Thèse n° 9804

# Modeling Structured Data in Attention-based Models

Présentée le 13 octobre 2023

Faculté informatique et communications
Laboratoire de systèmes d'information répartis
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

**Alireza MOHAMMADSHAHI**

Acceptée sur proposition du jury

École
polytechnique
fédérale
de Lausanne

2023

# Abstract

Natural language processing (NLP) has experienced significant improvements with the development of Transformer-based (Vaswani et al., 2017) models e.g. BERT (Devlin et al., 2019) and GPT (Brown et al., 2020), which employ self-attention mechanism and pre-training strategies. Although Transformer-based models have achieved significant accomplishments, they still present several obstacles. A notable issue is their inability to encode structured data, e.g. graphs, which is crucial for tasks involving structured knowledge processing, including syntactic and semantic parsing, and information extraction. Additionally, the considerable size of pre-trained Transformer-based models poses challenges for real-world deployment, prompting a growing interest in applying compression techniques e.g. knowledge distillation, pruning, and quantisation. Understanding the impact of compression requires examining several aspects including attention patterns in compressed models for different languages, gender and semantic biases. In this thesis, we present an approach for extending Transformer-based models to encode graphs in the attention mechanism and examine the different behaviours of these models (including the attention patterns) following compression.

Our first contribution is to propose Graph-to-Graph Transformer (G2GTr) architecture, which modifies the self-attention mechanism of Transformer for conditioning on graph structures in addition to the input sequences. This mechanism incorporates graph relations as continuous embeddings rather than previous works that use a discrete model structure or pre-defined discrete attention heads. An explicit representation of relations is supported by inputting these embeddings into the self-attention mechanism, which is applied to every pair of tokens. In this way, each attention head can easily learn to attend only to tokens in a given relation, while also having the ability to learn additional structures in conjunction with other tokens. We then apply the G2GTr model to encode the partially constructed graph in the transition-based dependency parsing task.

Our second contribution is to propose Recursive Non-autoregressive Graph-to-Graph Transformer (RNGTr), a graph prediction model that exploits the complete graph-to-graph capabilities of G2GTr model to recursively refine the output predicted graph. This model, despite predicting all graph edges simultaneously and being non-autoregressive, is capable of capturing any between-edge dependencies by conditioning on the prior predicted graph, similar to an auto-regressive model. This proposed architecture comprises three modules. Firstly, an initialization parser is used to generate an initial graph structure, which could be any model suited to the task, even a trivial one. Secondly, a G2GTr model is used to encode the previously predicted graph and computes the output distribution over the target graph. Thirdly, a decoding algorithm is utilized to identify the optimal graph based on these edge scores.

## Abstract

Our third contribution is to propose Syntax-aware Graph-to-Graph Transformer (SynG2G-Tr) architecture to encode the syntactic knowledge in the semantic role labelling task. It conditions on the sentence's dependency structure and predicts both span-based and dependency-based SRL graphs at the same time. The modified self-attention function models the interaction of the graph relations with both the query and key vectors of the self-attention mechanism, not just the query. We also discover that omitting the interaction of the graph structure with the value vectors of self-attention does not negatively impact the performance. We show empirically that our model surpasses the performance of prior comparable models.

Our fourth contribution is to demonstrate the effects of compression methods on multilingual Transformer-based NMT models, that have been pre-trained in a great number of languages in different domains. We analyse different aspects of compressed models, including attention patterns, gender and semantic biases. In this study, we concentrate on *light* compression techniques, specifically post-training quantisation and magnitude pruning without any further fine-tuning.[1] We exploit the recent and largest MNMT model, M2M-100 (Fan et al., 2020a), that encompasses 100 languages and contains nearly 12 billion parameters. We evaluate the impact of compression on different language pairs using the FLORES-101 (Goyal et al., 2021b) benchmark, which covers 101 languages. We also consider MT-Gender (Stanovsky et al., 2019) and DiBiMT (Campolungo et al., 2022) benchmarks allowing us to assess different types of biases that could be present in the data and MNMT model.

Our fifth contribution is to propose SMaLL-100, a **S**hallow **M**ultilingual **Ma**chine Translation Model for **L**ow-Resource **L**anguages covering 100 languages, which is a distilled version of M2M-100 (12B) (Fan et al., 2020a), the most recent and largest available multilingual NMT model. We particularly focus on low and very low-resource language pairs, given the absence of a reasonably-sized model that delivers satisfactory performance across a substantial number of low-resource languages. We achieve this by training our model on a balanced dataset, where each language direction has the same sampling probability, irrespective of their training data size. While this leads to lower performance on the high-resource languages, we claim that this loss is easily recoverable through further fine-tuning. We evaluate SMaLL-100 on various low-resource benchmarks, such as FLORES-101 (Goyal et al., 2021b), Tatoeba (Tiedemann, 2020), and TICO-19 (Anastasopoulos et al., 2020).

**Keywords:** generalization, compression, graphs, parsing, semantic role labelling, machine translation, multilinguality, fairness, low-resource.

[1]The reason is that fine-tuning MNMT models is extremely computationally demanding.

# Zusammenfassung

Die Verarbeitung natürlicher Sprache hat mit der Entwicklung Transformer-basierter (Vaswani et al., 2017) Modelle, z. B. BERT (Devlin et al., 2019) und GPT (Brown et al., 2020), die Selbstaufmerksamkeitsmechanismen und Pre-Training-Strategien nutzen. Obwohl Transformer-basierte Modelle erhebliche Erfolge erzielt haben, stellen sie immer noch einige Hindernisse dar. Ein bemerkenswertes Problem ist ihre Unfähigkeit, strukturierte Daten zu kodieren, z. Diagramme, was für Aufgaben mit strukturierter Wissensverarbeitung, einschließlich syntaktischer und semantischer Analyse sowie Informationsextraktion, von entscheidender Bedeutung ist. Darüber hinaus stellt die beträchtliche Größe vorab trainierter Transformer-basierter Modelle Herausforderungen für den Einsatz in der Praxis dar und führt zu einem wachsenden Interesse an der Anwendung von Komprimierungstechniken, z. B. Wissensdestillation, Beschneidung und Quantisierung. Um die Auswirkungen der Komprimierung zu verstehen, müssen mehrere Aspekte untersucht werden, darunter Aufmerksamkeitsmuster in komprimierten Modellen für verschiedene Sprachen, Geschlecht und semantische Verzerrungen. In dieser Arbeit stellen wir einen Ansatz zur Erweiterung transformatorbasierter Modelle vor, um Graphen im Aufmerksamkeitsmechanismus zu kodieren, und untersuchen das unterschiedliche Verhalten dieser Modelle (einschließlich der Aufmerksamkeitsmuster) nach der Komprimierung.

Unser erster Beitrag besteht darin, eine Graph-to-Graph Transformer (G2GTr)-Architektur vorzuschlagen, die den Selbstaufmerksamkeitsmechanismus von Transformer für die Konditionierung auf Graphstrukturen zusätzlich zu den Eingabesequenzen modifiziert. Dieser Mechanismus umfasst Graphbeziehungen als kontinuierliche Einbettungen und nicht frühere Arbeiten, die eine diskrete Modellstruktur oder vordefinierte diskrete Aufmerksamkeitsköpfe verwenden. Eine explizite Darstellung von Beziehungen wird durch die Eingabe dieser Einbettungen in den Selbstaufmerksamkeitsmechanismus unterstützt, der auf jedes Tokenpaar angewendet wird. Auf diese Weise kann jeder Aufmerksamkeitskopf leicht lernen, sich nur auf Token in einer bestimmten Beziehung zu konzentrieren, während er gleichzeitig die Fähigkeit hat, zusätzliche Strukturen in Verbindung mit anderen Token zu lernen. Anschließend wenden wir das G2GTr-Modell an, um den teilweise erstellten Graphen in der übergangsbasierten Abhängigkeitsanalyseaufgabe zu kodieren.

Unser zweiter Beitrag besteht darin, den rekursiven nicht autoregressiven Graph-to-Graph-Transformer (RNGTr) vorzuschlagen, ein Graph-Vorhersagemodell, das die vollständigen Graph-to-Graph-Fähigkeiten des G2GTr-Modells nutzt, um den ausgegebenen vorhergesagten Graphen rekursiv zu verfeinern. Obwohl dieses Modell alle Diagrammkanten gleichzeitig vorhersagt und nicht autoregressiv ist, ist es in der Lage, alle Abhängigkeiten zwischen Kanten zu erfassen, indem

es auf das zuvor vorhergesagte Diagramm konditioniert wird, ähnlich einem autoregressiven Modell. Diese vorgeschlagene Architektur umfasst drei Module. Zunächst wird ein Initialisierungsparser verwendet, um eine anfängliche Diagrammstruktur zu generieren, bei der es sich um jedes für die Aufgabe geeignete Modell handeln kann, auch um ein triviales. Zweitens wird ein G2GTr-Modell verwendet, um den zuvor vorhergesagten Graphen zu kodieren und die Ausgabeverteilung über den Zielgraphen zu berechnen. Drittens wird ein Dekodierungsalgorithmus verwendet, um anhand dieser Kantenwerte den optimalen Graphen zu identifizieren.
Unser dritter Beitrag besteht darin, eine syntaxbewusste Graph-to-Graph Transformer (SynG2G-Tr)-Architektur vorzuschlagen, um das syntaktische Wissen in der semantischen Rollenbezeichnungsaufgabe zu kodieren. Es richtet sich nach der Abhängigkeitsstruktur des Satzes und sagt gleichzeitig sowohl spannenbasierte als auch abhängigkeitsbasierte SRL-Diagramme voraus. Die modifizierte Selbstaufmerksamkeitsfunktion modelliert die Interaktion der Graphbeziehungen sowohl mit der Abfrage als auch mit Schlüsselvektoren des Selbstaufmerksamkeitsmechanismus, nicht nur mit der Abfrage. Wir entdecken auch, dass das Weglassen der Interaktion der Diagrammstruktur mit den Wertvektoren der Selbstaufmerksamkeit keinen negativen Einfluss auf die Leistung hat. Wir zeigen empirisch, dass unser Modell die Leistung früherer vergleichbarer Modelle übertrifft.
Unser vierter Beitrag besteht darin, die Auswirkungen von Komprimierungsmethoden auf mehrsprachige Transformer-basierte NMT-Modelle zu demonstrieren, die in einer großen Anzahl von Sprachen in verschiedenen Domänen vorab trainiert wurden. Wir analysieren verschiedene Aspekte komprimierter Modelle, einschließlich Aufmerksamkeitsmuster, Geschlecht und semantische Verzerrungen. In dieser Studie konzentrieren wir uns auf *leichte* Komprimierungstechniken, insbesondere auf Post-Training-Quantisierung und Größenbereinigung ohne weitere Feinabstimmung.[2] Wir nutzen das jüngste und größte MNMT-Modell, M2M-100 (Fan et al., 2020a), das 100 Sprachen umfasst und fast 12 Milliarden Parameter enthält. Wir bewerten die Auswirkung der Komprimierung auf verschiedene Sprachpaare mithilfe des FLORES-101 (Goyal et al., 2021b)-Benchmarks, der 101 Sprachen abdeckt. Wir berücksichtigen auch die Benchmarks MT-Gender (Stanovsky et al., 2019) und DiBiMT (Campolungo et al., 2022), die es uns ermöglichen, verschiedene Arten von Verzerrungen zu bewerten, die in den Daten und im MNMT-Modell vorhanden sein könnten.
Unser fünfter Beitrag besteht darin, SMaLL-100 vorzuschlagen, ein **S**heiliges **M**mehrsprachiges **Ma**chinesisches Übersetzungsmodell für **L**ow-Resource **L**Sprachen, das 100 Sprachen abdeckt , eine destillierte Version von M2M-100 (12B) (Fan et al., 2020a), dem neuesten und größten verfügbaren mehrsprachigen NMT-Modell. Wir konzentrieren uns insbesondere auf Sprachpaare mit geringen und sehr geringen Ressourcen, da es kein Modell mit angemessener Größe gibt, das eine zufriedenstellende Leistung für eine beträchtliche Anzahl von Sprachen mit geringen Ressourcen liefert. Dies erreichen wir, indem wir unser Modell auf einem ausgewogenen Datensatz trainieren, bei dem jede Sprachrichtung unabhängig von der Größe ihrer Trainingsdaten die gleiche Stichprobenwahrscheinlichkeit aufweist. Dies führt zwar zu einer geringeren Leistung bei ressourcenintensiven Sprachen, wir behaupten jedoch, dass dieser Verlust durch weitere Feinabstimmung leicht ausgeglichen werden kann. Wir bewerten SMaLL-100 anhand verschiedener

[2]Der Grund dafür ist, dass die Feinabstimmung von MNMT-Modellen äußerst rechenintensiv ist.

ressourcenarmer Benchmarks, wie FLORES-101 (Goyal et al., 2021b), Tatoeba (Tiedemann, 2020) und TICO-19 (Anastasopoulos et al., 2020).

