao., Chen, Kehai., and Sumita, Eiichiro (2017). "Instance weighting for neural machine translation domain adaptation". *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1482–1488.

Wang, Wei., Bi, Bin., Yan, Ming., Wu, Chen., Bao, Zuyi., Xia, Jiangnan., Peng, Liwei., and Si, Luo (2020). "StructBERT: Incorporating Language Structures into Pre-training for Deep Language Understanding". *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=BJgQ4lSFPH.

Wang, Xing., Tu, Zhaopeng., Wang, Longyue., and Shi, Shuming (July 2019). "Exploiting Sentential Context for Neural Machine Translation". *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 6197–6203. DOI: 10.18653/v1/P19-1624. URL: https://aclanthology.org/P19-1624.

Weng, Rongxiang., Yu, Heng., Huang, Shujian., Cheng, Shanbo., and Luo, Weihua (2020). "Acquiring knowledge from pre-trained model to neural machine translation". *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 05, pp. 9266–9273.

Wiegreffe, Sarah and Pinter, Yuval (Nov. 2019). "Attention is not not Explanation". *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 11–20. DOI: 10.18653/v1/D19-1002. URL: https://aclanthology.org/D19-1002.

Wieting, John., Mallinson, Jonathan., and Gimpel, Kevin (2017). "Learning paraphrastic sentence embeddings from back-translated bitext". *arXiv preprint arXiv:1706.01847*.

Wu, Felix., Fan, Angela., Baevski, Alexei., Dauphin, Yann N., and Auli, Michael (2019). "Pay Less Attention with Lightweight and Dynamic Convolutions". *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=SkVhlh09tX.

Wu, Shuangzhi., Zhang, Dongdong., Yang, Nan., Li, Mu., and Zhou, Ming (July 2017a). "Sequence-to-Dependency Neural Machine Translation". *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 698–707. DOI: 10.18653/v1/P17-1065. URL: https://aclanthology.org/P17-1065.

Wu, Shuangzhi., Zhou, Ming., and Zhang, Dongdong (2017b). "Improved Neural Machine Translation with Source Syntax." *IJCAI*, pp. 4179–4185.

Wu, Xueqing., Xia, Yingce., Zhu, Jinhua., Wu, Lijun., Xie, Shufang., and Qin, Tao (2022). "A study of BERT for context-aware neural machine translation". *Machine Learning* 111.3, pp. 917–935.

Wu, Yonghui., Schuster, Mike., Chen, Zhifeng., Le, Quoc V., Norouzi, Mohammad., Macherey, Wolfgang., Krikun, Maxim., Cao, Yuan., Gao, Qin., Macherey, Klaus., et al. (2016). "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". *CoRR* abs/1609.08144. arXiv: 1609.08144. URL: http://arxiv.org/abs/1609.08144.

Wu, Zonghan., Pan, Shirui., Long, Guodong., Jiang, Jing., and Zhang, Chengqi (2019). "Graph WaveNet for Deep Spatial-Temporal Graph Modeling". *International Joint Conference on Artificial Intelligence*. URL: https://api.semanticscholar.org/CorpusID:173990443.

Xie, Jun., Mi, Haitao., and Liu, Qun (2011). "A novel dependency-to-string model for statistical machine translation". *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 216–226.

Xiong, Wayne., Droppo, Jasha., Huang, Xuedong., Seide, Frank., Seltzer, Mike., Stolcke, Andreas., Yu, Dong., and Zweig, Geoffrey (2016). "Achieving human parity in conversational speech recognition". *arXiv preprint arXiv:1610.05256*.

Yang, Baosong., Tu, Zhaopeng., Wong, Derek F., Meng, Fandong, Chao, Lidia S., and Zhang, Tong (Oct. 2018). "Modeling Localness for Self-Attention Networks". *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Ed. by Ellen. Riloff, David. Chiang, Julia. Hockenmaier, and Jun'ichi Tsujii. Brussels, Belgium: Association for Computational Linguistics, pp. 4449–4458. DOI: 10.18653/v1/D18-1475. URL: https://aclanthology.org/D18-1475.

Yang, Baosong., Wang, Longyue., Wong, Derek F., Chao, Lidia S., and Tu, Zhaopeng (June 2019). "Convolutional Self-Attention Networks". *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)*. Ed. by Jill. Burstein, Christy. Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4040–4045. DOI: 10.18653/v1/N19-1407. URL: https://aclanthology.org/N19-1407.

Yang, Jiacheng., Wang, Mingxuan., Zhou, Hao., Zhao, Chengqi., Zhang, Weinan., Yu, Yong., and Li, Lei (2020). "Towards making the most of BERT in neural machine translation". *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 05, pp. 9378–9385.

Yang, Nan., Liu, Shujie., Li, Mu., Zhou, Ming., and Yu, Nenghai (2013). "Word alignment modeling with context dependent deep neural network". *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 166–175.

Yin., Yongjing, Meng., Fandong, Su., Jinsong, Zhou., Chulun, Yang., Zhengyuan, Zhou., Jie, and Luo, Jiebo (2020). "A Novel Graph-based Multi-modal Fusion Encoder for Neural Machine Translation". *Annual Meeting of the Association for Computational Linguistics*. URL: https://api.semanticscholar.org/CorpusID:220045417.

You, Jiaxuan., Ying, Rex., Ren, Xiang., Hamilton, William., and Leskovec, Jure (2018). "Graphrnn: Generating realistic graphs with deep auto-regressive models". *International conference on machine learning*. PMLR, pp. 5708–5717.

Yu, Shanshan., Su, Jindian., and Luo, Da (2019). "Improving bert-based text classification with auxiliary sentence and domain knowledge". *IEEE Access* 7, pp. 176600–176612.

Zhang, Jiajun., Liu, Shujie., Li, Mu., Zhou, Ming., and Zong, Chengqing (2014). "Mind the gap: Machine translation by minimizing the semantic gap in embedding space". *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 28. 1.

Zhang, Jingyi., Utiyama, Masao., Sumita, Eiichiro., and Zhao, Hai (2015). "Learning word reorderings for hierarchical phrase-based statistical machine translation". *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 542–548.

Zhang, Meishan., Li, Zhenghua., Fu, Guohong., and Zhang, Min (June 2019). "Syntax-Enhanced Neural Machine Translation with Syntax-Aware Word Representations". *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)*. Ed. by Jill. Burstein, Christy. Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 1151–1161. DOI: 10.18653/v1/N19-1118. URL: https://aclanthology.org/N19-1118.

Zhang, Zhebin., Wu, Sai., Jiang, Dawei., and Chen, Gang (2021). "BERT-JAM: Maximizing the utilization of BERT for neural machine translation". *Neurocomputing* 460, pp. 84–94.

Zhang, Zhuosheng., Wu, Yuwei., Zhou, Junru., Duan, Sufeng., Zhao, Hai., and Wang, Rui (2020). "SG-Net: Syntax guided transformer for language representation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zhou, Xiaotang., Zhang, Tao., Cheng, Chao., and Song, Shinan (2022). "Dynamic multichannel fusion mechanism based on a graph attention network and BERT for aspect-based sentiment classification". *Applied Intelligence*, pp. 1–14.

Zhu., Jinhua, Xia., Yingce, Wu., Lijun, He., Di, Qin., Tao, Zhou., Wengang, Li., Houqiang, and Liu, Tieyan (2020). "Incorporating BERT into Neural Machine Translation". *International Conference on Learning Representations*. URL: `https://openreview.net/forum?id=Hyl7ygStwB`.

Zou, Will Y., Socher, Richard., Cer, Daniel., and Manning, Christopher D (2013). "Bilingual word embeddings for phrase-based machine translation". *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1393–1398.

Appendix A

Difficulties of BERT and Translation

A.1 Investigation of Syntax in BERT

Figures A.1 to Figures A.6 demonstrate the results of the syntactic probing experiments performed on BERT in Chapter 3.3. For each dependency relation result, the x-axis represents the 12 layers in BERT, and the y-axis represents the F1-score from 0 to 1, with higher scores representing more familiarity with the model with this dependency relation. The syntactic probing experiments for all languages (Zh, Ru, De) are conducted based on the PUD and GSD corpus and investigate how BERT detects syntactic knowledge before and after MT fine-tuning. Given the small number of partial dependency relations in the PUD or GSD corpus, the experiments do not record their results. Table A.1 shows the syntactic patterns for each dependency relation in all three languages, where *S* denotes *Smooth*, *C+D* denotes *Climb + Decline*, and *F* denotes *Fluctuate*. Two syntactic patterns are shown in some dependency relations, which are influenced by differences in both PUD and GSD corpuses.

A.2 Investigation of Translation Quality

Tables A.2 to Tables A.4 illustrate the quantitative relationship between dependency relations in high-quality and low-quality translations in different translation tasks in Chapter 3.4. χ^2 represents the theory based on the chi-square goodness of fit test to show the gap of the given dependency relation in different quality translations. Layer-6 and Layer-12 are from the syntactic probing experiments in Chapter 3.3 and show the F1-score of this dependency relation in the middle and top layers of BERT.

	Zh	Ru	De
acl	S	C+D	C+D
acl:relcl	C+D	C+D	C+D
advcl	C+D	F	F
advmod	C+D	S	C+D
amod	C+D	S	C+D
appos	C+D	C+D	C+D
aux	C+D	-	C+D
aux:pass	S	S	S
case	C+D	S	S
case:loc	C+D	-	-
cc	S	S	S
ccomp	C+D / F	C+D	C+D
clf	C+D	-	-
compound	C+D	-	F
compound:prt	-	-	F
conj	C+D	C+D	C+D
cop	S	C+D	C+D / S
csubj	F	F	F
dep	C+D	-	S
det	C+D	S	S
discourse	S	-	-
discourse:sp	C+D	-	-
dislocated	F	-	-
expl	-	-	F
fixed	-	C+D	-
flat	C+D	S / F	S / F
flat:foreign	S	C+D / F	-
flat:name	C+D	C+D	C+D
iobj	-	F	F
mark	C+D	C+D	S
mark:adv	S	-	-
mark:prt	C+D	-	-
mark:relcl	C+D	-	-
nmod	C+D	C+D	C+D
nmod:poss	-	-	C+D / F
nmod:tmod	C+D	-	-
nsubj	C+D	C+D	C+D
nsubj:pass	F	C+D / F	F
nummod	S	F	C+D
nummod:gov	-	C+D	-
obj	C+D	C+D	C+D
obl	C+D	C+D	C+D
obl:agent	-	F	-
obl:patient	F	-	-
obl:tmod	C+D	-	C+D
expl	-	-	C+D / F
parataxis	-	C+D	F
root	C+D	C+D	C+D
xcomp	C+D	C+D	C+D

Table A.1: The corresponding syntactic patterns for each dependency relation in all three languages.