**Schlüsselwörter:** Generalisierung, Komprimierung, Diagramme, Parsing, semantische Rollenbezeichnung, maschinelle Übersetzung, Mehrsprachigkeit, Fairness, geringe Ressourcen.

# Acknowledgements

This thesis is the result of invaluable guidance, support, and mentorship from several people. First and foremost, I would like to extend my deepest gratitude to my supervisors, Dr. James Henderson and Prof. Karl Aberer. Their expertise, dedication, and leadership have been instrumental in shaping my skills and helping me excel in my study. I have been fortunate to work under their guidance and have benefited tremendously from their vast knowledge and experience in the field. Especially, I thank Dr. James Henderson, whose insightful feedback and constructive criticism has been instrumental in refining my work and pushing me to reach new heights. His exceptional leadership and vision have inspired me to set ambitious goals and pursue them with dedication and enthusiasm. I am truly grateful for the opportunities he has provided me to learn and grow professionally. I would like to thank the jury members of my thesis: Prof. Alexandre Alahi, Prof. Antoine Bosselut, Prof. Noah A. Smith, and Prof. Ivan Titov, for reviewing and joining my oral exam. I would also like to thank Swiss National Science Foundation for funding my research through the INTREPID project.

My gratitude also goes to my hosting team at the Meta AI and Meta AI Research, especially my wonderful mentors, Marzieh Saeidi, Majid Yazdani, Pouya Yanki, and Angela Fan for providing me opportunities to gain hands-on experience in various aspects of the organization. It was an invaluable experience for me to wotk with these awesome researchers.

I would like to express my appreciation to the team at NAVERLABS Europe who hosted me during my internship. I am particularly grateful to my managers, Vassilina Nikoulina, Prof. Laurent Besacier, and Alexandre Berard for their exceptional guidance and support. They provided me with valuable support and help during my project with them. Working alongside these exceptional researchers was a truly priceless experience for me.

Also, I want to express my heartfelt thanks to my research group at IDIAP and EPFL. Their unwavering commitment to excellence, collaborative spirit, and support have made my research journey immensely rewarding and fulfilling. I especially thank members of NLU group at IDIAP, and Remi Lebret at EPFL for their wonderful support during my study.

Last but not the least, I am deeply grateful to my family members for their invaluable support throughout my PhD path. Their constant encouragement, love, and belief in my abilities have been instrumental in my success.

# Publications Based on the Thesis

Chapter 2 is based on:

- "Graph-to-Graph Transformer for Transition-based Dependency Parsing", Alireza Mohammadshahi & James Henderson, *Findings of EMNLP 2020* (Mohammadshahi and Henderson, 2020).

Chapter 3 is based on:

- "Recursive Non-Autoregressive Graph-to-Graph Transformer for Dependency Parsing with Iterative Refinement", Alireza Mohammadshahi & James Henderson, *Transaction of ACL 2021* (Mohammadshahi and Henderson, 2021a).

Chapter 4 is based on:

- "Syntax-Aware Graph-to-Graph Transformer for Semantic Role Labelling", Alireza Mohammadshahi & James Henderson, *Rep4NLP of ACL 2023* (Mohammadshahi and Henderson, 2021b).

Chapter 5 is based on:

- "What Do Compressed Multilingual Machine Translation Models Forget?", Alireza Mohammadshahi & Vassilina Nikoulina & Alexandre Berard & Caroline Brun & James Henderson & Laurent Besacier, *Findings of EMNLP 2022* (Mohammadshahi et al., 2022b).

Chapter 6 is based on:

- "SMaLL-100: Introducing Shallow Multilingual Machine Translation Model for Low-Resource Languages", Alireza Mohammadshahi & Vassilina Nikoulina & Alexandre Berard & Caroline Brun & James Henderson & Laurent Besacier, *EMNLP 2022* (Mohammadshahi et al., 2022a).

## 0.1 Other Publications

During my PhD studies, I also had the chance to make contributions to the papers listed below:

- "RQUGE: Reference-Free Metric for Evaluating Question Generation by Answering the Question", Alireza Mohammadshahi & Thomas Scialom & Majid Yazdani & Pouya Yanki & Angela Fan & James Henderson & Marzieh Saeidi, *ACL 2023* (Mohammadshahi et al., 2022c).

## Publications Based on the Thesis

- "Multilingual Extraction and Categorization of Lexical Collocations with Graph-aware Transformers", Luis Espinosa Anke & Alexander Shvets & Alireza Mohammadshahi & James Henderson & Leo Wanner, **SEM 2022* (Espinosa Anke et al., 2022).
- "Aligning Multilingual Word Embeddings for Cross-Modal Retrieval Task", Alireza Mohammadshahi & Rémi Lebret & Karl Aberer, *FEVER at EMNLP 2019* (Mohammadshahi et al., 2019).
- "The DCU-EPFL Enhanced Dependency Parser at the IWPT 2021 Shared Task", James Barry & Alireza Mohammadshahi & Joachim Wagner & Jennifer Foster & James Henderson, *IWPT at ACL 2021* (Barry et al., 2021).

# Contents

# List of Figures

# List of Tables

# 1 Introduction & Background

Natural language processing (NLP) refers to the branch of artificial intelligence, concerned with giving computers the ability to understand text and spoken words in much the same way human beings can. In the early days of NLP, researchers focused on understanding the fundamental elements of natural languages, such as syntax (Henderson, 2003; Chomsky, 2002; Henderson, 1990), semantics (Gesmundo et al., 2009; Carnap, 1947; Katz and Fodor, 1963), and pragmatics. This led to the development of rule-based systems (Allen and Perrault, 1980; Wilks, 1975; Woods, 1970), where linguistic rules were manually crafted to facilitate language processing. However, these systems were limited in their ability to scale and adapt to new linguistic phenomena, paving the way for the introduction of statistical methods and machine learning techniques.

The introduction of word embeddings, such as Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), revolutionized NLP by enabling the representation of words as continuous vectors in a high-dimensional space. These embeddings capture semantic and syntactic relationships between words, allowing for more accurate language modelling and improved performance across a wide range of NLP tasks. These representations, however, still fell short in capturing context-dependent meanings and long-range dependencies between words in a sentence.
The emergence of Transformer architectures (Vaswani et al., 2017), such as the GPT (Generative Pre-trained Transformer) (Brown et al., 2020) and BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019), addressed these limitations by employing self-attention mechanisms and pre-training strategies. The self-attention mechanism allows Transformer to effectively model long-range dependencies and capture contextual information, while pre-training strategies, such as masked language modelling and next-sentence prediction, enable the models to learn powerful contextual representations from vast amounts of unlabelled text data. These advancements have led to significant improvements in NLP performance and have set new benchmarks across various tasks, including generation (e.g. machine translation and question answering) and classification (e.g. sentiment analysis and natural language inference). In recent years, increasing the size of these models has become crucial for achieving top-notch performance in natural language processing (OpenAI, 2023; Touvron et al., 2023; Scao et al., 2023; Zhang et al., 2022), as it has been observed that certain abilities only manifest when models surpass a

specific scale (Wei et al., 2022a). This is particularly evident in the domain of Neural Machine Translation (NMT), where large-scale multilingual NMT models (NLLB et al., 2022; Zhang et al., 2020a; Fan et al., 2020a; Tang et al., 2020; Aharoni et al., 2019) have showcased impressive outcomes. These models are especially beneficial for languages with limited resources, as they greatly profit from the transfer of knowledge.

Despite these successes, Transformer-based models also come with their own set of challenges. One prominent issue is their inability to effectively encode structured data (e.g. graphs), which can be critical for tasks that require reasoning over structured knowledge (e.g. syntactic and semantic parsing, and information extraction). Another challenge faced by pre-trained Transformer-based models is their substantial size, which makes them computationally expensive and difficult to deploy in real-world applications with limited resources (Treviso et al., 2023). This has led to a growing interest in model compression techniques, such as knowledge distillation (Li et al., 2021a; Kim and Rush, 2016), pruning (Zhang et al., 2021a; Behnke and Heafield, 2020), and quantisation (Dettmers et al., 2022; Tao et al., 2022; Yao et al., 2022), to reduce the size of these models without sacrificing performance (Kim et al., 2021a; Sun et al., 2020; Jiao et al., 2020; Wang et al., 2020a; Chen et al., 2020; Sanh et al., 2019; Lan et al., 2020; Voita et al., 2019). During the compression process, it is crucial to investigate the attention patterns present in compressed models and analyse them concerning diverse aspects, including gender and semantic biases to reach a compact and fair model with reasonable performance across all languages.

## 1.1 Open Problems

In pursuit of a general Transformer architecture, capable of modelling both sequences and graphs, several challenges must be addressed. This thesis looks at 1) effectively modelling structured data in Transformer-based models. For the efficiency of Transformer-based models, this thesis aims to 2) analyse the biases (e.g. gender and semantic) and attention patterns of compressed models to provide a compact and efficient multilingual model, that covers a great number of languages.

- *Modelling structured data.* There are several graph structures defined in the NLP area e.g. syntactic and semantic graphs (Nivre et al., 2018; Henderson et al., 2013; Pradhan et al., 2012; Hajič et al., 2009; Carreras and Màrquez, 2005), which require a general architecture to model them. Previous work has shown different alternatives to encode graph structures, such as adding a second graph encoder on top of the sequence encoder (Marcheggiani and Titov, 2020; Ji et al., 2019; Marcheggiani and Titov, 2017) or defining pre-defined discrete attention heads in Transformer-based encoders (Strubell et al., 2018). However, these proposals are task-specific and add a hard bias toward graph structure. Proposing a general Transformer-based architecture that can simultaneously encode both sequences and graphs is an open problem and overcoming this challenge has the potential to significantly enhance various natural language processing tasks. In this thesis, we consider three NLP tasks which highly depend on modelling graph structures: 1. *transition-based dependency parsing*, 2. *graph-based dependency parsing*, and 3. *semantic role labelling*.

- *Analysis of biases and attention pattern of compressed generative models.* Efficient inference with very large models has become a crucial problem. This challenge can be overcome through model compression, e.g. knowledge distillation (Wang et al., 2021; Li et al., 2021a; Sanh et al., 2019; Kim and Rush, 2016), pruning (Zhang et al., 2021a; Behnke and Heafield, 2020; Michael H. Zhu, 2018; Frankle and Carbin, 2019), and quantisation (Yao et al., 2022; Yang et al., 2022; Tao et al., 2022; Kim et al., 2021a; Bondarenko et al., 2021; Wu et al., 2020; Xu et al., 2018). These methods can be applied with a little loss in top-line metrics while reducing the memory-footprint and enhancing inference time. However, recent work (Ahia et al., 2021; Xu et al., 2021; Li et al., 2021a; Renduchintala et al., 2021; Hooker et al., 2020) has demonstrated that under-represented features can suffer from a drastic decrease in performance which is not necessarily reflected by global (aggregated) metrics. Particularly in multilingual neural machine translation, the overall metrics are often reported as an average across all the language pairs, where the performance between individual language pairs can vary a lot. Therefore, it is even more critical to understand what would be the exact impact of compression on different aspects of these models including attention patterns in different languages, gender, and semantic biases.

- *Proposing a compact and efficient multilingual model, especially for low-resource languages.* Over the past few years, previous work has proposed several approaches to improve the quality of translations in low-resource languages, e.g., Multilingual Neural Machine Translation (MNMT) models (Goyal et al., 2021b; Tang et al., 2021; Fan et al., 2020a; Johnson et al., 2017), back-translation (Edunov et al., 2018; Sennrich et al., 2016) and unsupervised machine translation (Garcia et al., 2021; Ko et al., 2021). Massively MNMT models are particularly interesting for low-resource languages as they benefit the most from knowledge transfer from related languages (Arivazhagan et al., 2019). However, it is also seen that *curse of multilinguality* hurts the performance of high-resource languages. So, previous work attempted to increase the model size to maintain the translation performance in both high and low-resource languages. This makes the use of these massively MNMT models challenging in real-world resource-constrained environments. Fairly compressing these massively multilingual models by focusing on very-low and low-resource language pairs is a critical issue, as there is no reasonable-size universal model that achieves acceptable performance over a great number of low-resource languages.