Languages	Dependencies	High-quality	Low-quality	χ^2	Layer-6	Layer-12
Zh	acl:relcl	67	99	15.28	0.67	0.42
	acl	3	1	1.33	-	-
	advcl	62	113	41.95	0.43	0.33
	advmod	231	250	1.56	0.79	0.66
	amod	95	75	4.21	0.69	0.43
	appos	10	73	396.9	0.4	0.48
	aux	130	132	0.03	0.9	0.73
	aux:pass	20	10	5	0.86	0.84
	case	214	286	24.22	0.82	0.76
	case:loc	63	77	3.11	0.87	0.67
	cc	52	48	0.3	0.91	0.84
	ccomp	70	72	0.05	0.30	0.24
	clf	65	77	2.21	0.94	0.84
	compound	267	333	16.31	0.70	0.60
	conj	61	71	1.63	0.52	0.49
	cop	40	65	15.62	0.62	0.55
	dep	42	99	77.35	0.28	0.29
	discourse:sp	16	21	1.56	0.90	0.75
	flat	2	21	242	0.70	0.74
	flat:name	1	53	2704	0.68	0.78
	mark	32	66	36.12	0.72	0.54
	mark:prt	40	45	0.625	0.62	0.44
	mark:relcl	51	69	6.35	0.86	0.88
	nmod	123	145	3.96	0.45	0.34
	nsubj	293	380	25.83	0.63	0.60
	nsubj:pass	17	9	3.76	0	0.12
	nummod	137	169	7.47	0.93	0.85
	obj	285	297	5.89	0.60	0.51
	obl	86	132	24.6	0.38	0.21
	obl:tmod	28	54	24.14	0.68	0.46
	xcomp	76	111	16.11	0.34	0.32

Table A.2: Syntactic dependencies with the value of χ^2 in Chinese.

Languages	Dependencies	High-quality	Low-quality	χ^2	Layer-6	Layer-12
	acl:relcl	21	28	2.33	0.43	0.47
	acl	35	41	1.02	0.65	0.52
	advcl	44	26	7.36	0.37	0.26
	advmod	189	162	3.85	0.93	0.83
	amod	274	347	19.44	0.90	0.89
	appos	8	31	66.12	0.36	0.47
	aux:pass	24	27	0.375	0.94	0.94
	case	306	406	32.67	0.98	0.91
	cc	82	123	20.5	0.98	0.93
	ccomp	28	22	1.28	0.30	0.50
	conj	93	147	31.35	0.63	0.72
	cop	13	17	1.23	0.76	0.76
	csubj	5	7	0.8	0.44	0
	det	75	98	7.05	0.91	0.84
	flat	1	31	900	0.22	0.18
Ru	flat:name	12	57	168.75	0.82	0.86
	flat:foreign	1	55	2916	0.69	0.18
	fixed	41	39	0.09	0.69	0.59
	mark	53	45	1.2	0.85	0.75
	nmod	309	330	1.42	0.69	0.68
	nsubj	243	273	3.7	0.75	0.73
	nsubj:pass	34	35	0.02	0.12	0.21
	nummod	34	29	0.73	0.63	0.55
	nummod:gov	10	20	10	0.67	0.44
	obj	21	35	9.33	0.63	0.62
	obl	115	137	4.2	0.71	0.66
	iobj	207	307	48.3	0.4	0.53
	xcomp	65	55	1.53	0.67	0.60
	parataxis	21	50	40.04	0.48	0.48

Table A.3: Syntactic dependencies with the value of χ^2 in Russian.

Languages	Dependencies	High-quality	Low-quality	χ^2	Layer-6	Layer-12
	acl:relcl	43	56	3.93	0.67	0.66
	advcl	27	47	14.8	0.34	0.36
	advmod	222	208	0.88	0.86	0.64
	amod	197	217	2.03	0.83	0.63
	appos	10	96	739.6	0.42	0.25
	aux	70	52	4.62	0.95	0.82
	aux:pass	47	44	0.19	0.94	0.83
	case	324	459	56.25	0.97	0.84
	cc	129	142	1.31	0.96	0.82
	ccomp	20	26	1.8	0.09	0.32
	compound	25	72	88.36	0.47	0.51
	compound:prt	10	34	57.6	0.92	0.48
	conj	140	172	7.31	0.5	0.57
	cop	47	61	4.17	0.93	0.75
De	det	469	531	8.19	0.98	0.90
	flat	0	9	-	0.25	0
	flat:name	6	61	504.16	0.52	0.39
	mark	73	91	4.43	0.98	0.79
	nmod	193	177	1.32	0.57	0.58
	nmod:poss	39	67	20.1	0.96	0.85
	nsubj	241	308	18.62	0.73	0.68
	nsubj:pass	44	34	2.27	0.26	0.46
	nummod	40	45	0.62	0.76	0.73
	obj	154	179	4.05	0.59	0.59
	obl	212	327	62.38	0.63	0.61
	obl:tmod	14	34	28.57	0.55	0.62
	expl	22	11	5.5	0.90	0.76
	iobj	15	10	1.66	0.54	0.53
	xcomp	33	38	0.75	0.42	0.43
	parataxis	11	15	1.45	0	0

Table A.4: Syntactic dependencies with the value of χ^2 in German.

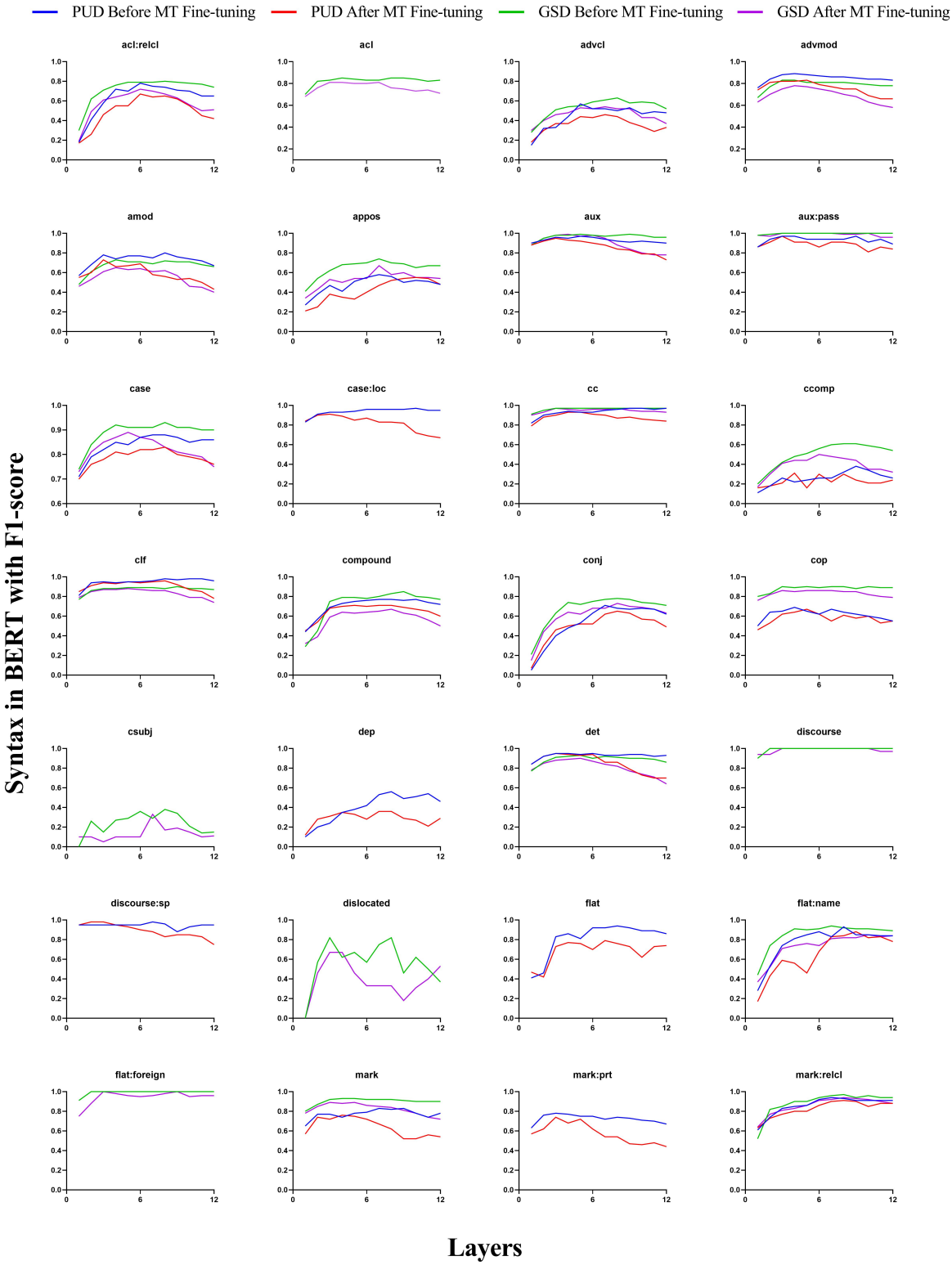


Figure A.1: Detection of dependency relations in Chinese by syntactic probing experiments.

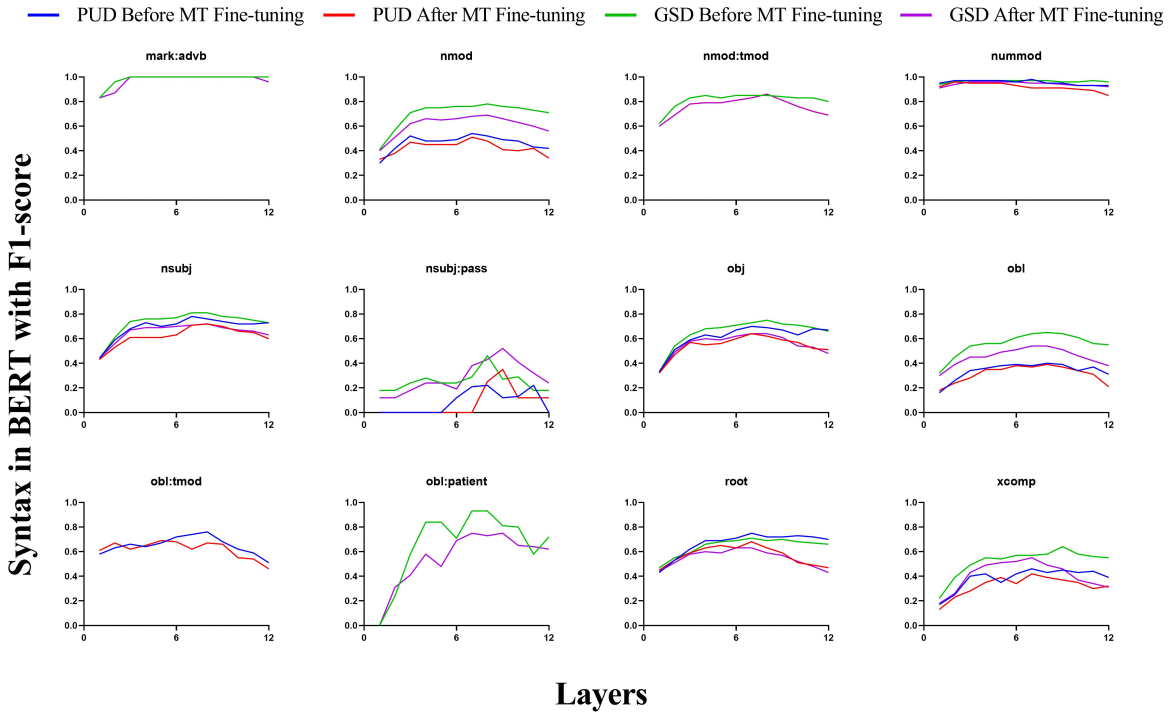


Figure A.2: Detection of dependency relations in Chinese by syntactic probing experiments.

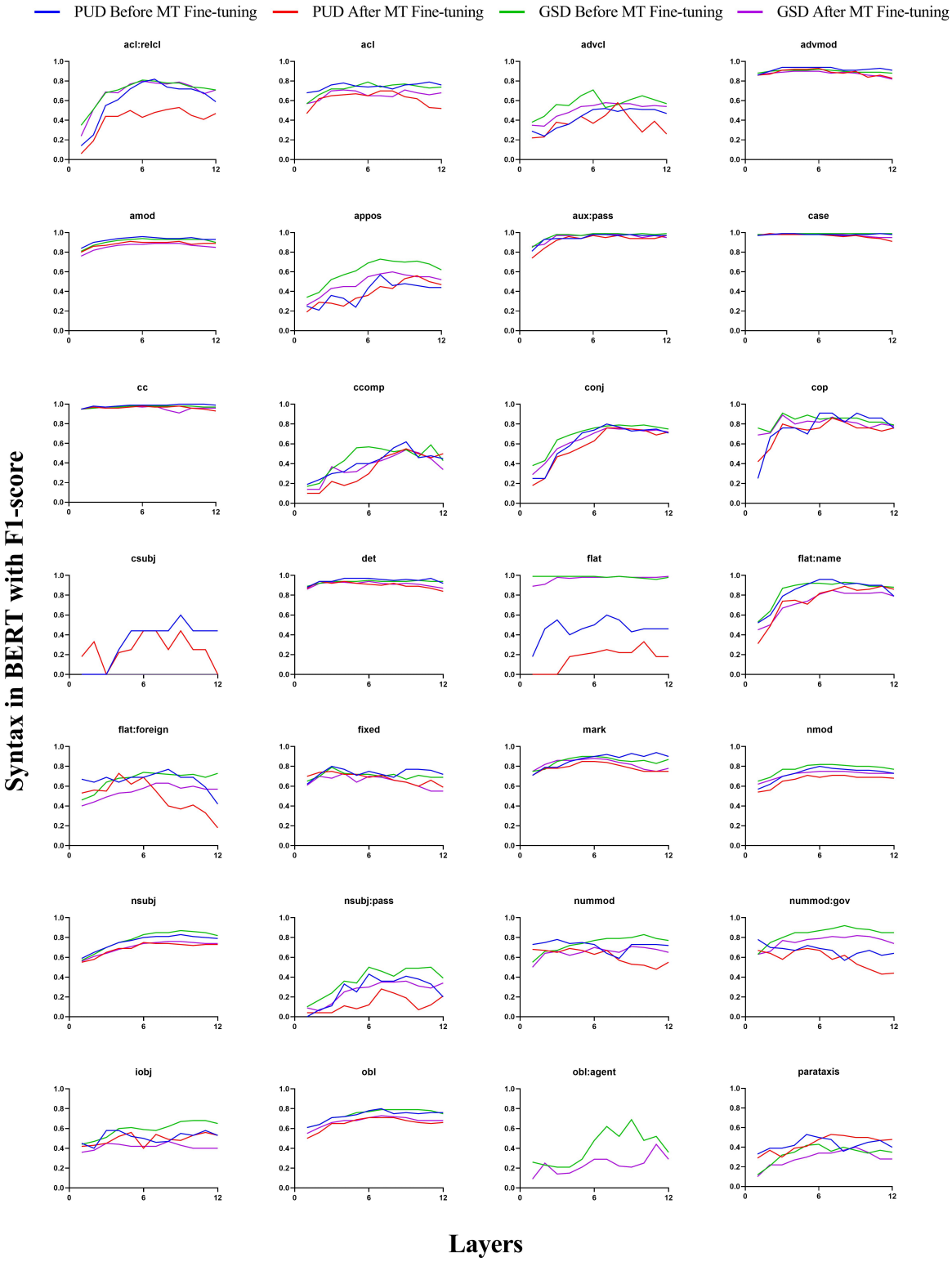


Figure A.3: Detection of dependency relations in Russian by syntactic probing experiments.

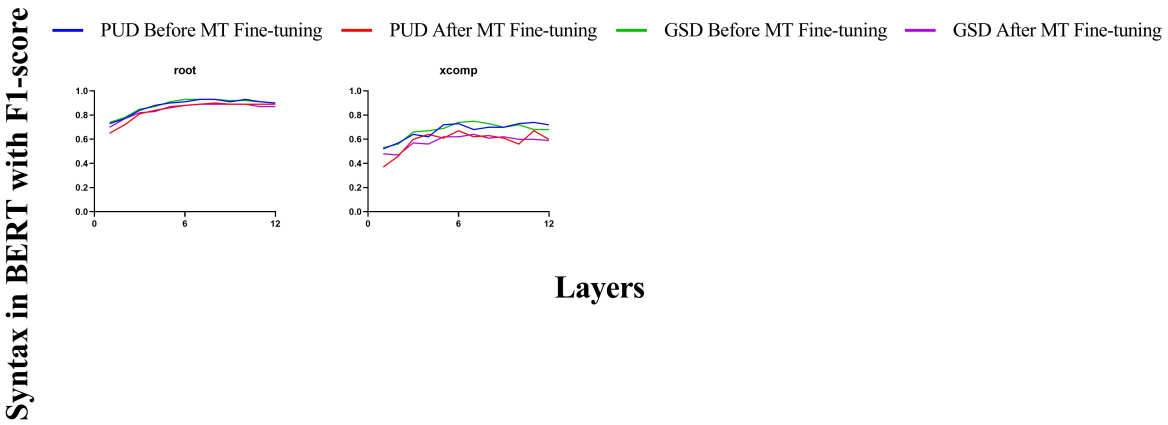


Figure A.4: Detection of dependency relations in Russian by syntactic probing experiments.

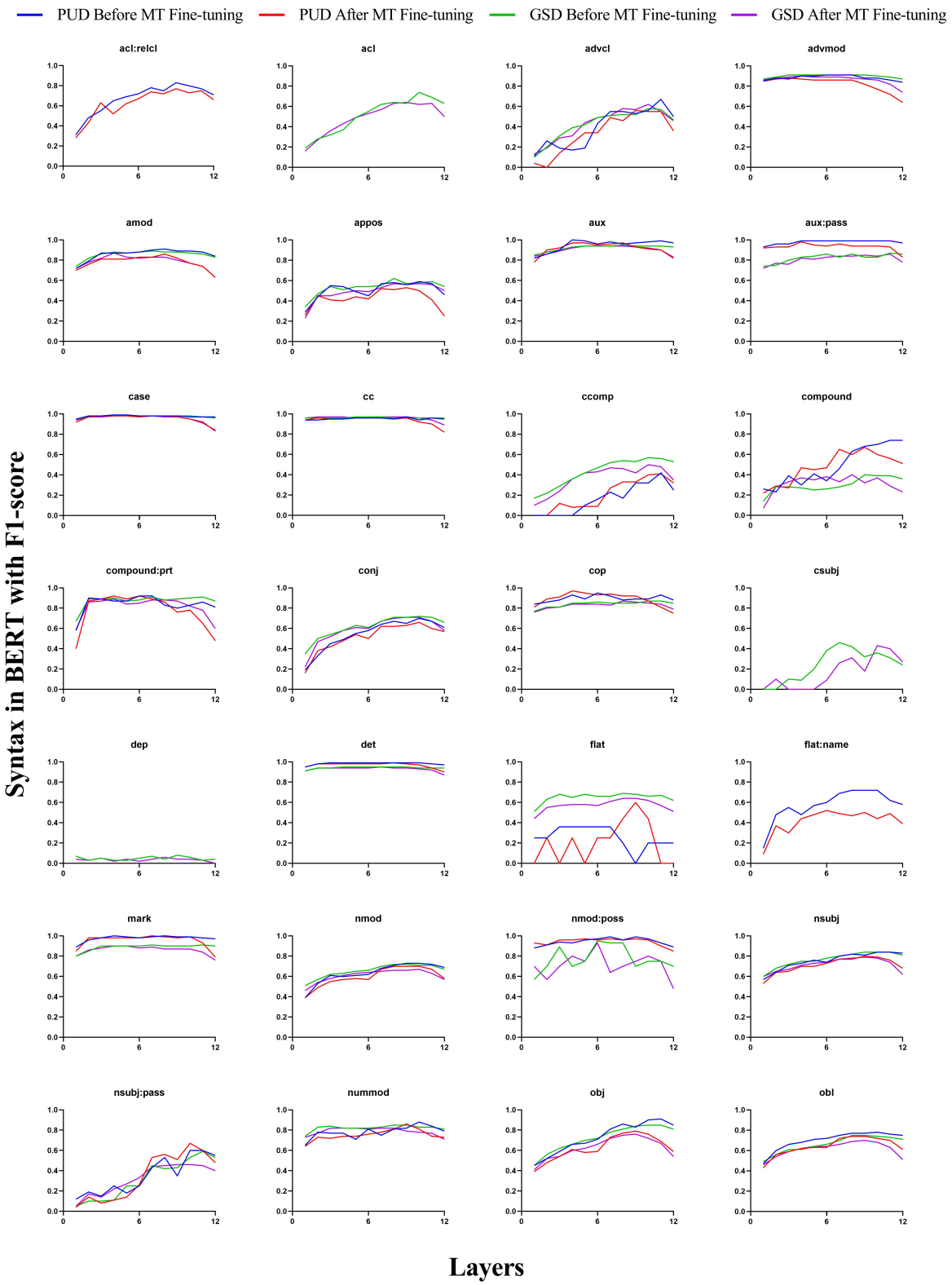


Figure A.5: Detection of dependency relations in German by syntactic probing experiments.