## 1.2 Related Work & Background

### 1.2.1 Original Transformer

Transformer (Vaswani et al., 2017) is an encoder-decoder model, as shown in Figure 1.1. A Transformer encoder (left side) computes an output embedding for each token in the input sequence through stacked layers of Transformer block. Each Transformer block consists of a multi-head self-attention and position-wise feed-forward networks. Each attention head takes its

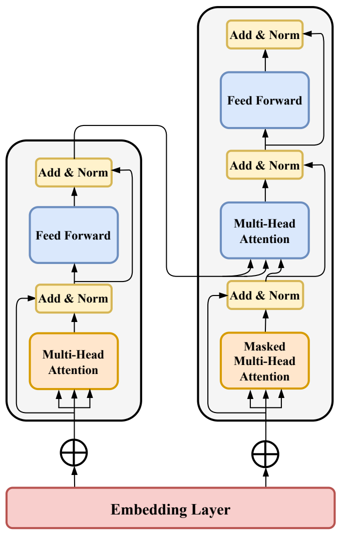


Figure 1.1: Transformer Architecture.

input vectors $(x_1, ..., x_n)$ and computes its output attention vectors $(z_1, ..., z_n)$. Each $z_i \in R^m$ is a weighted sum of transformed input vectors $x_j \in R^m$:

$$z_i = \sum_j \alpha_{ij}(x_j \boldsymbol{W^V}) \tag{1.1}$$

with the attention weights $\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$ and

$$e_{ij} = \frac{(x_i \boldsymbol{W^Q})(x_j \boldsymbol{W^K})}{\sqrt{d}} \tag{1.2}$$

where $\boldsymbol{W^V}, \boldsymbol{W^Q}, \boldsymbol{W^K} \in R^{m \times d}$ are the trained value, query and key matrices, $m$ is the embedding size, and $d$ is the attention head size. The output of multi-head self-attention module is the input of residual connection with layer normalisation (Ba et al., 2016), then it goes through a position-wise feed-forward network. The decoder (right side) is similar to the encoder module. Additionally, the decoder inserts a third sub-layer, which performs multi-head attention over the output vectors of the encoder.

### 1.2.2 Syntactic Parsing

Syntactic parsing is the process of analysing the grammatical structure of a sentence, including identifying the subject, verb, and object. Syntactic dependency parsing is a critical component in a variety of natural language understanding tasks, such as semantic role labelling (Marcheggiani and Titov, 2020, 2017; Henderson et al., 2013), machine translation (Chen et al., 2017), relation extraction (Zhang et al., 2018), and natural language inference (Pang et al., 2019).
It consists of two popular grammar styles; constituency parsing (i.e. phrase-structure parsing) (Titov and Henderson, 2007a; Henderson, 2004, 2003; Manning and Schutze, 1999) and

dependency parsing (Dyer et al., 2015; Nivre and McDonald, 2008; Carreras, 2007; Titov and Henderson, 2007c; Nivre, 2003). The former one analyses the grammar structure of a sentence by identifying the constituents or phrases in the sentence and their hierarchical relationships. The prevalent strategies in constituent parsing include chart-based and transition-based architectures, which are built by statistical and neural network models (Zhou et al., 2020a; Kitaev and Klein, 2018; Titov and Henderson, 2007a; Henderson, 2004; Hall et al., 2014). The later style (i.e. dependency parsing) examines the syntactic dependencies between the words of a sentence to generate a tree-style grammatical structure. There are two main approaches to compute the dependency tree; transition-based and graph-based parsers. Transition-based parsers predict the dependency graph one edge at a time through a sequence of parsing actions (Weiss et al., 2015; Yazdani and Henderson, 2015; Yamada and Matsumoto, 2003; Nivre and Scholz, 2004; Titov and Henderson, 2007c; Zhang and Nivre, 2011), and graph-based parsers (Zhou and Zhao, 2019; Kuncoro et al., 2016; McDonald et al., 2005a; Koo and Collins, 2010) compute scores for every possible dependency edge and then apply a decoding algorithm to find the highest scoring total tree. In the following, we further describe these two approaches.

**Transition-based Parser.** The development of the transition-based approach to dependency parsing was initiated by Yamada and Matsumoto (2003); Nivre (2003), drawing inspiration from history-based parsing (Titov and Henderson, 2007b; Black et al., 1992) and data-driven shift-reduce parsing (Veenstra and Daelemans, 2000). This approach simplifies the complex parsing task by focusing on predicting the next parsing action and implementing parsing as a greedy search for the optimal sequence of actions, making them highly efficient and often having linear time complexity in terms of the length of the sentence.
For instance, arc-standard algorithm (Nivre, 2004) makes parsing decisions in bottom-up order. The main data structures for representing the state of an arc-standard parser are a buffer of words and a stack of partially constructed syntactic sub-trees. At each step, the parser chooses between adding a leftward or rightward labelled arc between the top two words on the stack (`LEFT-ARC(`$l$`)` or `RIGHT-ARC(`$l$`)`, where $l$ is a dependency label) or shifting a word from the buffer onto the stack (`SHIFT`). To handle non-projective dependency trees, Nivre (2009) allow the `SWAP` action, which shifts the second-from-top element of the stack to the front of the buffer, resulting in the reordering of the top two elements of the stack. Modelling the parser state (i.e. partially constructed dependency graph) is the main challenge in the transition-based methods. Previous work addressed this challenge by feature engineering (Ballesteros and Bohnet, 2014; Chen et al., 2014; Zhang and Nivre, 2011), and applying neural network models (Ballesteros et al., 2017; Dyer et al., 2015; Chen and Manning, 2014). Specifically, Dyer et al. (2015) proposed a composition function to model the partially constructed graph by passing the concatenated vectors of head, dependent, and label of constructed relations to a non-linear function (tahn).

**Graph-based Parser.** These models (Zhou and Zhao, 2019; Ballesteros et al., 2016; Wang and Chang, 2016; Kuncoro et al., 2016; McDonald et al., 2005a; Koo and Collins, 2010) generate the dependency graph in a non-autoregressive manner as they calculate scores for all candidate

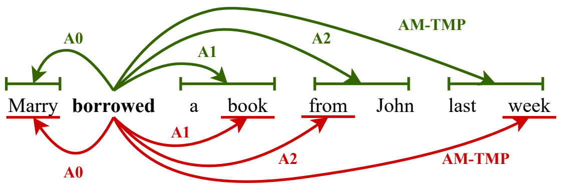


Figure 1.2: Example of SRL graphs. The upper structure is in the span-based style, and the lower one is in the dependency-based style.

dependency relations and then use a decoding method to reach the maximum scoring structure. The primary challenge lies in devising methods to accurately model the interactions of dependency edges without increasing the time complexity. There are several approaches to capture correlations between dependency edges in graph-based models. In first-order models, such as Maximum Spanning Tree (MST) (McDonald et al., 2005b; Edmonds, 1967; i Chu and Liu, 1965), the score for an edge must be computed without being sure what other edges the model will choose. The model itself only imposes the discrete tree constraint between edges. In higher-order models (Tchernowitz et al., 2016; Zhang and McDonald, 2012; Ma and Zhao, 2012; Koo and Collins, 2010; Carreras, 2007; McDonald and Pereira, 2006), they keep some between-edge information, but require more decoding time.

### 1.2.3 Semantic Role Labelling

The semantic role labelling (SRL) task provides a shallow semantic representation of a sentence and builds event properties and relations among relevant words, and is defined in both dependency-based (Surdeanu et al., 2008) and span-based (Pradhan et al., 2012; Carreras and Màrquez, 2005) styles, as shown in Figure 1.2. Semantic role labelling graphs enhance many NLP tasks including question answering (Yih et al., 2016; Shen and Lapata, 2007), machine translation (Kazemi et al., 2017; Wang et al., 2016), and natural language inference (Zhang et al., 2020c).
Traditionally, syntactic structure was regarded as a pre-requisite for SRL models (Henderson et al., 2013; Punyakanok et al., 2008; Gildea and Palmer, 2002), but new models outperform syntax-aware architectures by leveraging deep neural network architectures (Chen et al., 2019; Cai et al., 2018; Tan et al., 2017; He et al., 2017; Marcheggiani et al., 2017) without explicitly encoding syntactic structure. However, recent works (Zhou et al., 2020a; Marcheggiani and Titov, 2020; Strubell et al., 2018; He et al., 2017; Marcheggiani and Titov, 2017) claim that deep neural network models could benefit from using syntactic information, rather than discarding it. Roth and Lapata (2016) embed dependency paths, while some researchers (Fei et al., 2021; Munir et al., 2021; Marcheggiani and Titov, 2017) use graph convolutional networks to encode the syntactic structure. Strubell et al. (2018) incorporates a dependency graph by training one attention head of Transformer to attend to syntactic parents for each token, in a multi-task setting. He et al. (2019, 2018b) use syntactic information to guide the argument pruning. Xia et al. (2019) exploit different alternatives e.g. tree-structured GRU and graph features of dependency tree to encode syntactic knowledge. Kasai et al. (2019) apply BiLSTM to tag the text with supertags extracted from

dependency parses and feed them into SRL models. Xia et al. (2020) showed that incorporating heterogeneous syntactic knowledge results in significant improvement. The question remains open as to the most effective way to incorporate the auxiliary syntactic information into deep learning architectures for SRL.

### 1.2.4 Compression of Pre-trained Multilingual Models

Multilingual NMT provides a single model to translate between any pair of languages, which significantly improves performance on low-resource languages thanks to knowledge transfer (Haddow et al., 2021). Several works (Berard et al., 2021; Fan et al., 2020a; Platanios et al., 2018; Firat et al., 2016; Dong et al., 2015) propose to include both language-specific, and language-independent parameters in MNMT models. Recently, massively MNMT models (Fan et al., 2020a; Zhang et al., 2020a; Arivazhagan et al., 2019; Aharoni et al., 2019; Neubig and Hu, 2018) have been proposed to translate between more than 100 languages. However, these models usually contain a huge number of parameters to maintain performance in both high and low-resource languages, and they demand significant memory and exhibit slow inference speeds. So, compressing these massive multilingual NMT models is a critical problem. Previous research has tackled this issue by employing various compression techniques, such as pruning, quantisation, and knowledge distillation. In the subsequent section, a concise overview of M2M-100 (Fan et al., 2020a), the top-performing and largest multilingual NMT model, along with common compression methods, is presented.

**M2M-100.** is the best performing and the largest massively multilingual MT model, which covers more than 10K language directions, including a great number of low and medium-resource language pairs. M2M-100 is trained on large-scale multilingual corpora (Schwenk et al., 2021; El-Kishky et al., 2020) with a novel data mining procedure, that uses language similarities. The biggest model introduced consists of 24 encoder, and 24 decoder Transformer (Vaswani et al., 2017) layers. Using several scaling techniques, it is trained with nearly 12 billion parameters. We refer to Fan et al. (2020a) for more details.

**Pruning.** is a popular technique for both memory footprint reduction and inference speed-up. It reduces the model size by removing redundant nodes that do not contribute to the resulting performance. It usually achieves comparable results with state-of-the-art models with further fine-tuning (Menghani, 2021; Ahia et al., 2021; Gale et al., 2019; Michael H. Zhu, 2018). In addition, prior research (Wang et al., 2020b; Louizos et al., 2018) has suggested pruning techniques that function as regularisers and can be employed during both pre-training and fine-tuning stages. Some studies have explored pruning specifically during fine-tuning (Sanh et al., 2020; Han et al., 2015) or in a dynamic manner during the inference process (Fan et al., 2020b). Additionally, there are several works on structured pruning, which prunes a larger component of the network. These methods encompass the elimination of attention heads (Voita et al., 2019; Michel et al., 2019), the removal of weak attention values (Qu et al., 2022; Ji et al., 2021), and even the complete deletion of hidden layers (Sajjad et al., 2023; Dong et al., 2017).

**Quantisation.** Mapping high-precision data types to low-precision ones is referred to as quantisation. Numerous studies have focused on specific precision levels, including integers (Kim et al., 2021a), 8-bit representations (Dettmers et al., 2022; Prato et al., 2020; Quinn and Ballesteros, 2018), ternary formats (Zadeh et al., 2022; Zhang et al., 2020b), and even binary representations (Bai et al., 2021). Due to varying sensitivities of sub-networks to quantisation, some works have focused on proposing mixed-precision quantization approaches. (Kim et al., 2021a; Shen et al., 2020). Finally, recent work applies post-training, and training-aware quantisation to pre-trained machine translation and language models (Wei et al., 2022b; Menghani, 2021; Liang et al., 2021; Bondarenko et al., 2021; Wu et al., 2020), and achieves promising results while lowering the inference latency, and the model size.