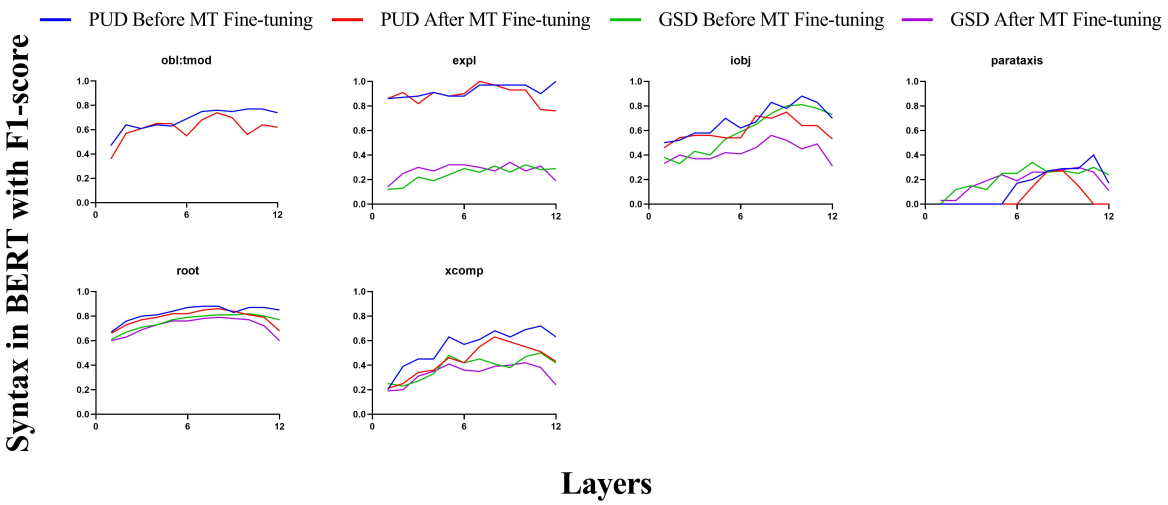


Figure A.6: Detection of dependency relations in German by syntactic probing experiments.

Appendix B

Syntactic Interpretability for GAT

B.1 Investigation of Syntax in GAT

Tables B.1 to B.5 show the prediction scores of GAT for syntactic knowledge of the three languages (Zh, Ru, De) under different numbers of attention heads (A) and layers (L). The PUD corpus provides all the syntactic knowledge, and given the insufficient number of partial dependency relations, the experiments do not record their results. Differences may arise in the kinds of dependency relations between languages due to the diversity of syntactic knowledge.

					Zh					
L-A	acl:relcl	advcl	advmod	amod	appos	aux	case	case:loc	cc	ccomp
2–2	0.82	0	0.90	0.80	0.60	0.90	0.98	0.95	0.99	0.41
2–4	0.83	0	0.90	0.81	0.55	0.91	0.99	0.94	0.99	0.40
2–6	0.87	0.14	0.91	0.85	0.61	0.91	0.99	0.91	0.99	0.53
2–8	0.84	0.15	0.90	0.80	0.58	0.91	0.99	0.94	0.99	0.30
3–2	0.87	0	0.90	0.84	0.54	0.90	0.99	0.92	0.99	0.66
3–4	0.85	0	0.91	0.83	0.57	0.89	0.59	0.95	0.99	0.38
3–6	0.88	0	0.90	0.87	0.61	0.90	0.59	0.95	0.99	0.66
3–8	0.87	0	0.91	0.85	0.60	0.91	0.59	0.94	0.99	0.64
4–2	0.83	0	0.89	0.80	0.55	0.90	0.97	0.89	0.99	0
4–4	0.87	0	0.90	0.80	0.60	0.90	0.98	0.94	0.99	0
4–6	0.89	0.19	0.91	0.83	0.56	0.90	0.99	0.94	0.99	0.21
4–8	0.83	0	0.90	0.78	0	0.87	0.98	0.95	0.80	0
5–2	0	0.36	0.90	0.74	0.52	0.88	0.56	0.83	0.99	0
5–4	0.91	0.38	0.90	0.76	0.62	0.90	0.92	0	0.75	0
5–6	0.87	0.36	0.90	0.79	0.54	0.87	0.88	0	0.99	0
5–8	0.86	0	0.89	0.80	0	0.86	0.97	0.85	0.99	0
6–2	0.79	0	0.83	0.71	0	0.82	0.81	0	0.99	0
6–4	0.84	0	0.86	0.73	0	0.88	0.88	0	0.77	0
6–6	0	0	0.84	0.59	0	0.86	0.83	0	0.75	0
6–8	0	0	0.86	0	0	0.85	0.89	0	0.73	0
L-A	clf	compound	conj	cop	dep	det	discourse:sp	flat	flat:name	mark
2–2	0.87	0.86	0.99	0.88	0.64	0.97	0.22	0.96	0.88	0.99
2–4	0.82	0.86	0.99	0.95	0.63	0.97	0.22	0.99	0.88	0.99
2–6	0.89	0.87	0.99	0.97	0.66	0.97	0.29	0.96	0.88	0.98
2–8	0.83	0.87	0.99	0.98	0.62	0.97	0.33	0.99	0.88	0.99
3–2	0.88	0.87	0.99	0.94	0.64	0.97	0.22	0.96	0.92	0.90
3–4	0.86	0.85	0.99	0.95	0.64	0.97	0.20	0.96	0.94	0.96
3–6	0.88	0.86	0.99	0.97	0.66	0.97	0.21	0.96	0.94	0.96
3–8	0.90	0.87	0.99	0.97	0.66	0.97	0.22	0.92	0.97	0.96
4–2	0.68	0.82	0.97	0.91	0.64	0.95	0.18	0.96	0	0.95
4–4	0.66	0.82	0.99	0.97	0.65	0.95	0.22	0.99	0	0.98
4–6	0.69	0.84	0.99	0.97	0.68	0.97	0.29	0.99	0	0.92
4–8	0	0.78	0	0.92	0.64	0.85	0	0.76	0	0.96
5–2	0	0.83	0.99	0.91	0.64	0.84	0.33	0.99	0	0.93
5–4	0	0.81	0	0.97	0	0.84	0.29	0.99	0.80	0.88
5–6	0	0.82	0.99	0.95	0	0.85	0	0.99	0	0.91
5–8	0	0.83	0.86	0.97	0	0.85	0.22	0.81	0.84	0.84
6–2	0	0.83	0.53	0.92	0	0.85	0	0.96	0	0.82
6–4	0	0.76	0	0.94	0	0.83	0	0.73	0	0.87
6–6	0	0.66	0	0.91	0	0.82	0	0.88	0	0.82
6–8	0	0.62	0	0.92	0	0.83	0	0.81	0.72	0.84
L-A	mark:prt	mark:relcl	nmod	nsubj	nummod	obj	obl	obl:tmod	punct	root
2–2	0.68	0.96	0.92	0.64	0.97	0.53	0.79	0.40	0.99	0.98
2–4	0.66	0.97	0.93	0.66	0.98	0.58	0.79	0.42	0.99	0.98
2–6	0.71	0.97	0.92	0.68	0.98	0.61	0.77	0.44	0.99	0.98
2–8	0.70	0.97	0.92	0.67	0.98	0.59	0.80	0.41	0.99	0.98
3–2	0.75	0.98	0.92	0.68	0.98	0.63	0.81	0.42	0.99	0.99
3–4	0.73	0.74	0.73	0.66	0.99	0.58	0.84	0.44	0.99	0.98
3–6	0.69	0.77	0.72	0.66	0.99	0.60	0.79	0.42	0.99	0.98
3–8	0.69	0.79	0.71	0.68	0.99	0.63	0.84	0.53	0.99	0.99
4–2	0	0.97	0.92	0.64	0.97	0.55	0.80	0.34	0.99	0.99
4–4	0	0.96	0.94	0.69	0.99	0.62	0.82	0.37	0.99	0.98
4–6	0.72	0.97	0.92	0.67	0.99	0.60	0.82	0.44	0.99	0.99
4–8	0	0.97	0.90	0.62	0.98	0.44	0.78	0.34	0.98	0.98
5–2	0	0.62	0.72	0.65	0.98	0.56	0	0.36	0.99	0.98
5–4	0	0.97	0.92	0.66	0.86	0.60	0.77	0	0.99	0.99
5–6	0	0.97	0.91	0.65	0.85	0.58	0.73	0.37	0.99	0.98
5–8	0	0.97	0.92	0.56	0.83	0.52	0.73	0	0.99	0.89
6–2	0	0.97	0.89	0.42	0.83	0	0	0	0.98	0
6–4	0	0.97	0.90	0.50	0.86	0	0.64	0	0.98	0.82
6–6	0	0.88	0.68	0.47	0	0	0.66	0	0.96	0.88
6–8	0	0.72	0.80	0.51	0	0	0.66	0	0.99	0.79

Table B.1: GAT predictions of syntactic dependency in Chinese via a different number of attention heads and layer pairs.

	Zh
L-A	xcomp
2–2	0.48
2–4	0.54
2–6	0.56
2–8	0.58
3–2	0.63
3–4	0.53
3–6	0.65
3–8	0.68
4–2	0.47
4–4	0.44
4–6	0.56
4–8	0.47
5–2	0.41
5–4	0.53
5–6	0.48
5–8	0
6–2	0
6–4	0
6–6	0
6–8	0

Table B.2: GAT predictions of syntactic dependency in Chinese via a different number of attention heads and layer pairs.