**Knowledge Distillation.** is a process in which a student model is trained using supervision signals from a teacher model (Hinton et al., 2015). This often results in the student model outperforming a comparably-sized model trained without such supervision. Early research primarily focused on distilling task-specific models (Kim and Rush, 2016), while more recent studies have shifted towards distilling pre-trained models that can be fine-tuned for specific downstream tasks (Gou et al., 2021; Jiao et al., 2020; Sanh et al., 2019). However, the drawbacks of distillation include the increased cost of tuning student hyper-parameters, as well as the potential for diminished performance and generalisation capabilities (Stanton et al., 2021).

After the compression, recent work (Ahia et al., 2021; Xu et al., 2021; Li et al., 2021a; Renduchintala et al., 2021; Hooker et al., 2020) has demonstrated that under-represented features can suffer from a drastic decrease in performance which is not necessarily reflected by global (aggregated) metrics. Specifically in multilingual NMT, the overall metrics are often reported as an average across all the language pairs, where the performance between individual language pairs can vary a lot. Therefore it is even more critical to understand what would be the exact impact of compression on multilingual NMT models, beyond the aggregated metrics. Analysis of the compression effect on different aspects (e.g. language-level performance, gender and semantic biases) could be a starting point to bringing a comprehensive understanding between fairness and compression in multilingual NMT models. This investigation may provide valuable insights into developing fairer compression strategies for massively multilingual models, with a particular focus on devising a fair and reasonable-size multilingual model tailored to low-resource languages that currently lack suitably sized models capable of delivering acceptable performance across a vast number of languages.

## 1.3 Research Questions and Our Contributions

After examining the pertinent background information, we are prepared to explore the specific research questions addressed in this thesis.

**Research Question 1.** *Can a general Transformer-based model be developed that possesses the ability to encode both sequences and graphs effectively?*

In Chapter 2, we address this research question by proposing a version of the Transformer architecture which combines this attention-based mechanism for conditioning on graphs with an attention-like mechanism for predicting graphs and demonstrate its effectiveness on syntactic dependency parsing. We call this architecture Graph-to-Graph Transformer (G2GTr). This mechanism for conditioning on graphs differs from previous proposals in that it inputs graph relations as continuous embeddings, instead of discrete model structure (e.g. (Dyer et al., 2015; Henderson et al., 2013; Henderson, 2003)) or predefined discrete attention heads (e.g. (Ji et al., 2019; Strubell et al., 2018)). An explicit representation of binary relations is supported by inputting these relation embeddings to the attention functions, which are applied to every pair of tokens. In this way, each attention head can easily learn to attend only to tokens in a given relation, but it can also learn other structures in combination with other inputs. This gives a bias towards attention weights which respects locality in the input graph but does not hard-code any specific attention weights. In Chapter 2, we apply our architecture to transition-based dependency parsing, as in auto-regressive structured prediction, after each edge of the graph has been predicted, the model must condition on the partially specified graph to predict the next edge of the graph.

**Research Question 2.** *Can we perform an iterative refinement on the graph-based dependency parsing task without increasing the time complexity?*

In Chapter 3, we propose a new graph prediction architecture which takes advantage of the full graph-to-graph functionality of G2GTr to apply a G2GTr model to refine the output graph recursively. This architecture predicts all edges of the graph in parallel, and is therefore non-autoregressive, but can still capture any between-edge dependency by conditioning on the previous version of the graph, like an auto-regressive model. This proposed Recursive Non-autoregressive Graph-to-Graph Transformer (RNGTr) architecture has three components. First, an initialisation model computes an initial graph, which can be any given model for the task, even a trivial one. Second, a G2GTr model takes the previous graph as input and predicts each edge of the target graph. Third, a decoding algorithm finds the best graph given these edge predictions. The second and third components are applied recursively to do iterative refinement of the output graph until some stopping criterion is met. The final output graph is the graph output by the final decoding step. The RNG Transformer architecture can be applied to any task with a sequence or graph as input and a graph over the same set of nodes as output.

**Research Question 3.** *Is syntax required for semantic role labelling? What is the best method of encoding syntax for the semantic role labelling task?*

In Chapter 4, we propose a novel method for encoding syntactic knowledge by introducing Syntax-aware Graph-to-Graph Transformer (SynG2G-Tr) architecture. The model conditions on the sentence's dependency structure and jointly predicts both span-based and dependency-based SRL structures. Inspired by G2GTr, the model inputs graph relations as embeddings incorporated into the self-attention mechanism of Transformer (Vaswani et al., 2017). The self-attention function models the interaction of the graph relations with both the query and key vectors of self-attention mechanism, instead of just the query. We also find that excluding the interaction of graph structure with the value vectors of self-attention does not harm the performance. Furthermore, the architecture uses different types of graphs as the input and output. We show empirically that our model outperforms previous comparable models.

**Research Question 4.** *What do compressed multilingual machine translation models forget?*

We addressed this research question in Chapter 5 by illustrating the impacts of applying compression methods to massively multilingual NMT models, that are pre-trained in a great number of languages in several domains. To the best of our knowledge, this is the first attempt to analyse how compression impacts massively multilingual models. In this study, we concentrate on *light* compression techniques, specifically post-training quantisation and magnitude pruning without any further fine-tuning.[1] We exploit the recent and largest MNMT model, M2M-100 (Fan et al., 2020a) that covers 100 languages and contains nearly 12B parameters and analyse the impact of compression on different language pairs (based on attention patterns and BLEU performance) evaluated on FLORES-101 benchmark (Goyal et al., 2021b) (covering 101 languages). We also consider MT-Gender (Stanovsky et al., 2019) and DiBiMT (Campolungo et al., 2022) benchmarks allowing us to assess different types of biases that could be present in the data and MNMT model.

**Research Question 5.** *How can we develop a compact massively multilingual model, having a reasonable performance particularly for low-resource languages?*

In Chapter 6, we propose SMaLL-100, a **S**hallow **M**ultilingual M**a**chine Translation Model for **L**ow-Resource **L**anguages covering 100 languages, which is a distilled alternative of M2M-100 (12B) (Fan et al., 2020a), the most recent and biggest available multilingual NMT model. We focus on very-low and low-resource language pairs as there is no reasonable-size universal model that achieves acceptable performance over a great number of low-resource languages. We do so by training SMaLL-100 on a perfectly balanced dataset.[2] While this leads to lower performance on the high-resource languages, we claim that this loss is easily recoverable through further fine-tuning. We evaluate SMaLL-100 on different low-resource benchmarks, e.g., FLORES-

[1]The reason is that fine-tuning MNMT models is extremely computationally demanding.

[2]All language pairs have the same sampling probability, regardless of their training data size.

101 (Goyal et al., 2021b), Tatoeba (Tiedemann, 2020), and TICO-19 (Anastasopoulos et al., 2020).

# 2 Graph-to-Graph Transformer for Transition-based Dependency Parsing

In this chapter, we propose Graph-to-Graph Transformer architecture for conditioning on and predicting arbitrary graphs, and apply it to the challenging task of transition-based dependency parsing. After proposing two novel Transformer models of transition-based dependency parsing as strong baselines, we show that adding the proposed mechanisms for conditioning on and predicting graphs of Graph-to-Graph Transformer results in significant improvements, both with and without BERT pre-training. The novel baselines and their integration with Graph-to-Graph Transformer significantly outperform the state-of-the-art in traditional transition-based dependency parsing on both English Penn Treebank, and 13 languages of Universal Dependencies Treebanks. Graph-to-Graph Transformer can be integrated with many previous structured prediction methods, making it easy to apply to a wide range of NLP tasks.

## 2.1 Introduction

In recent years, there has been a huge amount of research on applying self-attention models to NLP tasks. Transformer (Vaswani et al., 2017) is the most common architecture, which can capture long-range dependencies by using a self-attention mechanism over a set of vectors. To encode the sequential structure of sentences, typically absolute position embeddings are input to each vector in the set, but recently a mechanism has been proposed for inputting relative positions (Shaw et al., 2018). For each pair of vectors, an embedding for their relative position is input to the self-attention function. This mechanism can be generalised to input arbitrary graphs of relations.

In this chapter, we propose a version of the Transformer architecture which combines this attention-based mechanism for conditioning on graphs with an attention-like mechanism for predicting graphs and demonstrate its effectiveness on syntactic dependency parsing. We call this architecture Graph-to-Graph Transformer. This mechanism for conditioning on graphs differs from previous proposals in that it inputs graph relations as continuous embeddings, instead of discrete model structure (e.g. (Dyer et al., 2015; Henderson et al., 2013; Henderson, 2003)) or predefined discrete attention heads (e.g. (Ji et al., 2019; Strubell et al., 2018)). An explicit representation of binary relations is supported by inputting these relation embeddings to the attention functions, which are applied to every pair of tokens. In this way, each attention head can easily learn to attend only to tokens in a given relation, but it can also learn other structures in combination with other inputs. This gives a bias towards attention weights which respects locality in the input graph but does not hard-code any specific attention weights.

We focus our investigation on this novel graph input method and therefore limit our investigation to models which predict the output graph one edge at a time, in an auto-regressive fashion. In auto-regressive structured prediction, after each edge of the graph has been predicted, the model must condition on the partially specified graph to predict the next edge of the graph. Thus, our proposed Graph-to-Graph Transformer parser is a transition-based dependency parser. At each step, the model predicts the next parsing decision, and thereby the next dependency relation, by conditioning on the partial parse structure specified by the previous decisions. It inputs embeddings for the previously specified dependency relations into the Graph-to-Graph Transformer model via the self-attention mechanism. It predicts the next dependency relation using only the vectors for the tokens involved in that relation.

To evaluate this architecture, we also propose two novel Transformer models of transition-based dependency parsing, called Sentence Transformer, and State Transformer. Sentence Transformer computes contextualised embeddings for each token of the input sentence and then uses the current parser state to identify which tokens could be involved in the next valid parse transition and uses their contextualised embeddings to choose the best transition. For State Transformer, we directly use the current parser state as the input to the model, along with an encoding of the partially constructed parse graph, and choose the best transition using the embeddings of the tokens involved in that transition. Both baseline models achieve competitive or better results than previous state-of-the-art traditional transition-based models, but we still get substantial improvement by integrating Graph-to-Graph Transformer with them.

We also demonstrate that, despite the modified input mechanisms, this Graph-to-Graph Transformer architecture can be effectively initialised with standard pre-trained Transformer models. Initialising the Graph-to-Graph Transformer parser with pre-trained BERT (Devlin et al., 2019) parameters leads to substantial improvements. The resulting model significantly improves over the state-of-the-art in traditional transition-based dependency parsing.

This success demonstrates the effectiveness of Graph-to-Graph Transformers for conditioning on and predicting graph relations. This architecture can be easily applied to other NLP tasks that have any graph as the input and need to predict a graph over the same set of nodes as output.

In summary, the contributions of this chapter are:

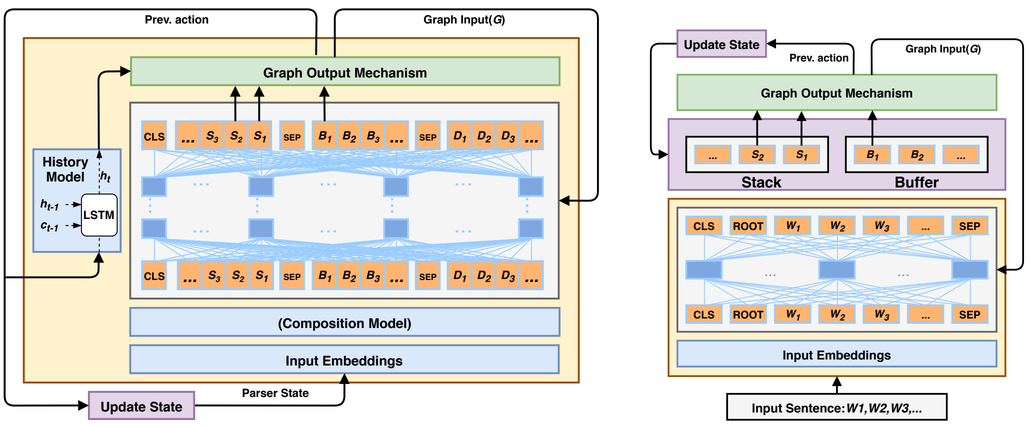
- We propose Graph-to-Graph Transformer for conditioning on and predicting graphs.
- We propose two novel Transformer models of transition-based dependency parsing.
- We successfully integrate pre-trained BERT initialisation in Graph-to-Graph Transformer.