					Ru					
L-A	acl	acl:relcl	advcl	advmod	amod	appos	aux	aux:pass	case	cc
2–2	0.54	0	0	0.90	0.98	0.32	0.75	0.96	0.99	0.97
2–4	0.52	0	0.71	0.91	0.98	0.55	0.89	0.96	0.99	0.99
2–6	0.64	0.81	0	0.89	0.98	0.24	0	0	0.98	0.96
2–8	0.64	0	0	0.90	0.98	0.50	0.67	0.92	0.98	0.97
3–2	0.57	0	0	0.90	0.98	0.12	0	0	0.98	0.96
3–4	0.63	0	0.56	0.92	0.98	0.45	0	0	0.98	0.96
3–6	0.63	0.84	0	0.90	0.98	0.48	0	0	0.98	0.96
3–8	0.67	0.72	0	0.91	0.98	0.13	0	0	0.99	0.96
4–2	0.51	0	0	0.92	0.97	0	0	0	0.97	0.84
4–4	0.60	0.64	0	0.89	0.97	0	0.67	0	0.99	0.82
4–6	0.73	0.84	0.39	0.90	0.98	0.65	0	0.86	0.99	0.82
4–8	0.65	0	0	0.92	0.99	0.55	0.44	0	0.99	0.96
5–2	0.57	0	0.23	0.91	0.96	0	0	0	0.97	0.85
5–4	0.67	0.78	0.49	0.91	0.97	0	0	0	0.98	0.82
5–6	0.77	0.75	0.17	0.91	0.97	0.44	0	0	0.97	0.81
5–8	0.56	0	0	0.91	0.96	0.54	0	0.86	0.99	0.86
6–2	0	0	0	0.90	0.96	0	0	0.89	0.94	0.83
6–4	0	0.42	0	0.88	0.88	0	0	0	0.95	0.78
6–6	0.30	0	0	0.88	0.91	0	0	0	0.94	0.79
6–8	0	0	0	0.90	0.96	0	0	0	0.96	0.85
L-A	ccomp	conj	cop	csubj	det	fixed	flat	flat:forign	flat:name	mark
2–2	0.70	0.84	0.96	0	0.99	0.43	0.85	0.87	0.58	0.97
2–4	0.67	0.87	0.99	0	0.99	0.57	0.86	0.92	0.56	0.94
2–6	0.54	0.88	0.58	0	0.98	0.61	0.87	0.80	0.52	0.96
2–8	0.57	0.87	0.96	0	0.99	0.50	0.86	0.87	0.64	0.90
3–2	0.50	0.88	0.56	0	0.98	0	0	0.74	0.51	0.93
3–4	0.81	0.90	0.67	0	0.99	0.67	0.86	0.87	0.55	0.94
3–6	0.67	0.89	0.67	0	0.99	0.56	0.77	0.83	0.59	0.93
3–8	0.63	0.87	0.65	0	0.99	0.67	0.86	0.92	0.61	0.93
4–2	0.60	0	0.63	0	0.99	0	0	0.69	0.52	0.94
4–4	0.31	0	0.73	0	0.99	0.76	0.77	0.83	0.64	0.94
4–6	0	0	0.96	0.13	0.99	0.84	0.67	0.83	0.69	0.97
4–8	0.72	0.88	0.85	0	0.99	0.80	0.80	0.92	0.68	0.94
5–2	0.63	0	0.56	0	0.99	0	0.55	0.88	0.59	0.93
5–4	0.69	0	0.58	0	0.99	0.71	0.77	0.87	0.59	0.96
5–6	0	0	0.61	0	0.99	0	0.67	0.80	0.62	0.93
5–8	0.49	0	0.96	0	0.99	0.80	0.48	0	0.61	0.96
6–2	0.28	0	0.88	0	0	0	0	0.71	0.58	0.91
6–4	0.48	0	0.63	0	0.94	0	0	0.81	0.43	0.97
6–6	0	0	0.58	0	0.93	0	0	0.74	0.43	0.93
6–8	0.49	0	0.56	0	0.99	0	0	0.83	0.55	0.93
L-A	nmod	nsubj	nummod	nummod:gov	obj	obl	punct	root	xcomp	
2–2	0.90	0.71	0.76	0.33	0.58	0.89	0.99	0.98	0.53	
2–4	0.90	0.67	0.75	0.43	0.56	0.91	0.99	0.98	0.53	
2–6	0.88	0.67	0.76	0	0.48	0.90	0.99	0.98	0	
2–8	0.90	0.69	0.75	0	0.54	0.91	0.99	0.98	0	
3–2	0.88	0.67	0.65	0.31	0.55	0.93	0.99	0.98	0	
3–4	0.89	0.69	0.71	0.43	0.59	0.92	0.99	0.99	0.56	
3–6	0.91	0.67	0.73	0.50	0.52	0.92	0.99	0.98	0	
3–8	0.91	0.70	0.71	0.40	0.60	0.93	0.99	0.99	0	
4–2	0.83	0.70	0.70	0.43	0.57	0.90	0.99	0.94	0.45	
4–4	0.86	0.65	0.71	0.43	0.52	0.91	0.99	0	0	
4–6	0.91	0.72	0.75	0.43	0.59	0.92	0.99	0.98	0	
4–8	0.92	0.71	0.77	0.40	0.63	0.93	0.99	0.98	0.61	
5–2	0.87	0.63	0.78	0.53	0.44	0.90	0.99	0	0	
5–4	0.83	0.71	0.72	0.31	0.56	0.90	0.99	0.97	0.52	
5–6	0.87	0.69	0.72	0.31	0.60	0.89	0.99	0	0.52	
5–8	0.89	0.68	0.79	0.43	0.50	0.91	0.99	0.98	0	
6–2	0.78	0.67	0.68	0	0.41	0.88	0.98	0.96	0	
6–4	0	0.64	0.62	0	0.46	0.75	0.99	0.95	0	
6–6	0	0.53	0.54	0	0.40	0.75	0.98	0	0	
6–8	0.83	0.53	0.63	0	0.40	0.88	0.99	0	0	

Table B.3: GAT predictions of syntactic dependency in Russian via a different number of attention heads and layer pairs.

				De					
L-A	acl	acl:relel	advcl	advmod	amod	appos	aux	aux:pass	case
2–2	0	0.71	0.83	0.99	0.95	0.39	0.85	0.81	0.99
2–4	0.5	0.75	0.89	0.99	0.95	0.56	0.91	0.81	0.99
2–6	0.5	0.75	0.89	0.99	0.95	0.56	0.91	0.81	0.99
2–8	0	0.41	0	0.99	0.94	0	0.86	0.81	0.99
3–2	0	0.60	0	0.99	0.94	0	0.85	0.81	0.99
3–4	0	0.45	0	0.99	0.94	0	0.85	0.81	0.99
3–6	0	0.41	0	0.98	0.94	0	0.88	0.81	0.99
3–8	0	0.46	0	0.99	0.94	0	0.88	0.81	0.99
4–2	0	0.52	0	0.99	0.95	0	0.81	0	0.99
4–4	0	0.45	0	0.99	0.94	0	0	0	0.99
4–6	0	0.40	0	0.98	0.93	0	0	0.48	0.99
4–8	0	0.45	0	0.98	0.93	0	0	0.52	0.99
5–2	0	0.42	0	0.99	0.92	0	0.86	0.81	0.99
5–4	0	0.68	0	0.99	0.93	0	0.85	0.81	0.99
5–6	0	0.44	0	0.99	0.94	0	0	0	0.99
5–8	0	0.43	0	0.97	0.94	0	0	0	0.99
6–2	0	0	0	0.98	0.90	0.07	0.62	0	0.98
6–4	0	0	0	0.97	0.91	0	0	0.70	0.98
6–6	0	0	0	0.97	0.91	0	0	0	0.98
6–8	0	0.37	0	0.97	0.91	0	0	0	0.98
L-A	cc	ccomp	compound	compound:prt	conj	cop	det	flat:name	mark
2–2	0.99	0.56	0.80	0	0.78	0.93	0.99	0.83	0.97
2–4	0.99	0.60	0.81	0	0.81	0.98	0.99	0.85	0.97
2–6	0.99	0.60	0.81	0	0.81	0.98	0.99	0.85	0.97
2–8	0.99	0	0.72	0	0.82	0.95	0.99	0.81	0.96
3–2	0.99	0.48	0.83	0	0.78	0.93	0.99	0.82	0.95
3–4	0.99	0	0.80	0	0.80	0.95	0.99	0.84	0.86
3–6	0.99	0	0.78	0	0.80	0.95	0.99	0.81	0.91
3–8	0.99	0	0.72	0	0.80	0.95	0.99	0.84	0.91
4–2	0.99	0	0.86	0	0.76	0.93	0.99	0.90	0.93
4–4	0.99	0	0.82	0	0.79	0.57	0.99	0.82	0.84
4–6	0.99	0	0.76	0	0.79	0.90	0.99	0.85	0.93
4–8	0.99	0	0.80	0	0.80	0.88	0.99	0.84	0.85
5–2	0.99	0	0.82	0	0.82	0.95	0.99	0.83	0.92
5–4	0.99	0.52	0.74	0	0.82	0.95	0.99	0.80	0.94
5–6	0.99	0	0.75	0	0.82	0.65	0.99	0.78	0.85
5–8	0.99	0	0	0	0.79	0.57	0.99	0.78	0.86
6–2	0.98	0	0.65	0.67	0.74	0	0.96	0.84	0.82
6–4	0.99	0	0.69	0	0.78	0.70	0.97	0.83	0.84
6–6	0.99	0	0.63	0.69	0.68	0.54	0.98	0.71	0.81
6–8	0.93	0	0.71	0	0	0.55	0.99	0.73	0.87
L-A	nmod	nmod:poss	nsubj	nummod	obj	obl	obl:tmod	punct	root
2–2	0.82	0.85	0.75	0.84	0.63	0.80	0	0.99	0.96
2–4	0.83	0.88	0.72	0.84	0.63	0.83	0	0.99	0.97
2–6	0.83	0.88	0.72	0.84	0.63	0.83	0	0.99	0.97
2–8	0.76	0.86	0.69	0.84	0.56	0.80	0	0.99	0.94
3–2	0.80	0.85	0.78	0.87	0.67	0.84	0	0.99	0.97
3–4	0.80	0.86	0.71	0.84	0.37	0.84	0	0.99	0.92
3–6	0.79	0.85	0.72	0.87	0.56	0.86	0	0.99	0.93
3–8	0.81	0.83	0.74	0.87	0.59	0.84	0	0.99	0.93
4–2	0.81	0.86	0.74	0.84	0.65	0.85	0	0.99	0.95
4–4	0.78	0.85	0.73	0.87	0.51	0.86	0	0.99	0.93
4–6	0.81	0.82	0.77	0.84	0.65	0.85	0	0.99	0.93
4–8	0.78	0.86	0.74	0.87	0.64	0.86	0	0.99	0.95
5–2	0.81	0.83	0.78	0.90	0.62	0.83	0.44	0.99	0.89
5–4	0.82	0.84	0.79	0.90	0.66	0.87	0.44	0.99	0.96
5–6	0.82	0.85	0.72	0.87	0.56	0.82	0	0.99	0.96
5–8	0.76	0.83	0.73	0.80	0.60	0.85	0	0.97	0.89
6–2	0.73	0.81	0.65	0.67	0.23	0.72	0	0.97	0.89
6–4	0.75	0.85	0.65	0.76	0.23	0.87	0	0.97	0.79
6–6	0.81	0.85	0.67	0.81	0.22	0.85	0	0.98	0.90
6–8	0.66	0	0.63	0.81	0	0.86	0	0.98	0.89

Table B.4: GAT predictions of syntactic dependency in German via a different number of attention heads and layer pairs.

	De
L-A	xcomp
2–2	0.55
2–4	0.49
2–6	0.49
2–8	0
3–2	0.38
3–4	0
3–6	0
3–8	0
4–2	0.41
4–4	0
4–6	0
4–8	0
5–2	0
5–4	0
5–6	0
5–8	0
6–2	0
6–4	0
6–6	0
6–8	0

Table B.5: GAT predictions of syntactic dependency in German via a different number of attention heads and layer pairs.

Appendix C

Syntax via Graph with BERT

C.1 Syntactic Predictions in GAT

The results of the syntactic prediction task on the GAT are depicted in Table C.1 to Table C.3. To ensure the accuracy of the experiment, some dependency relations that are insufficient in the number have been removed.

C.2 Representational Similarity Analysis

The RSA tests for the dependency relations in the Baseline, SGBC, and SGBD engines of BERT for various source languages are displayed in Table C.4 to Table C.9 across 12 layers (L).