- We improve state-of-the-art accuracies for traditional transition-based dependency parsing.[1]

## 2.2 Transition-based Dependency Parsing

Our transition-based parser uses arc-standard parsing sequences (Nivre, 2004), which makes parsing decisions in bottom-up order. The main data structures for representing the state of an arc-standard parser are a buffer of words and a stack of partially constructed syntactic sub-trees. At each step, the parser chooses between adding a leftward or rightward labelled arc between the top two words on the stack (`LEFT-ARC(`$l$`)` or `RIGHT-ARC(`$l$`)`, where $l$ is a dependency label) or shifting a word from the buffer onto the stack (`SHIFT`). To handle non-projective dependency trees, we allow the `SWAP` action proposed in Nivre (2009), which shifts the second-from-top element of the stack to the front of the buffer, resulting in the reordering of the top two elements of the stack.

## 2.3 Graph-to-Graph Transformer



(a) StateTr+G2GTr. (b) SentTr+G2GTr.

Figure 2.1: The State Transformer and Sentence Transformer parsers with Graph-to-Graph Transformer integrated.

We propose a version of the Transformer which is designed for both conditioning on graphs and predicting graphs, which we call Graph-to-Graph Transformer (G2GTr), and show how it can be applied to transition-based dependency parsing. G2GTr supports arbitrary input graphs and arbitrary edges in the output graph. But since the nodes of both these graphs are the input tokens, the nodes of the output graph are limited to the set of nodes in the input graph.

Inspired by the relative position embeddings of Shaw et al. (2018), we use the attention mechanism of Transformer to input arbitrary graph relations. By inputting the embedding for a relation label into the attention functions for the related tokens, the model can more easily learn to pass

[1]Our implementation is available at: `https://github.com/alirezamshi/G2GTr`

information between graph-local tokens, which gives the model an appropriate linguistic bias, without imposing hard constraints.

Given that the attention function is being used to input graph relations, it is natural to assume that graph relations can also be predicted with an attention-like function. We do not go so far as to restrict the form of the prediction function, but we do restrict the vectors used to predict graph relations to only the tokens involved in the relation.

### 2.3.1 Original Transformer

Transformer (Vaswani et al., 2017) is an encoder-decoder model, of which we only use the encoder component. A Transformer encoder computes an output embedding for each token in the input sequence through stacked layers of multi-head self-attention. Each attention head takes its input vectors $(x_1, ..., x_n)$ and computes its output attention vectors $(z_1, ..., z_n)$. Each $z_i \in R^m$ is a weighted sum of transformed input vectors $x_j \in R^m$:

$$z_i = \sum_j \alpha_{ij}(x_j \boldsymbol{W^V}) \tag{2.1}$$

with the attention weights $\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n}\exp(e_{ik})}$ and

$$e_{ij} = \frac{(x_i \boldsymbol{W^Q})(x_j \boldsymbol{W^K})}{\sqrt{d}} \tag{2.2}$$

where $\boldsymbol{W^V}, \boldsymbol{W^Q}, \boldsymbol{W^K} \in R^{m\times d}$ are the trained value, query and key matrices, $m$ is the embedding size, and $d$ is the attention head size.

### 2.3.2 Graph Inputs

Graph-to-Graph Transformer extends the architecture of the Transformer to accept any arbitrary graph as input. In particular, we input the dependency tree as its set of dependency relations. Each labelled relation $(x_i, x_j, l')$ is input by modifying Equation 2.2 as follows:

$$e_{ij} = \frac{(x_i \boldsymbol{W^Q})(x_j \boldsymbol{W^K} + p_{ij}\boldsymbol{W_1^L})}{\sqrt{d}} \tag{2.3}$$

where $p_{ij} \in \{0,1\}^k$ is a one-hot vector which specifies the type $l'$ of the relation between $x_i$ and $x_j$, discussed below, and $\boldsymbol{W_1^L} \in R^{k\times d}$ is a matrix of learned parameters. We also modify Equation 2.1 to transmit information about relations to the output of the attention layer:

$$z_i = \sum_j \alpha_{ij}(x_j \boldsymbol{W^V} + p_{ij}\boldsymbol{W_2^L}) \tag{2.4}$$

where $\boldsymbol{W_2^L} \in R^{k\times d}$ are learned parameters.

In this chapter, we consider graph input for only unlabelled directed dependency relations $l'$, so

$p_{ij}$ has only three dimensions ($k$=3), for `leftward`, `rightward` and `none`. This choice was made mostly to simplify our extension of the Transformer, as well as to limit the computational cost of this extension. The dependency labels are input as label embeddings added to the input token embeddings of the dependent word.

### 2.3.3 Graph Outputs

The graph output mechanism of Graph-to-Graph Transformer predicts each labelled edge of the graph using the output embeddings of the tokens that are connected by that edge. Because in this work we are investigating auto-regressive models, this prediction is done one edge at a time. Chapter 3 provides an investigation of non-autoregressive models using the G2GTr architecture.

In this chapter, the graph edges are labelled dependency relations, which are predicted as part of the actions of a transition-based dependency parser. In particular, the Relation classifier uses the output embeddings of the top two elements on the stack and predicts the label of their dependency relation, conditioned on its direction. There is also an Exist classifier, which uses the output embeddings of the top two elements on the stack and the front of the buffer to predict the type of parser action, `SHIFT`, `SWAP`, `RIGHT-ARC`, or `LEFT-ARC`.

$$\begin{aligned} a^t &= \text{Exist}([g^t_{s_2}, g^t_{s_1}, g^t_{b_1}]) \\ l^t &= \text{Relation}([g^t_{s_2}, g^t_{s_1}] \mid a^t) \end{aligned} \tag{2.5}$$

where $g^t_{s_2}$, $g^t_{s_1}$, and $g^t_{b_1}$ are the output embeddings of top two tokens in the stack and the front of buffer, respectively. The Exist and Relation classifiers are MLPs with one hidden layer.

For the transition-based dependency parsing task, the chosen parser action and dependency label are used both to update the current partial dependency structure and to update the parser state.

## 2.4 Parsing Models

In this section, we define two Transformer-based models for transition-based dependency parsing, and integrate the Graph-to-Graph Transformer architecture with them, as illustrated in Figure 2.1.

### 2.4.1 State Transformer

We propose a novel attention-based architecture, called State Transformer (StateTr), which computes a comprehensive representation for the parser state. Inspired by Dyer et al. (2015), we directly use the parser state, meaning both the stack and buffer elements, as the input to the Transformer model. We additionally incorporate components that have proved successful in Dyer et al. (2015). In the remaining paragraphs, we describe each component in more detail.

### Input Embeddings

The Transformer architecture takes a sequence of input tokens and converts them into a sequence of input embedding vectors, before computing its context-dependent token embeddings. For the State Transformer model, the sequence of input tokens represents the current parser state, as illustrated in Figure 2.1a.

**Input Sequence.** The input symbols include the words of the sentence $\Omega = (w_1, w_2, ..., w_n)$ with their associated part-of-speech tags (PoS) $(\alpha_1, \alpha_2, ..., \alpha_n)$. Each of these words can appear in the stack or buffer of the parser state. Besides, there is the `ROOT` symbol, for the root of the dependency tree, which is always on the bottom of the stack. Inspired by the input representation of BERT (Devlin et al., 2019), we also use two special symbols, `CLS` and `SEP`, which indicate the different parts of the parser state.

The sequence of input tokens starts with the `CLS` symbol, then includes the tokens on the stack from bottom to top. Then it has a `SEP` symbol, followed by the tokens on the buffer from front to back so that they are in the same order in which they appeared in the sentence. Given this input sequence, the model computes a sequence of vectors which are input to the Transformer network. Each vector is the sum of several embeddings, which are defined below.

**Input Token Embeddings.** The embedding of each token $(w_i)$ is calculated as:

$$T_{w_i} = \text{Emb}(w_i) + \text{Emb}(\alpha_i) \tag{2.6}$$

where $\text{Emb}(w_i), \text{Emb}(\alpha_i) \in R^m$ are the word and PoS embeddings respectively. For the word embeddings, we use the pre-trained word vectors of the BERT model. During training and evaluation, we use the pre-trained embedding of first sub-word as the token representation of each word and discard embeddings of non-first sub-words due to training efficiency.[2] The PoS embeddings are trained parameters.

**Composition Model.** As an alternative to our proposed graph input method, previous work has shown that complex phrases can be input to a neural network by using recursive neural networks to recursively compose the embeddings of sub-phrases (Socher et al., 2011, 2014, 2013; Hermann and Blunsom, 2013; Tai et al., 2015). We extend the proposed composition model of Dyer et al. (2015) by applying a one-layer feed-forward neural network as a composition model and adding skip connections to each recursive step.[3] Since a syntactic head may contain an arbitrary number of dependents, we compute new token embeddings of head-dependent pairs one at a time as

[2]Using embeddings of first sub-word for each word results in better performance than using the last one or averaging all of them as also shown in previous works (Kondratyuk and Straka, 2019; Kitaev et al., 2019).

[3]These skip connections help address the vanishing gradient problem, and preliminary experiments indicated that they were necessary to integrate pre-trained BERT (Devlin et al., 2019) parameters with the model (discussed in Section 2.4.4 and Appendix 2.8.1).

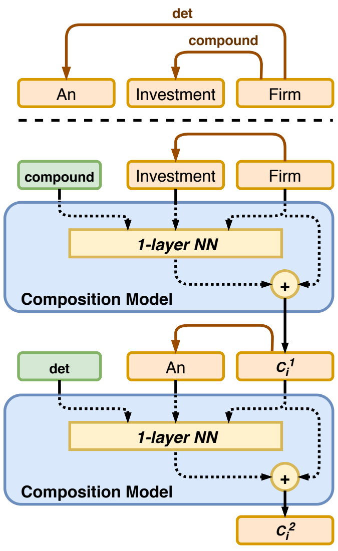


Figure 2.2: An Example of Composition model.

they are specified by the parser, as shown in Figure 2.2. At each parser step $t$, we compute each new token embedding $C_i^t$ of token $i$ by inputting to the composition model, its previous token embedding $C_j^{t-1}$ and the embedding of the most recent dependent with its associated dependency label, where $j$ is the position of token $i$ in the previous parser state. At $t = 0$, $C_i^0$ is set to the initial token embedding $T_{w_i}$. More mathematical and implementation details are given in Appendix 2.9.

**Position and Segment Embeddings.** To distinguish the different positions and roles of words in the parser state, we add their embeddings to the token embeddings. Position embeddings $\beta_i$ encode the token's position in the whole sequence.[4] Segment embeddings $\gamma_i$ encode that the input sequence contains distinct segments (e.g. stack and buffer).

**Total Input Embeddings.** Finally, at each step $t$, we sum the outputs of the composition model with the segment and position embeddings and consider them as the sequence of input embeddings for our State Transformer model.

$$x_i^t = C_i^t + \gamma_i + \beta_i \tag{2.7}$$

### History Model

We define a history model similar to Dyer et al. (2015), to capture the information about previously specified transitions. The output $h^t$ of the history model is computed as follows:

$$h^t,\ c^t = \mathrm{LSTM}((h^{t-1}, c^{t-1}), a^t + l^t) \tag{2.8}$$

[4]Preliminary experiments showed that using position embeddings for the whole sequence achieves better performance than applying separate position embeddings for each segment (More detail in Appendix 2.8.2).

where $a^t$ and $l^t$ are the previous transition and its associated dependency label, and $h^{t-1}$ and $c^{t-1}$ are the previous output vector and cell state of the history model. The output of the history model is input directly to the parser action classifiers in (2.5).

### 2.4.2 Sentence Transformer

We propose another attention-based architecture, called Sentence Transformer (SentTr), to compute a representation for the parser state. This model first uses a Transformer to compute context-dependent embeddings for the tokens in the input sentence. Similarly to Cross and Huang (2016), a separate stack and buffer data structure is used to keep track of the parser state, as shown in Figure 2.1b, and the context-dependent embeddings of the tokens that are involved in the next parser action are used to predict the next transition. More specifically, the input sentence tokens are computed with the BERT tokeniser (Devlin et al., 2019) and the next transition is predicted from the embeddings of the first sub-words of the top two elements of the stack and the front element of the buffer.[5]

In the baseline version of this model, the Transformer which computes the token embeddings does not see the structure of the parser state nor the partial dependency structure.

In Sentence Transformer, the sequence of input tokens starts with a `CLS` token and ends with a `SEP` token, as in the BERT (Devlin et al., 2019) input representation. It also includes the `ROOT` symbol for the root of the dependency tree. The input embeddings are derived from input tokens as:

$$x_i = \text{Emb}(w_i) + \text{Emb}(\alpha_i) + \beta_i \tag{2.9}$$

where $x_i$ is the input embedding for token $w_i$, Emb(.) is defined as in Equation (2.6), and $\beta_i$ is the positional embedding for the element at position $i$.