	Zh			
	GAT Prediction	Baseline	SGBC	SGBD
acl:relcl	0.917	0.435	0.515	0.505
advcl	0.421	0.430	0.512	0.518
advmod	0.909	0.433	0.512	0.522
amod	0.927	0.435	0.528	0.520
appos	0.405	0.404	0.482	0.518
aux	0.882	0.421	0.514	0.532
aux:pass	0	0.436	0.477	0.525
case	0.934	0.428	0.511	0.526
case:loc	0.782	0.429	0.523	0.531
cc	0.984	0.436	0.513	0.512
ccomp	0.337	0.441	0.513	0.524
clf	0.745	0.437	0.527	0.533
compound	0.886	0.427	0.512	0.524
conj	0.987	0.435	0.521	0.518
cop	0.945	0.426	0.520	0.511
dep	0.611	0.429	0.514	0.513
det	0.935	0.438	0.530	0.528
flat	0.877	0.387	0.494	0.473
flat:name	0.886	0.415	0.518	0.506
iobj	0	0.387	0.494	0.473
mark	0.986	0.424	0.510	0.529
mark:adv	0.387	0.365	0.427	0.386
mark:prt	0.229	0.435	0.532	0.517
mark:relcl	0.741	0.431	0.518	0.513
nmod	0.933	0.429	0.509	0.523
nsubj	0.623	0.426	0.510	0.523
nsubj:pass	0	0.423	0.512	0.545
nummod	0.933	0.429	0.514	0.522
obj	0.587	0.428	0.514	0.522
obl	0.834	0.432	0.511	0.534
obl:agent	0	0.379	0.576	0.597
obl:patient	0	0.365	0.460	0.434
obl:tmod	0.123	0.417	0.509	0.495
root	0.955	0.426	0.514	0.523
xcomp	0.423	0.438	0.522	0.528

Table C.1: The prediction scores of GAT for dependency relations in Chinese source language sentences and the translation quality changes from different MT engines for these sentences.

	Ru			
	GAT Prediction	Baseline	SGBC	SGBD
acl	0.396	0.708	0.758	0.749
acl:relcl	0.359	0.706	0.756	0.747
advcl	0.337	0.695	0.739	0.752
advmod	0.934	0.704	0.750	0.747
amod	0.982	0.707	0.753	0.752
appos	0.440	0.695	0.742	0.740
aux	0.804	0.700	0.764	0.777
aux:pass	0.974	0.718	0.749	0.760
case	0.978	0.702	0.748	0.748
cc	0.930	0.698	0.751	0.748
ccomp	0.576	0.681	0.745	0.747
compound	0	0.758	0.802	0.811
conj	0.832	0.699	0.749	0.748
cop	0.971	0.720	0.774	0.871
det	0.990	0.697	0.747	0.746
fixed	0.552	0.688	0.742	0.750
flat	0.490	0.696	0.730	0.738
flat:foreign	0.921	0.678	0.701	0.727
flat:name	0.823	0.703	0.761	0.751
iobj	0	0.700	0.746	0.746
mark	0.858	0.703	0.745	0.750
nmod	0.882	0.705	0.750	0.751
nsubj	0.637	0.701	0.749	0.748
nsubj:pass	0	0.708	0.750	0.754
nummod	0.654	0.707	0.748	0.762
obj	0.512	0.705	0.755	0.756
obl	0.900	0.701	0.749	0.749
obl:agent	0	0.716	0.748	0.734
parataxis	0.240	0.693	0.725	0.724
root	0.987	0.700	0.748	0.750
xcomp	0.623	0.712	0.760	0.757

Table C.2: The prediction scores of GAT for dependency relations in Russian source language sentences and the translation quality changes from different MT engines for these sentences.

	De			
	GAT Prediction	Baseline	SGBC	SGBD
acl:relcl	0.612	0.506	0.578	0.582
advcl	0.467	0.514	0.570	0.556
advmod	0.932	0.506	0.573	0.582
amod	0.913	0.507	0.567	0.571
appos	0.523	0.500	0.556	0.565
aux	0.868	0.520	0.586	0.597
aux:pass	0.927	0.498	0.576	0.556
case	0.992	0.504	0.568	0.574
cc	0.987	0.509	0.565	0.561
ccomp	0.186	0.514	0.575	0.579
compound	0.870	0.495	0.577	0.565
conj	0.651	0.510	0.565	0.561
cop	0.743	0.502	0.577	0.586
det	0.987	0.504	0.565	0.571
flat	0.223	0.442	0.553	0.625
flat:name	0.876	0.505	0.551	0.565
iobj	0	0.546	0.590	0.589
mark	0.981	0.511	0.561	0.570
nmod	0.719	0.517	0.570	0.574
nsubj	0.719	0.504	0.571	0.574
nsubj:pass	0	0.504	0.580	0.575
nummod	0.802	0.507	0.581	0.562
obj	0.452	0.506	0.576	0.577
obl	0.830	0.502	0.544	0.574
obl:tmod	0.287	0.501	0.531	0.557
parataxis	0	0.512	0.573	0.546
root	0.931	0.503	0.570	0.574
xcomp	0.224	0.513	0.565	0.553

Table C.3: The prediction scores of GAT for dependency relations in German source language sentences and the translation quality changes from different MT engines for these sentences.

Baseline vs SGBC												
Zh Relations	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12
acl:relcl	0.891	0.733	0.877	0.239	0.452	0.656	0.506	0.712	0.587	0.623	0.442	0.424
advcl	0.875	0.734	0.203	0.479	0.378	0.685	0.462	0.693	0.664	0.668	0.517	0.522
advmod	0.856	0.794	0.878	0.292	0.528	0.781	0.576	0.733	0.638	0.697	0.514	0.512
amod	0.818	0.705	0.962	0.632	0.483	0.662	0.379	0.580	0.398	0.587	0.341	0.335
appos	0.908	0.770	0.901	0.411	0.429	0.694	0.519	0.653	0.599	0.677	0.483	0.485
aux	0.873	0.803	0.954	0.449	0.600	0.760	0.551	0.718	0.614	0.683	0.476	0.441
aux:pass	0.872	0.637	0.972	0.666	0.663	0.504	0.468	0.672	0.540	0.748	0.394	0.300
case	0.880	0.743	0.893	0.576	0.529	0.677	0.514	0.699	0.588	0.649	0.550	0.599
case:loc	0.898	0.744	0.216	0.322	0.477	0.752	0.553	0.762	0.684	0.669	0.509	0.587
cc	0.915	0.782	0.498	0.274	0.442	0.702	0.620	0.660	0.667	0.710	0.588	0.557
ccomp	0.847	0.767	0.808	0.403	0.442	0.783	0.572	0.757	0.684	0.752	0.503	0.570
clf	0.857	0.753	0.840	0.219	0.560	0.673	0.543	0.698	0.606	0.662	0.420	0.501
compound	0.877	0.748	0.871	0.402	0.483	0.727	0.545	0.692	0.615	0.650	0.506	0.684
conj	0.910	0.770	0.479	0.396	0.380	0.706	0.604	0.651	0.664	0.701	0.571	0.566
cop	0.898	0.785	0.480	0.238	0.484	0.743	0.578	0.722	0.720	0.738	0.634	0.613
csubj	0.889	0.895	0.283	0.467	0.623	0.751	0.563	0.761	0.814	0.799	0.557	0.567
dep	0.868	0.798	0.599	0.386	0.584	0.777	0.552	0.703	0.708	0.751	0.447	0.428
det	0.860	0.753	0.937	0.386	0.414	0.721	0.535	0.707	0.573	0.677	0.572	0.511
discourse:sp	0.898	0.810	0.961	0.855	0.784	0.804	0.635	0.802	0.638	0.747	0.627	0.615
flat	0.884	0.858	0.277	0.220	0.408	0.776	0.364	0.607	0.511	0.731	0.542	0.644
flat:name	0.868	0.769	0.330	0.285	0.579	0.594	0.644	0.689	0.594	0.643	0.374	0.409
iobj	0.674	0.478	0.427	0.798	0.382	0.679	0.635	0.701	0.719	0.414	0.289	0.391
mark	0.880	0.705	0.596	0.478	0.418	0.749	0.598	0.722	0.682	0.683	0.467	0.432
mark:adv	0.992	0.936	0.961	0.993	0.698	0.999	0.993	0.984	0.973	0.833	0.999	0.994
mark:prt	0.847	0.741	0.249	0.639	0.354	0.703	0.560	0.697	0.601	0.697	0.644	0.727
mark:relcl	0.889	0.771	0.859	0.545	0.418	0.674	0.484	0.686	0.607	0.655	0.484	0.494
nmod	0.882	0.751	0.870	0.584	0.566	0.668	0.485	0.675	0.579	0.620	0.569	0.593
nsubj	0.863	0.788	0.874	0.437	0.555	0.751	0.538	0.725	0.619	0.691	0.532	0.515
nsubj:pass	0.869	0.729	0.979	0.664	0.690	0.480	0.649	0.728	0.589	0.754	0.531	0.505
nummod	0.870	0.785	0.380	0.274	0.560	0.691	0.519	0.696	0.649	0.697	0.459	0.512
obj	0.873	0.792	0.881	0.469	0.507	0.720	0.577	0.713	0.639	0.683	0.507	0.493
obl	0.881	0.747	0.898	0.491	0.514	0.670	0.498	0.698	0.619	0.602	0.514	0.504
obl:agent	0.956	0.922	0.675	0.753	0.633	0.782	0.900	0.904	0.812	0.764	0.657	0.456
obl:patient	0.840	0.767	0.688	0.580	0.770	0.633	0.737	0.730	0.408	0.560	0.416	0.559
obl:tmod	0.867	0.763	0.391	0.200	0.357	0.817	0.587	0.739	0.697	0.697	0.294	0.403
xcomp	0.831	0.790	0.776	0.519	0.474	0.769	0.682	0.769	0.564	0.400	0.577	0.322
root	0.863	0.791	0.893	0.216	0.541	0.757	0.561	0.741	0.638	0.704	0.503	0.494

Table C.4: When tested on Chinese sentences with target dependency relations, the representation of each layer from BERT in the Baseline and SGBC engine are compared via RSA.

	Baseline vs SGBD											
Zh Relations	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12
acl:relcl	0.902	0.726	0.267	0.237	0.231	0.574	0.339	0.554	0.477	0.554	0.425	0.444
advcl	0.893	0.747	0.278	0.425	0.251	0.554	0.425	0.522	0.454	0.507	0.386	0.411
advmod	0.874	0.768	0.262	0.260	0.401	0.664	0.409	0.492	0.493	0.581	0.448	0.548
amod	0.813	0.688	0.569	0.476	0.411	0.498	0.217	0.364	0.289	0.411	0.260	0.416
appos	0.905	0.779	0.463	0.454	0.357	0.657	0.432	0.455	0.457	0.610	0.466	0.449
aux	0.884	0.770	0.369	0.291	0.400	0.622	0.443	0.512	0.483	0.559	0.458	0.489
aux:pass	0.915	0.692	0.463	0.591	0.728	0.656	0.473	0.355	0.360	0.676	0.386	0.531
case	0.884	0.736	0.678	0.456	0.272	0.573	0.363	0.444	0.435	0.526	0.424	0.492
case:loc	0.909	0.771	0.372	0.297	0.299	0.627	0.391	0.491	0.477	0.489	0.379	0.475
cc	0.885	0.780	0.607	0.362	0.354	0.540	0.360	0.410	0.535	0.660	0.496	0.448
ccomp	0.886	0.725	0.355	0.249	0.400	0.666	0.381	0.459	0.400	0.482	0.401	0.449
clf	0.881	0.725	0.635	0.421	0.378	0.597	0.392	0.500	0.490	0.540	0.371	0.425
compound	0.888	0.750	0.484	0.398	0.308	0.639	0.388	0.447	0.438	0.550	0.443	0.434
conj	0.887	0.777	0.599	0.340	0.452	0.552	0.346	0.405	0.515	0.654	0.494	0.555
cop	0.894	0.772	0.431	0.434	0.272	0.638	0.455	0.524	0.510	0.498	0.480	0.393
csubj	0.913	0.820	0.748	0.591	0.483	0.831	0.347	0.655	0.563	0.643	0.608	0.689
dep	0.881	0.819	0.523	0.491	0.420	0.627	0.436	0.470	0.513	0.566	0.395	0.419
det	0.855	0.713	0.269	0.217	0.285	0.581	0.355	0.517	0.507	0.578	0.384	0.406
discourse:sp	0.922	0.747	0.234	0.603	0.614	0.705	0.409	0.577	0.640	0.760	0.578	0.434
flat	0.891	0.857	0.342	0.445	0.257	0.585	0.342	0.457	0.400	0.682	0.442	0.486
flat:name	0.897	0.776	0.282	0.419	0.274	0.481	0.385	0.362	0.395	0.482	0.309	0.455
iobj	0.699	0.917	0.556	0.470	0.357	0.669	0.695	0.560	0.598	0.467	0.386	0.558
mark	0.901	0.723	0.407	0.408	0.434	0.641	0.684	0.469	0.452	0.428	0.482	0.417
mark:adv	0.970	0.994	0.883	0.992	0.975	0.999	0.993	0.988	0.657	0.716	0.984	0.958
mark:prt	0.883	0.800	0.759	0.527	0.240	0.584	0.346	0.544	0.451	0.482	0.377	0.446
mark:relcl	0.892	0.754	0.459	0.226	0.239	0.575	0.352	0.520	0.478	0.551	0.452	0.520
nmod	0.874	0.737	0.552	0.424	0.298	0.595	0.353	0.422	0.439	0.510	0.395	0.495
nsubj	0.879	0.777	0.508	0.427	0.436	0.662	0.412	0.501	0.492	0.560	0.462	0.554
nsubj:pass	0.909	0.755	0.508	0.601	0.765	0.553	0.552	0.504	0.488	0.678	0.389	0.524
nummod	0.886	0.790	0.237	0.371	0.384	0.606	0.375	0.467	0.490	0.575	0.434	0.533
obj	0.880	0.779	0.424	0.272	0.388	0.626	0.413	0.496	0.509	0.554	0.451	0.435
obl	0.907	0.717	0.585	0.430	0.218	0.575	0.366	0.503	0.515	0.570	0.480	0.430
obl:agent	0.953	0.864	0.920	0.860	0.374	0.635	0.496	0.706	0.687	0.768	0.653	0.639
obl:patient	0.822	0.789	0.654	0.720	0.604	0.673	0.502	0.540	0.345	0.586	0.480	0.530
obl:tmod	0.872	0.781	0.442	0.229	0.375	0.589	0.377	0.536	0.571	0.647	0.544	0.605
xcomp	0.900	0.747	0.220	0.330	0.347	0.692	0.468	0.497	0.505	0.576	0.465	0.433
root	0.878	0.781	0.413	0.390	0.431	0.669	0.433	0.525	0.511	0.583	0.480	0.460

Table C.5: When tested on Chinese sentences with target dependency relations, the representation of each layer from BERT in the Baseline and SGBD engine are compared via RSA.

Baseline vs SGBC												
Ru Relations	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12
acl	0.824	0.424	0.392	0.625	0.555	0.738	0.646	0.618	0.571	0.644	0.641	0.559
acl:relcl	0.617	0.309	0.310	0.454	0.412	0.640	0.519	0.635	0.576	0.553	0.506	0.475
advcl	0.710	0.613	0.556	0.609	0.409	0.631	0.623	0.734	0.756	0.748	0.685	0.587
advmod	0.877	0.608	0.428	0.651	0.618	0.764	0.711	0.723	0.721	0.746	0.734	0.618
amod	0.855	0.572	0.444	0.635	0.576	0.731	0.668	0.694	0.693	0.731	0.722	0.597
appos	0.679	0.617	0.286	0.700	0.606	0.707	0.591	0.700	0.769	0.774	0.787	0.569
aux	0.627	0.590	0.504	0.445	0.556	0.527	0.303	0.690	0.768	0.571	0.431	0.396
aux:pass	0.699	0.528	0.357	0.706	0.644	0.730	0.586	0.632	0.605	0.691	0.742	0.560
case	0.856	0.574	0.572	0.462	0.591	0.756	0.694	0.725	0.721	0.740	0.733	0.624
cc	0.872	0.679	0.365	0.654	0.584	0.740	0.726	0.731	0.746	0.766	0.743	0.594
ccomp	0.600	0.566	0.320	0.568	0.561	0.714	0.716	0.806	0.835	0.792	0.778	0.700
compound	0.636	0.587	0.603	0.477	0.474	0.996	0.975	0.988	0.940	0.614	0.942	0.994
conj	0.821	0.663	0.355	0.641	0.595	0.744	0.738	0.739	0.751	0.753	0.743	0.585
cop	0.803	0.548	0.317	0.629	0.547	0.797	0.593	0.633	0.723	0.757	0.768	0.612
csubj	0.525	0.463	0.480	0.368	0.426	0.432	0.517	0.750	0.707	0.621	0.475	0.332
det	0.851	0.670	0.626	0.426	0.537	0.721	0.642	0.678	0.707	0.744	0.713	0.607
fixed	0.759	0.579	0.578	0.633	0.641	0.659	0.615	0.689	0.685	0.699	0.671	0.578
flat	0.665	0.404	0.514	0.572	0.565	0.608	0.484	0.666	0.677	0.627	0.593	0.424
flat:foreign	0.704	0.435	0.548	0.588	0.604	0.704	0.554	0.729	0.758	0.700	0.604	0.419
flat:name	0.703	0.533	0.442	0.596	0.636	0.748	0.629	0.658	0.639	0.599	0.596	0.555
iobj	0.629	0.474	0.553	0.685	0.606	0.659	0.603	0.719	0.697	0.655	0.673	0.556
mark	0.668	0.528	0.231	0.500	0.516	0.629	0.603	0.699	0.723	0.691	0.642	0.498
nmod	0.860	0.478	0.453	0.648	0.544	0.740	0.658	0.696	0.699	0.730	0.726	0.610
nsubj	0.820	0.584	0.466	0.687	0.567	0.732	0.685	0.718	0.719	0.738	0.729	0.596
nsubj:pass	0.711	0.580	0.336	0.723	0.561	0.711	0.575	0.614	0.618	0.708	0.732	0.610
nummod	0.575	0.624	0.270	0.515	0.610	0.689	0.526	0.669	0.618	0.591	0.562	0.445
nummod:gov	0.640	0.401	0.443	0.579	0.759	0.783	0.531	0.612	0.589	0.644	0.640	0.543
obj	0.756	0.542	0.483	0.661	0.506	0.691	0.641	0.683	0.645	0.675	0.674	0.535
obl	0.764	0.592	0.479	0.657	0.568	0.746	0.684	0.711	0.702	0.709	0.704	0.591
obl:agent	0.638	0.394	0.509	0.825	0.837	0.891	0.851	0.340	0.582	0.717	0.770	0.607
orphan	0.733	0.661	0.241	0.620	0.937	0.800	0.519	0.330	0.651	0.424	0.558	0.638
parataxis	0.825	0.629	0.391	0.598	0.659	0.786	0.714	0.723	0.683	0.670	0.621	0.680
xcomp	0.756	0.658	0.486	0.683	0.575	0.762	0.712	0.731	0.748	0.761	0.754	0.620
root	0.855	0.587	0.466	0.704	0.597	0.751	0.701	0.729	0.722	0.744	0.739	0.623

Table C.6: When tested on Russian sentences with target dependency relations, the representation of each layer from BERT in the Baseline and SGBC engine are compared via RSA.

	Baseline vs SGBD											
Ru Relations	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12
acl	0.918	0.416	0.296	0.617	0.501	0.541	0.611	0.562	0.573	0.752	0.779	0.627
acl:relcl	0.505	0.299	0.292	0.484	0.402	0.474	0.606	0.643	0.628	0.739	0.744	0.651
advcl	0.585	0.541	0.489	0.508	0.505	0.562	0.665	0.676	0.692	0.747	0.804	0.686
advmod	0.931	0.442	0.509	0.608	0.613	0.646	0.731	0.720	0.710	0.830	0.846	0.666
amod	0.910	0.548	0.488	0.391	0.573	0.605	0.679	0.683	0.674	0.796	0.777	0.594
appos	0.574	0.331	0.303	0.614	0.432	0.517	0.618	0.715	0.740	0.787	0.731	0.581
aux	0.344	0.563	0.494	0.457	0.433	0.288	0.321	0.208	0.321	0.496	0.458	0.401
aux:pass	0.491	0.385	0.327	0.537	0.633	0.528	0.618	0.708	0.723	0.779	0.669	0.588
case	0.903	0.502	0.504	0.413	0.602	0.634	0.721	0.722	0.721	0.808	0.808	0.639
cc	0.943	0.392	0.417	0.624	0.590	0.626	0.705	0.719	0.724	0.822	0.826	0.646
ccomp	0.517	0.432	0.341	0.521	0.540	0.615	0.722	0.741	0.763	0.864	0.885	0.667
compound	0.699	0.777	0.474	0.902	0.365	0.902	0.991	0.764	0.996	0.988	0.954	0.955
conj	0.887	0.442	0.452	0.600	0.402	0.594	0.687	0.698	0.707	0.799	0.797	0.634
cop	0.651	0.415	0.545	0.536	0.586	0.722	0.729	0.668	0.761	0.833	0.758	0.583
csubj	0.450	0.488	0.473	0.417	0.496	0.229	0.480	0.603	0.676	0.544	0.468	0.393
det	0.895	0.446	0.408	0.675	0.616	0.673	0.759	0.755	0.774	0.848	0.854	0.742
fixed	0.666	0.415	0.516	0.673	0.605	0.599	0.698	0.644	0.683	0.800	0.748	0.603
flat	0.643	0.511	0.452	0.519	0.430	0.512	0.627	0.690	0.711	0.749	0.764	0.620
flat:foreign	0.638	0.520	0.387	0.542	0.523	0.545	0.621	0.683	0.728	0.772	0.786	0.677
flat:name	0.657	0.357	0.472	0.587	0.546	0.531	0.647	0.664	0.678	0.786	0.772	0.641
iobj	0.519	0.287	0.599	0.663	0.552	0.563	0.675	0.690	0.671	0.787	0.821	0.699
mark	0.537	0.367	0.274	0.288	0.515	0.591	0.711	0.714	0.724	0.817	0.842	0.704
nmod	0.911	0.379	0.462	0.596	0.573	0.611	0.686	0.682	0.677	0.787	0.771	0.594
nsubj	0.884	0.528	0.508	0.623	0.576	0.621	0.706	0.720	0.711	0.803	0.785	0.598
nsubj:pass	0.504	0.314	0.292	0.538	0.585	0.574	0.634	0.667	0.695	0.791	0.703	0.551
nummod	0.467	0.588	0.389	0.525	0.426	0.460	0.555	0.648	0.647	0.786	0.827	0.703
nummod:gov	0.570	0.536	0.331	0.686	0.523	0.595	0.682	0.689	0.726	0.825	0.815	0.639
obj	0.826	0.578	0.487	0.598	0.508	0.609	0.703	0.717	0.717	0.793	0.775	0.613
obl	0.797	0.520	0.507	0.619	0.572	0.618	0.715	0.720	0.721	0.780	0.756	0.579
obl:agent	0.806	0.454	0.250	0.742	0.744	0.607	0.472	0.633	0.640	0.694	0.479	0.299
orphan	0.301	0.240	0.524	0.420	0.750	0.709	0.579	0.427	0.419	0.322	0.228	0.243
parataxis	0.935	0.444	0.472	0.657	0.574	0.618	0.704	0.733	0.711	0.828	0.833	0.643
xcomp	0.611	0.587	0.593	0.565	0.569	0.665	0.729	0.765	0.754	0.830	0.808	0.648
root	0.901	0.506	0.504	0.637	0.612	0.649	0.720	0.724	0.716	0.806	0.787	0.612