### 2.4.3 Integrating with G2G Transformer

We use the two proposed attention-based dependency parsers above as baselines, and evaluate the effects of integrating them with the Graph-to-Graph Transformer architecture. We modify the encoder component of each baseline model by adding the graph input mechanism defined in Section 2.3.2. Then, we compute the new partially constructed graph as follows:

$$\begin{aligned} Z^t &= \text{Gin}(X, G^t) \\ G^{t+1} &= G^t \cup \text{Gout}(\text{Select}(Z^t, P^t)) \end{aligned} \tag{2.10}$$

where $G^t$ is the current partially specified graph, $Z^t$ is the encoder's sequence of output token embeddings, $P^t$ is the parser state, and $G^{t+1}$ is the newly predicted partial graph. Gin, and Gout

[5]Predicting transitions with the embedding of first sub-word for each word results in better performance than using the last one or all of them as also shown in previous works (Kondratyuk and Straka, 2019; Kitaev et al., 2019).

are the graph input and graph output mechanisms defined in Sections 2.3.2 and 2.3.3. The Select function selects from $Z^t$, the token embeddings of the top two elements on the stack and the front of the buffer, based on the parser state $P^t$. More specifics about each baseline are given in the following paragraphs.[6]

**State Tr +G2GTr.** To input all the dependency relations in the current partial parse, we add a third segment to the parser state, called the Deleted list $D$, which includes words that have been removed from the buffer and stack after having both their children and parent specified. The order of words in $D$ is the same as the input sentence. The current partial dependency structure is then input with the graph input mechanism as relations between the words in this extended parser state. To show the effectiveness of the graph input mechanism, we exclude the composition model from the State Transformer model when integrated with the Graph-to-Graph Transformer architecture. We will demonstrate the impact of this replacement in Section 2.6.

**Sentence Tr +G2GTr.** The current partial dependency structure is input with the graph input mechanism as relations between the first sub-words of the head and dependent words of each dependency relation. For the non-first subwords of each word, we define a new dependency relation with these subwords dependent on their associated first sub-word.

### 2.4.4 Pre-Training with BERT

Initialising a Transformer model with the pre-trained parameters of BERT (Devlin et al., 2019), and then fine-tuning on the target task, has demonstrated large improvements in many tasks. But our version of the Transformer has novel inputs that were not present when BERT was trained, namely the graph inputs to the attention mechanism and the composition embeddings (for State Transformer). Also, the input sequence of State Transformer has a novel structure, which is only partially similar to the input sentences which BERT was trained on. So it is not clear that BERT pre-training will even work with this novel architecture. To evaluate whether BERT pre-training works for our proposed architectures, we also initialise the weights of our models with the first $n$ layers of BERT, where $n$ is the number of self-attention layers in the model.

## 2.5 Experimental Setup

### 2.5.1 Datasets

We evaluate our models on two types of datasets, WSJ Penn Treebank, and Universal Dependency (UD) Treebanks. Following Kulmizev et al. (2019), for evaluation, we include punctuation for UD treebanks and exclude it for the WSJ Penn Treebank (Nilsson and Nivre, 2008).[7]

[6] A worked example of both baseline models integrated with G2GTr is provided in Appendix 2.10.

[7] Description of Treebanks are provided in Appendix 2.11.

**WSJ Penn Treebank.** We train our models on the Stanford dependency version of the English Penn Treebank (Marcus et al., 1993). We use the same setting as defined in Dyer et al. (2015). We additionally add section 24 to our development set to avoid over-fitting. For PoS tags, we use Stanford PoS tagger (Toutanova et al., 2003).

**Universal Dependency Treebanks** We also train models on Universal Dependency Treebanks (UD v2.3) (Nivre et al., 2018). We evaluate our models on the list of languages defined in Kulmizev et al. (2019). This set of languages contains different scripts, various morphological complexity and character set sizes, different training sizes, and non-projectivity ratios.

### 2.5.2 Models

As strong baselines from previous work, we compare our models to previous traditional transition-based and Seq2Seq models. For a fair comparison with previous models, we consider "traditional" transition-based parsers to be those that predict a fixed set of scores for each decoding step.[8]

To investigate the usefulness of each component of the proposed parsing models, we evaluate several versions. For the State Transformer, we evaluate StateTr and StateTr+G2GTr models both with and without BERT initialisation. To further analyse the impact of Graph-to-Graph Transformer, we also compare to keeping the composition function of the StateTr model when integrated with G2GTr (StateTr+G2GTr+C). To further demonstrate the impact of the graph output mechanism, we compare to using the output embedding of the `CLS` token as the input to the transition classifiers for both the baseline model (StateCLSTr) and its combined version (StateTr+G2CLSTr). For Sentence Transformer, we evaluate the SentTr and SentTr+G2GTr models with BERT initialisation. We also evaluate the best variations of each baseline on the UD Treebanks.[9]

### 2.5.3 Details of Implementation

All hyper-parameter details are given in Appendix 2.13. Unless specified otherwise, all models have 6 self-attention layers. We use the AdamW optimiser provided by Wolf et al. (2019) to fine-tune model parameters. All our models use greedy decoding, meaning that at each step only the highest scoring parser action is considered for continuation. This was done for simplicity, although beam search could also be used. The pseudo-code for computing the elements of the graph input matrix ($p_{ij}$) for each baseline is provided in Appendix 2.14.

[8]We do not consider the models of Ma et al. (2018); Fernández-González and Gómez-Rodríguez (2019) to be comparable to traditional transition-based models like ours because they make decoding decisions between $O(n)$ alternatives. In this sense, they are in between the $O(1)$ alternatives for transition-based models and the $O(n^2)$ alternatives for graph-based models. Future work will investigate applying Graph-to-Graph Transformer to these types of parsers as well.

[9]The number of parameters and average running times for each model are provided in Appendix 2.12.

| | Dev Set | | Test Set | |
|---|---|---|---|---|
| | UAS | LAS | UAS | LAS |
| **Transition-based:** | | | | |
| Dyer et al. (2015) | | | 93.10 | 90.90 |
| Weiss et al. (2015) | | | 94.26 | 91.42 |
| Cross and Huang (2016) | | | 93.42 | 91.36 |
| Ballesteros et al. (2016) | | | 93.56 | 92.41 |
| Andor et al. (2016) | | | 94.61 | 92.79 |
| Kiperwasser and Goldberg (2016) | | | 93.90 | 91.90 |
| Yang et al. (2017) | | | 94.18 | 92.26 |
| **Seq2Seq-based:** | | | | |
| Zhang et al. (2017) | | | 93.71 | 91.60 |
| Li et al. (2018) | | | 94.11 | 92.08 |
| StateTr | 91.94 | 89.07 | 92.32 | 89.69 |
| StateTr+G2GTr | 92.53 | 90.16 | 93.07 | 91.08 |
| BERT StateTr | 94.66 | 91.94 | 95.18 | 92.73 |
| BERT StateCLSTr | 93.62 | 90.95 | 94.31 | 91.85 |
| BERT StateTr+G2GTr | **94.96** | **92.88** | **95.58** | **93.74** |
| BERT StateTr+G2CLSTr | 94.29 | 92.13 | 94.83 | 92.96 |
| BERT StateTr+G2GTr+C | 94.41 | 92.25 | 94.89 | 92.93 |
| BERT SentTr | 95.34 | 93.29 | 95.65 | 93.85 |
| BERT SentTr+G2GTr | **95.66** | **93.60** | **96.06** | **94.26** |
| BERT SentTr+G2GTr-7 layer | **95.78** | **93.74** | **96.11** | **94.33** |

Table 2.1: Comparisons to SoTA on English WSJ Treebank Stanford dependencies.

## 2.6 Results and Discussion

### 2.6.1 English Penn Treebank Result

In Table 2.1, we show several variations of our models, and previous state-of-the-art transition-based and Seq2Seq parsers on WSJ Penn Treebank.[10] For State Transformer, replacing the composition model (StateTr) with our graph input mechanism (StateTr+G2GTr) results in 9.97% / 11.66% LAS relative error reduction (RER) without / with BERT initialisation, which demonstrates its effectiveness. Comparing to the closest previous model for conditioning of the parse graph, the StateTr+G2GTr model reaches better results than the StackLSTM model (Dyer et al., 2015). Initialising our models with pre-trained BERT achieves 26.25% LAS RER for the StateTr model, and 27.64% LAS RER for the StateTr+G2GTr model, thus confirming the compatibility of our G2GTr architecture with pre-trained Transformer models. The BERT StateTr+G2GTr model outperforms previous state-of-the-art models. Removing the graph output mechanism (StateCLSTr / StateTr+G2CLSTr) results in a 12.28% / 10.53% relative performance drop for the StateTr and StateTr+G2GTr models, respectively, which demonstrates the importance of our graph output mechanism. If we consider both the graph input and output mechanisms together, adding them both (BERT StateTr+G2GTr) to BERT StateCLSTr achieves 21.33% LAS relative error reduction, which shows the synergy of using both mechanisms together. But then adding the composition model (BERT StateTr+G2GTr+C) results in an 8.84% relative drop in performance,

[10]Results are calculated with the official evaluation script provided in `https://depparse.uvt.nl/`.

| Language | Kulmizev et al. (2019) | BERT StateTr+G2GTr | BERT SentTr+G2GTr |
|---|---|---|---|
| Arabic | 81.9 | 82.63 | **83.65** |
| Basque | 77.9 | 74.03 | **83.88** |
| Chinese | 83.7 | 85.91 | **87.49** |
| English | 87.8 | 89.21 | **90.35** |
| Finnish | 85.1 | 80.87 | **89.47** |
| Hebrew | 85.5 | 87.0 | **88.75** |
| Hindi | 89.5 | 93.13 | **93.12** |
| Italian | 92.0 | 92.6 | **93.99** |
| Japanese | 92.9 | 95.25 | **95.51** |
| Korean | 83.7 | 80.13 | **87.09** |
| Russian | 91.5 | 92.34 | **93.30** |
| Swedish | 87.6 | 88.36 | **90.40** |
| Turkish | 64.2 | 56.87 | **67.77** |
| Average | 84.87 | 84.48 | **88.06** |

Table 2.2: Labelled attachment score on 13 UD corpora for Kulmizev et al. (2019) with BERT pre-training, BERT StateTr+G2GTr, and BERT SentTr+G2GTr models.

which demonstrates again that our proposed graph input method is a more effective way to model the partial parse than recursive composition models.

For Sentence Transformer, the synergy between its encoder and BERT results in excellent performance even for the baseline model (compared to Cross and Huang (2016)). Nonetheless, adding G2GTr achieves significant improvement (4.62% LAS RER), which again demonstrates the effectiveness of the Graph-to-Graph Transformer architecture. Finally, we also evaluate the BERT SentTr+G2GTr model with 7 self-attention layers instead of 6, resulting in 2.19% LAS RER, which motivates future work on larger Graph-to-Graph Transformer models.

### 2.6.2 UD Treebanks Results

In Table 2.2, we show LAS scores on 13 UD Treebanks[11]. As the baseline, we use scores of the transition-based model proposed by Kulmizev et al. (2019), which uses the deep contextualized word representations of BERT and ELMo (Peters et al., 2018) as an additional input to their parsing models. Our BERT StateTr+G2GTr model outperforms the baseline on 9 languages, again showing the power of the G2GTr architecture. But for morphology-rich languages such as Turkish and Finish, the StateTr parser design choice of only inputting the first sub-word of each word causes too much loss of information, resulting in lower results for our BERT StateTr+G2GTr model than the baseline. This problem is resolved by our SentTr parser design because all sub-words are input. The BERT SentTr+G2GTr model performs substantially better than the baseline on all languages, which confirms the effectiveness of our Graph-to-Graph Transformer architecture to capture a diversity of types of structure from a variety of corpus sizes.

[11]Unlabelled attachment scores, and results of development set are provided in the Appendix 2.15. Results are calculated with the official UD evaluation script (`https://universaldependencies.org/conll18/evaluation.html`).

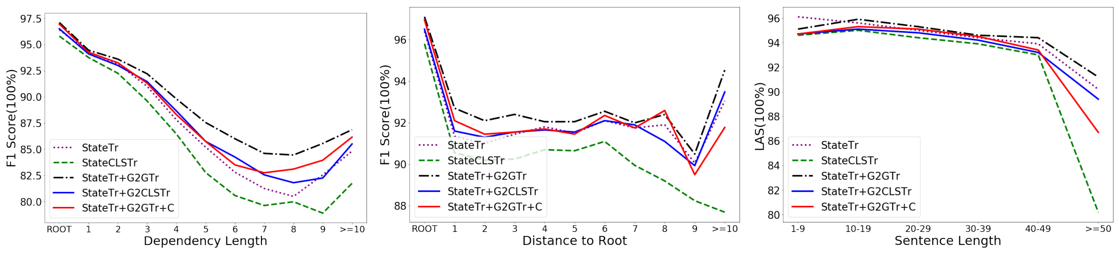


Figure 2.3: Error analysis of our models on the development set of the WSJ dataset.