Table C.7: When tested on Russian sentences with target dependency relations, the representation of each layer from BERT in the Baseline and SGBD engine are compared via RSA.

Baseline vs SGBC												
De Relations	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12
acl:relcl	0.696	0.776	0.763	0.690	0.601	0.604	0.670	0.627	0.621	0.629	0.613	0.629
advcl	0.640	0.776	0.781	0.716	0.645	0.506	0.632	0.602	0.572	0.514	0.527	0.575
advmod	0.775	0.819	0.841	0.800	0.737	0.733	0.750	0.793	0.747	0.790	0.750	0.748
amod	0.651	0.739	0.774	0.721	0.631	0.662	0.708	0.645	0.670	0.641	0.644	0.663
appos	0.695	0.766	0.814	0.751	0.682	0.664	0.702	0.667	0.671	0.678	0.680	0.674
aux	0.649	0.796	0.795	0.716	0.657	0.649	0.638	0.669	0.690	0.700	0.670	0.648
aux:pass	0.644	0.735	0.766	0.723	0.627	0.721	0.684	0.661	0.700	0.641	0.629	0.661
case	0.734	0.773	0.781	0.747	0.686	0.694	0.708	0.691	0.689	0.699	0.765	0.716
cc	0.613	0.721	0.719	0.675	0.602	0.591	0.631	0.592	0.606	0.595	0.595	0.598
ccomp	0.686	0.768	0.824	0.767	0.757	0.695	0.661	0.698	0.702	0.706	0.729	0.664
compound	0.687	0.780	0.785	0.733	0.661	0.649	0.721	0.700	0.688	0.653	0.654	0.691
compound:prt	0.671	0.760	0.763	0.662	0.703	0.694	0.730	0.680	0.717	0.735	0.681	0.790
conj	0.586	0.716	0.712	0.661	0.588	0.583	0.620	0.588	0.588	0.592	0.595	0.611
cop	0.679	0.794	0.808	0.772	0.649	0.690	0.753	0.735	0.730	0.670	0.695	0.726
csubj	0.686	0.730	0.860	0.809	0.770	0.853	0.798	0.660	0.824	0.860	0.714	0.737
cc:preconj	0.633	0.443	0.411	0.823	0.647	0.557	0.563	0.471	0.424	0.471	0.462	0.415
csubj:pass	0.868	0.742	0.886	0.904	0.492	0.937	0.977	0.731	0.760	0.806	0.785	0.638
det	0.628	0.757	0.773	0.724	0.654	0.694	0.702	0.584	0.597	0.596	0.587	0.597
expl	0.568	0.803	0.658	0.669	0.607	0.438	0.653	0.442	0.566	0.600	0.452	0.443
flat	0.609	0.770	0.921	0.721	0.761	0.554	0.923	0.455	0.577	0.520	0.786	0.649
flat:name	0.686	0.719	0.729	0.698	0.678	0.633	0.706	0.677	0.662	0.641	0.649	0.672
iobj	0.692	0.826	0.792	0.706	0.681	0.784	0.735	0.692	0.698	0.728	0.781	0.803
mark	0.693	0.787	0.799	0.752	0.701	0.676	0.684	0.696	0.708	0.681	0.682	0.693
nmod	0.725	0.767	0.776	0.750	0.677	0.711	0.695	0.586	0.649	0.649	0.617	0.657
nmod:poss	0.694	0.758	0.758	0.731	0.667	0.719	0.681	0.689	0.671	0.694	0.671	0.681
nsubj	0.655	0.794	0.806	0.768	0.695	0.705	0.725	0.610	0.780	0.788	0.793	0.760
nsubj:pass	0.694	0.758	0.758	0.731	0.667	0.719	0.681	0.689	0.671	0.694	0.671	0.681
nummod	0.716	0.858	0.839	0.728	0.714	0.705	0.730	0.777	0.790	0.714	0.741	0.729
obj	0.625	0.773	0.785	0.729	0.654	0.672	0.682	0.528	0.534	0.646	0.640	0.671
obl	0.659	0.767	0.776	0.753	0.684	0.685	0.703	0.656	0.678	0.663	0.667	0.706
obl:tmod	0.683	0.741	0.791	0.716	0.660	0.740	0.696	0.696	0.732	0.686	0.681	0.815
parataxis	0.652	0.798	0.792	0.756	0.775	0.674	0.645	0.658	0.700	0.667	0.674	0.689
xcomp	0.841	0.884	0.885	0.806	0.802	0.822	0.818	0.852	0.884	0.863	0.816	0.827
root	0.782	0.843	0.841	0.834	0.765	0.726	0.736	0.758	0.783	0.763	0.754	0.739

Table C.8: When tested on German sentences with target dependency relations, the representation of each layer from BERT in the Baseline and SGBC engine are compared via RSA.

Baseline vs SGBD												
De Relations	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12
acl:relcl	0.793	0.740	0.860	0.831	0.845	0.883	0.850	0.828	0.863	0.801	0.778	0.689
advcl	0.773	0.720	0.843	0.815	0.820	0.894	0.867	0.856	0.894	0.842	0.840	0.747
advmod	0.782	0.796	0.849	0.832	0.856	0.859	0.827	0.774	0.787	0.783	0.785	0.794
amod	0.773	0.732	0.802	0.816	0.844	0.808	0.801	0.800	0.812	0.768	0.766	0.780
appos	0.762	0.778	0.806	0.830	0.855	0.729	0.812	0.820	0.817	0.767	0.735	0.788
aux	0.747	0.735	0.833	0.810	0.836	0.796	0.717	0.777	0.781	0.742	0.746	0.734
aux:pass	0.766	0.728	0.799	0.839	0.867	0.825	0.825	0.806	0.815	0.746	0.798	0.748
case	0.774	0.759	0.819	0.812	0.849	0.830	0.825	0.820	0.826	0.790	0.797	0.797
cc	0.777	0.780	0.764	0.789	0.816	0.741	0.775	0.766	0.779	0.759	0.749	0.742
ccomp	0.792	0.794	0.822	0.831	0.877	0.841	0.829	0.829	0.818	0.775	0.788	0.798
compound	0.790	0.788	0.845	0.847	0.849	0.797	0.778	0.789	0.790	0.798	0.790	0.780
compound:prt	0.795	0.795	0.808	0.791	0.827	0.811	0.831	0.850	0.879	0.865	0.835	0.804
conj	0.797	0.787	0.795	0.784	0.814	0.773	0.784	0.778	0.787	0.786	0.780	0.783
cop	0.792	0.779	0.839	0.831	0.874	0.855	0.840	0.830	0.839	0.801	0.797	0.790
csubj	0.679	0.767	0.939	0.901	0.922	0.651	0.668	0.664	0.710	0.792	0.733	0.692
cc:preconj	0.634	0.557	0.642	0.684	0.818	0.459	0.411	0.595	0.678	0.673	0.644	0.520
csubj:pass	0.843	0.805	0.799	0.770	0.786	0.850	0.897	0.839	0.773	0.774	0.781	0.800
det	0.872	0.889	0.837	0.819	0.836	0.817	0.851	0.849	0.831	0.820	0.866	0.827
expl	0.753	0.770	0.719	0.850	0.884	0.840	0.822	0.829	0.860	0.843	0.824	0.786
flat	0.679	0.610	0.913	0.958	0.933	0.956	0.977	0.958	0.953	0.835	0.747	0.779
flat:name	0.682	0.643	0.817	0.831	0.869	0.833	0.829	0.833	0.832	0.811	0.777	0.655
iobj	0.769	0.797	0.791	0.746	0.793	0.832	0.889	0.871	0.881	0.869	0.843	0.789
mark	0.804	0.812	0.798	0.804	0.848	0.796	0.813	0.814	0.802	0.802	0.799	0.801
nmod	0.759	0.716	0.834	0.825	0.835	0.824	0.814	0.744	0.735	0.762	0.748	0.744
nmod:poss	0.796	0.795	0.793	0.809	0.841	0.768	0.815	0.786	0.785	0.792	0.782	0.797
nsubj	0.794	0.795	0.835	0.820	0.854	0.851	0.834	0.717	0.735	0.731	0.771	0.723
nsubj:pass	0.888	0.875	0.821	0.853	0.878	0.829	0.828	0.808	0.819	0.855	0.819	0.882
nummod	0.844	0.879	0.847	0.842	0.841	0.856	0.854	0.854	0.859	0.892	0.871	0.849
obj	0.775	0.784	0.801	0.799	0.824	0.812	0.797	0.732	0.793	0.760	0.791	0.799
obl	0.787	0.793	0.814	0.812	0.850	0.828	0.820	0.814	0.818	0.780	0.746	0.782
obl:tmod	0.794	0.805	0.829	0.816	0.870	0.805	0.752	0.815	0.849	0.844	0.858	0.851
parataxis	0.792	0.792	0.811	0.877	0.866	0.776	0.726	0.729	0.754	0.753	0.739	0.767
xcomp	0.877	0.889	0.861	0.847	0.868	0.858	0.858	0.855	0.856	0.888	0.893	0.875
root	0.797	0.795	0.828	0.819	0.854	0.846	0.829	0.717	0.791	0.728	0.772	0.743

Table C.9: When tested on German sentences with target dependency relations, the representation of each layer from BERT in the Baseline and SGBD engine are compared via RSA.