### 2.6.3 Error Analysis

To analyse the effectiveness of the proposed graph input and output mechanisms in variations of our StateTr model pre-trained with BERT, we follow McDonald and Nivre (2011) and measure their accuracy as a function of dependency length, distance to root, sentence length, and dependency type, as shown in Figure 2.3 and Table 2.3.[12]. These results demonstrate that most of the improvement of the StateTr+G2GTr model over other variations comes from the hard cases which require a more global view of the sentence.

**Dependency Length.** The leftmost plot shows labelled F-scores on dependencies binned by dependency lengths. The integrated G2GTr models outperform other models on the longer (more difficult) dependencies, which demonstrates the benefit of adding the partial dependency tree to the self-attention model, which provides a global view of the sentence when the model considers long dependencies. Excluding the graph output mechanism also results in a drop in performance particularly in long dependencies. Keeping the composition component in the StateTr+G2GTr model doesn't improve performance at any length.

**Distance to Root.** The middle plot shows the labelled F-score for dependencies binned by the distance to the root, computed as the number of dependencies in the path from the dependent to the root node. The StateTr+G2GTr models outperform baseline models on nodes that are of middle depths, which tend to be neither near the root nor near the leaves, and thus require more global information, as well as deeper nodes.

**Sentence Length.** The rightmost plot shows labelled attachment scores (LAS) for sentences with different lengths. The relative stability of the StateTr+G2GTr model across different sentence lengths again shows the effectiveness of the Graph-to-Graph Transformer model on the harder cases. Not using the graph output method shows particularly bad performance on long sentences, as does keeping the composition model.

[12]We use MaltEvalNilsson and Nivre (2008) tool for computing accuracies. Tables of results for the error analysis in Figure 2.3, and Table 2.3 are in the Appendix 2.16.

| Type | StateTr+G2GTr | StateTr | StateTr+G2CLSTr |
|---|---|---|---|
| `rcmod` | 86.84 | 76.38 (-79.5%) | 83.91 (-22.3%) |
| `nsubjpass` | 95.49 | 92.70 (-61.9%) | 94.08 (-31.1%) |
| `ccomp` | 89.49 | 81.82 (-73.0%) | 87.56 (-18.4%) |
| `infmod` | 87.38 | 79.19 (-64.9%) | 84.93 (-19.4%) |
| `neg` | 95.75 | 94.84 (-21.4%) | 93.78 (-46.2%) |
| `csubj` | 76.94 | 67.93 (-39.0%) | 70.83 (-26.5%) |
| `cop` | 93.08 | 92.62 (-6.5%) | 91.58 (-21.7%) |
| `cc` | 90.90 | 90.45 (-4.9%) | 88.80 (-23.1%) |

Table 2.3: F-scores (and RER) of our full BERT model (StateTr+G2GTr), without graph inputs (StateTr), and without graph outputs (StateTr+G2CLSTr) for some dependency types on the development set of WSJ Treebank, ranked by total negative RER. Relative error reduction is computed w.r.t. the StateTr+G2GTr scores.

**Dependency Type.** Table 2.3 shows F-scores of different dependency types. Excluding the graph input (StateTr) or graph output (StateTr+G2CLSTr) mechanisms results in a substantial drop for many dependency types, especially hard cases where accuracies are relatively low, and cases such as `ccomp` which require a more global view of the sentence.

## 2.7 Conclusion

In this chapter, we proposed the Graph-to-Graph Transformer architecture, which inputs and outputs arbitrary graphs through its attention mechanisms. Each graph relation is input as a label embedding to each attention function involving the relation's tokens, and each graph relation is predicted from its token's embeddings like an attention function. We demonstrate the effectiveness of this architecture on transition-based dependency parsing, where the input graph is the partial dependency structure specified by the parse history, and the output graph is predicted one dependency at a time by the parser actions.

To establish strong baselines, we also propose two Transformer-based models for this task, called State Transformer and Sentence Transformer. The former model incorporates history and composition models, as proposed in previous work. Despite the competitive performance of these extended-Transformer parsers, adding our graph input and output mechanisms results in significant improvement. Also, the graph inputs are effective replacements for the composition models. All these results are preserved with the incorporation of BERT pre-training, which results in substantially improving the state-of-the-art in traditional transition-based dependency parsing.

As well as the generality of the graph input mechanism, the generality of the graph output mechanism means that Graph-to-Graph Transformer can be integrated with a wide variety of decoding algorithms. In Chapter 3, we apply it to non-autoregressive decoding, which addresses the computational cost of running the G2GTr model once for every dependency edge. In Chapter 4, we use G2GTr to encode the syntactic graph structure for improving the performance of semantic role labelling (SRL) task. Graph-to-Graph Transformer can also easily be applied to a wide variety of NLP tasks, which require encoding graph structures.

# Appendix

## 2.8 Preliminary Experiments

### 2.8.1 Skip Connection

Skip connections of composition model help address the vanishing gradient problem, and following experiments show that they are necessary to integrate pre-trained BERT (Devlin et al., 2019) parameters with the model:

| Model | UAS | LAS |
|---|---|---|
| BERT StateTr | 94.78 | 92.06 |
| BERT StateTr without skip | 93.16 | 90.51 |

Table 2.4: Preliminary experiments on the development set of WSJ Penn Treebank for BERT StateTr model with/without skip connections.

### 2.8.2 Position Embeddings

Following experiments show that using position embeddings for the whole sequence achieves better performance than applying separate position embeddings for each segment:

| Model | UAS | LAS |
|---|---|---|
| BERT StateTr | 94.78 | 92.06 |
| BERT StateTr with separate pos | 93.10 | 90.39 |

Table 2.5: Preliminary experiments on the development set of WSJ Penn Treebank for BERT StateTr model, and its variation with separate position embeddings for each section.

## 2.9 Composition Model

Previous work has shown that recursive neural networks are capable of inducing a representation for complex phrases by recursively embedding sub-phrases (Socher et al., 2011, 2014, 2013; Hermann and Blunsom, 2013). Dyer et al. (2015) showed that this is an effective technique for embedding the partial parse subtrees specified by the parse history in transition-based dependency parsing. Since a word in a dependency tree can have a variable number of dependents, they combined the dependency relations incrementally as they are specified by the parser.

We extend this idea by using a feed-forward neural network with `Tanh` as the activation function and skip connections. For every token in position $i$ on the stack or buffer, after deciding on step $t$, the composition model computes a vector $C^{t+1}_{a,i}$ which is added to the input embedding for that token:

$$\begin{aligned} &C^{t+1}_{a,i} = \text{Comp}((\psi^t_{a,i}, \omega^t_{a,i}, l^t_{a,i})) + \psi^t_{a,i} \\ &\text{where:} a \in \{S, B\} \end{aligned} \tag{2.11}$$

where the Comp function is a one-layer feed forward neural network, and $(\psi^t_{a,i}, \omega^t_{a,i}, l^t_{a,i})$ represents the most recent dependency relation with head $\psi^t_{a,i}$ specified by the decision at step $t$ for element in position $i$ in the stack or buffer. In arc-standard parsing, the only word which might have received a new dependent by the previous decision is the word on the top of the stack, $i$=1. This gives us the following definition of $(\psi^t_{a,i}, \omega^t_{a,i}, l^t_{a,i})$:

$$\begin{cases} \texttt{RIGHT-ARC}(l): \\ \quad \psi^t_{S,1}=C^t_{S,2},\ \omega^t_{S,1}=C^t_{S,1},\ l^t_{S,1}=l, \\ \quad \psi^t_{S,i\neq1}=C^t_{S,i+1},\ \omega^t_{S,i\neq1}=\omega^t_{S,i+1},\ l^t_{S,i\neq1}=l^t_{S,i+1} \\ \quad \psi^t_{B,i}=C^t_{B,i} \\ \texttt{LEFT-ARC}(l): \\ \quad \psi^t_{S,1}=C^t_{S,1},\ \omega^t_{S,1}=C^t_{S,2},\ l^t_{S,1}=l, \\ \quad \psi^t_{S,i\neq1}=C^t_{S,i+1},\ \omega^t_{S,i\neq1}=\omega^t_{S,i+1},\ l^t_{S,i\neq1}=l^t_{S,i+1} \\ \quad \psi^t_{B,i}=C^t_{B,i} \\ \texttt{SHIFT}: \\ \quad \psi^t_{S,1}=C^t_{B,1},\ \omega^t_{S,1}=\omega^t_{B,1},\ l^t_{S,1}=l^t_{B,1} \\ \quad \psi^t_{S,i\neq1}=C^t_{S,i-1},\ \omega^t_{S,i\neq1}=\omega^t_{S,i-1},\ l^t_{S,i\neq1}=l^t_{S,i-1} \\ \quad \psi^t_{B,i}=C^t_{B,i+1},\ \omega^t_{B,i}=\omega^t_{B,i+1},\ l^t_{B,i}=l^t_{B,i+1} \\ \texttt{SWAP}: \\ \quad \psi^t_{S,1}=C^t_{S,1} \\ \quad \psi^t_{S,i\neq1}=C^t_{S,i+1},\ \omega^t_{S,i\neq1}=\omega^t_{S,i+1},\ l^t_{S,i\neq1}=l^t_{S,i+1} \\ \quad \psi^t_{B,1}=C^t_{S,2},\ \omega^t_{B,1}=\omega^t_{S,2},\ l^t_{B,1}=l^t_{S,2} \\ \quad \psi^t_{B,i\neq1}=C^t_{B,i-1},\ \omega^t_{B,i\neq1}=\omega^t_{B,i-1},\ l^t_{B,i\neq1}=l^t_{B,i-1} \end{cases} \tag{2.12}$$

where $C^t_{S,1}$ and $C^t_{S,2}$ are the embeddings of the top two elements of the stack at time step $t$, and $C^t_{B,1}$ is the embedding of the word on the front of the buffer at time $t$. $l^t_{a,i} \in R^m$ is the label embedding of the specified relation, including its direction. For all words which have not received a new dependent, the composition is computed anyway with the most recent dependent and label (with a `[NULL]` dependent and label of that position `[L-NULL]` if there is no dependency relation with element $i$ as the head).[13]

At $t = 0$, $C^t_{a,i}$ is set to the initial token embedding $T_{w_i}$. The model then computes Equation 2.11 iteratively at each step $t$ for each token on the stack or buffer.

There is a skip connection in Equation 2.11 to address the vanishing gradient problem. Also, preliminary experiments showed that without this skip connection to bias the composition model towards the initial token embeddings $T_{w_i}$, integrating pre-trained BERT (Devlin et al., 2019) parameters into the model did not work.

[13]Preliminary experiments indicated that not updating the composition embedding for these cases resulted in worse performance.

## 2.10 Example of the Graph-to-Graph Transformer parsing

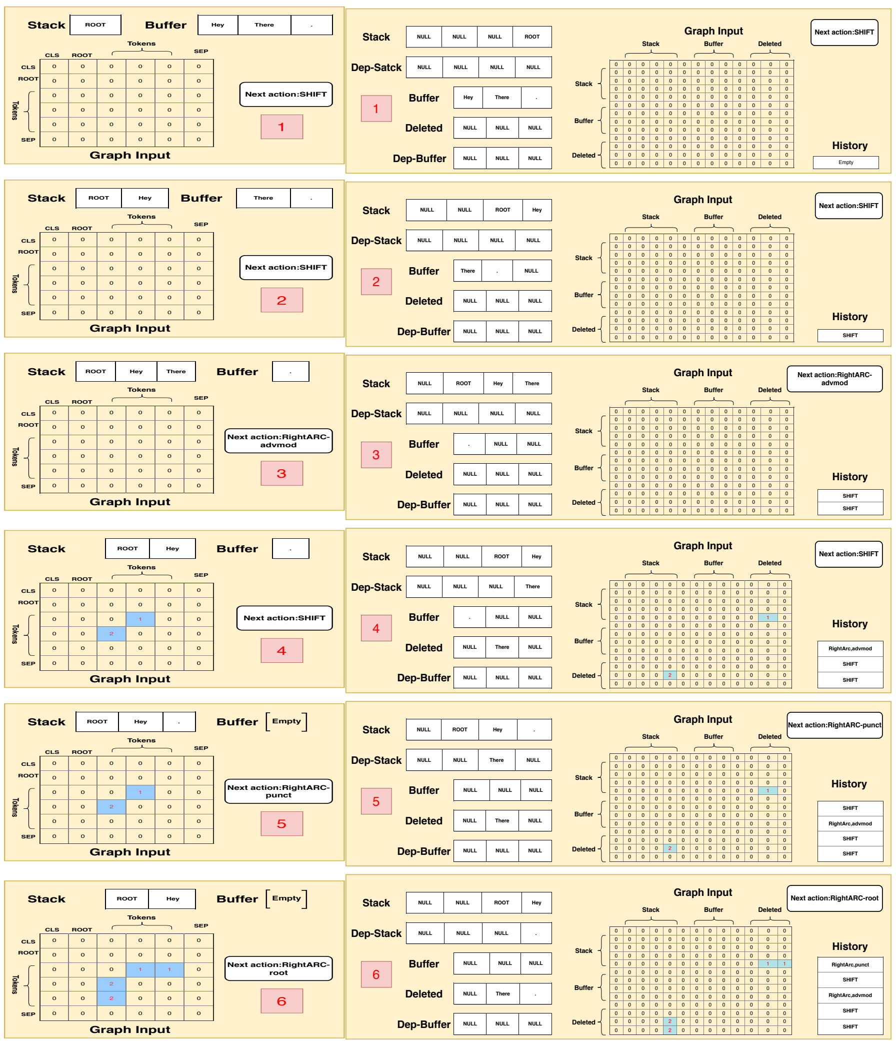

(a) SentTr+G2G-Tr. (b) StateTr+G2G-Tr.

Figure 2.4: Example of Graph-to-Graph Transformer model integrated with SentTr and StateTr on UD English Treebank (initial sentence: "Hey There .").

## 2.11 Description of Treebanks

### 2.11.1 English Penn Treebank Description

The dataset can be found here under LDC licence. Stanford PoS tagger and constituency converter can be downloaded from here and here, respectively. Here is the detailed information of English Penn Treebank:

| Language | Version | Non-projectivity ratio | Train size(2-21) | Development size(22,24) | Test size(23) |
|---|---|---|---|---|---|
| English | 3 | 0.1% | 39’832 | 3’046 | 2’416 |

Table 2.6: Description of English Penn Treebank.

### 2.11.2 UD Treebanks Description

UD Treebanks v2.3 are provided in here. Pre-processing tools can be found here.

| Language | Family | Treebank | Order | Train size | Development size | Test size | Non-projectivity ratio |
|---|---|---|---|---|---|---|---|
| Arabic | non-IE | PADT | VSO | 6.1K | 0.9K | 0.68K | 9.2% |
| Basque | non-IE | BDT | SOV | 5.4K | 1.8K | 1.8K | 33.5% |
| Chinese | non-IE | GSD | SVO | 4K | 0.5K | 0.5K | 0.6% |
| English | IE | EWT | SVO | 12.5K | 2k | 2.1K | 5.3% |
| Finnish | non-IE | TDT | SVO | 12.2K | 1.3K | 1.5K | 6.2% |
| Hebrew | non-IE | HTB | SVO | 5.2K | 0.48K | 0.49K | 7.6% |
| Hindi | IE | HDTB | SOV | 13.3K | 1.7K | 1.7K | 13.8% |
| Italian | IE | ISDT | SVO | 13.1K | 0.56K | 0.48K | 1.9% |
| Japanese | non-IE | GSD | SOV | 7.1K | 0.51K | 0.55K | 2.7% |
| Korean | non-IE | GSD | SOV | 4.4K | 0.95K | 0.99K | 16.2% |
| Russian | IE | SynTagRus | SVO | 48.8K | 6.5K | 6.5K | 8.0% |
| Swedish | IE | Talbanken | SVO | 4.3K | 0.5K | 1.2K | 3.3% |
| Turkish | non-IE | IMST | SOV | 3.7K | 0.97K | 0.97K | 11.1% |

Table 2.7: Description of languages chosen from UD Treebanks v2.3.

## 2.12 Running Details of Proposed Models

We provide the number of parameters and average run times for each model. For a better understanding, average run time is computed per transition (Second/transition). All experiments are computed with graphics processing unit (GPU), specifically the NVIDIA V100 model. The total number of transitions in the train and development sets are 79664 and 6092, respectively.

| Model | No. parameters | Train (sec/transition) | Evaluation (sec/transition) | Evaluation (sent/sec) | Evaluation (token/sec) |
|---|---|---|---|---|---|
| StateTr | 90.33M | 0.098 | 0.031 | 16.2 | 388.13 |
| StateTr+G2GTr | 105.78M | 0.226 | 0.071 | 7.05 | 168.91 |
| SentTr | 63.23M | 0.112 | 0.026 | 19.27 | 460.6 |
| SentTr+G2GTr | 63.27M | 0.138 | 0.031 | 16.2 | 388.13 |

Table 2.8: Running details of our models on WSJ Penn Treebank.

## 2.13 Hyper-parameters for our parsing models

For hyper-parameter selection, we use manual tuning to find the best numbers. For BERT (Devlin et al., 2019) hyper-parameters, we apply the same optimization strategy as suggested by Wolf et al. (2019). For classifiers and composition model, we use a one-layer feed-forward neural network for simplicity. Then, we pick hyper-parameters based on previous works (Devlin et al., 2019; Dyer et al., 2015). We use two separate optimisers for pre-trained parameters (BERT here) and randomly initialised parameters for better convergence that is shown to be useful in Kondratyuk and Straka (2019). Early stopping (based on LAS) is used during training. The only tuning strategy that has been tried is to use one optimiser for all parameters or two different optimisers for pre-trained parameters and randomly initialised ones. For the latter case, Learning rate for randomly initialised parameters is set to $1e-4$. Results of different variations on the development set of WSJ Penn Treebank are as follows:

| Model | UAS | LAS |
|---|---|---|
| BERT StateTr with one optimiser | 94.66 | 91.94 |
| BERT StateTr with two optimisers | 94.24 | 91.67 |
| Expected (Average) Performance | 94.45 | 91.81 |
| BERT StateTr+G2GTr with one optimiser | 94.96 | 92.88 |
| BERT StateTr+G2GTr with two optimisers | 94.75 | 92.49 |
| Expected (Average) Performance | 94.86 | 92.69 |
| BERT SentTr with one optimiser | 95.34 | 93.29 |
| BERT SentTr with two optimisers | 95.49 | 93.29 |
| Expected (Average) Performance | 95.42 | 93.29 |
| BERT SentTr+G2GTr with one optimiser | 95.27 | 93.18 |
| BERT SentTr+G2GTr with two optimisers | 95.66 | 93.60 |
| Expected (Average) Performance | 95.47 | 93.40 |

Table 2.9: Results on the development set of WSJ Penn Treebank for different optimisation strategy.

Hyper-parameters for training our models are defined as [14]:

| Component | Specification |
|---|---|
| **Optimiser** | BertAdam |
| Learning Rate | 1e-5 |
| Base Learning Rate | 1e-4 |
| Adam Betas($b_1$,$b_2$) | (0.9,0.999) |
| Adam Epsilon | 1e-6 |
| Weight Decay | 0.01 |
| Max-Grad-Norm | 1 |
| Warm-up | 0.01 |
| **Self-Attention** | |
| No. Layers($n$) | 6 |
| No. Heads | 12 |
| Embedding size | 768 |
| Max Position Embedding | 512 |
| BERT model | bert-base-uncased |
| **Classifiers** | |
| No. Layers | 2 |
| Hidden size(Exist) | 500 |
| Hidden size(Relation) | 100 |
| Drop-out | 0.05 |
| Activation | $ReLU$ |
| **History Model** | LSTM |
| No. Layers | 2 |
| Hidden Size | 768 |
| **Composition Model** | |
| No. Layers | 2 |
| Hidden size | 768 |
| Epochs | 12 |

Table 2.10: Hyper-parameters for training our models.

[14]For UD Treebanks, we train our model for 20 epochs, and use "bert-multilingual-cased" for the initialisation. We use pre-trained BERT models of `https://github.com/google-research/bert`.

## 2.14 Pseudo-Code of Graph Input Mechanism

**Algorithm 1** Pseudo-code of building graph input matrix for StateTr+G2GTr model.

```
1: Graph Sentence(input of attention): $P$
2: Graph Input: $G$
3: Actions: $A = (a_1, ..., a_T)$
4: Input: $(S, B, D)$
5: for $k \leftarrow 1, T$ do
6:     if $a_k$ = SHIFT or SWAP then
7:         continue
8:     else
9:         new relation: $i \xrightarrow{l} j$
10:        $G_{i,j} = 1$
11:        $G_{j,i} = 2$
12:        pop $x_j$ from stack
13:        change mask of $x_j$ to one
14:        add $l$ to input embedding of $x_j$
15:        $P$:select $G$ based on Input $(S, B, D)$
16:    end if
17: end for
```

**Algorithm 2** Pseudo-code of building graph input matrix for SentTr+G2GTr model.

```
1: Graph Sentence(input of attention): $P$
2: Graph Input: $G$
3: Actions: $A = (a_1, ..., a_T)$
4: Input: initial tokens
5: for $k \leftarrow 1, T$ do
6:     if $a_k$ = SHIFT or SWAP then
7:         continue
8:     else
9:         new relation: $i \xrightarrow{l} j$
10:        $G_{i,j} = 1$
11:        $G_{j,i} = 2$
12:        add $l$ to input embedding of $j$-th word
13:        $P = G$
14:    end if
15: end for
```

## 2.15 UD Treebanks Results

BERT SentTr+G2GTr results:

| Language | Test set-UAS | Dev set-UAS | Dev set-LAS |
|---|---|---|---|
| Arabic | 87.65 | 87.01 | 82.64 |
| Basque | 87.17 | 86.53 | 83.25 |
| Chinese | 89.74 | 88.36 | 85.79 |
| English | 92.05 | 93.05 | 91.3 |
| Finnish | 91.46 | 91.37 | 89.72 |
| Hebrew | 90.85 | 91.92 | 89.55 |
| Hindi | 95.77 | 95.86 | 93.17 |
| Italian | 95.15 | 95.22 | 93.9 |
| Japanese | 96.21 | 96.68 | 96.04 |
| Korean | 89.42 | 87.57 | 84.94 |
| Russian | 94.42 | 94.0 | 92.70 |
| Swedish | 92.49 | 87.26 | 85.39 |
| Turkish | 74.23 | 72.52 | 66.05 |
| Average | 90.51 | 89.80 | 87.27 |

Table 2.11: Dependency scores of BERT SentTr+G2GTr model on the development and test sets of UD Treebanks.

BERT StateTr+G2GTr results:

| Language | Test set-UAS | Dev set-UAS | Dev set-LAS |
|---|---|---|---|
| Arabic | 86.85 | 86.41 | 81.73 |
| Basque | 80.91 | 80.01 | 73.2 |
| Chinese | 87.90 | 86.64 | 84.15 |
| English | 90.91 | 91.85 | 90.11 |
| Finnish | 84.35 | 82.91 | 78.73 |
| Hebrew | 89.51 | 90.36 | 87.85 |
| Hindi | 95.65 | 95.92 | 93.30 |
| Italian | 93.5 | 93.61 | 92.18 |
| Japanese | 95.99 | 96.18 | 95.58 |
| Korean | 84.35 | 82.13 | 77.78 |
| Russian | 93.87 | 93.41 | 92.09 |
| Swedish | 92.49 | 90.72 | 88.36 |
| Turkish | 65.99 | 65.92 | 56.96 |
| Average | 87.87 | 87.39 | 84.01 |

Table 2.12: Dependency scores of BERT StateTr+G2GTr model on the development and test sets of UD Treebanks.

## 2.16 Error-Analysis

### 2.16.1 Dependency Length

| Model | ROOT | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | >=10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| BERT StateTr | 96.40 | 94.30 | 93.15 | 91.00 | 87.80 | 85.19 | 82.80 | 81.25 | 80.49 | 82.54 | 84.82 |
| BERT StateCLSTr | 95.80 | 93.75 | 92.25 | 89.60 | 86.45 | 82.77 | 80.58 | 79.62 | 79.98 | 78.90 | 81.74 |
| BERT StateTr+G2GTr | 97.10 | 94.45 | 93.60 | 92.20 | 89.80 | 87.54 | 86.00 | 84.60 | 84.45 | 85.55 | 86.86 |
| BERT StateTr+G2CLSTr | 96.50 | 94.10 | 93.00 | 91.45 | 88.65 | 85.74 | 84.25 | 82.55 | 81.80 | 82.25 | 85.50 |
| BERT StateTr+G2GTr+C | 96.95 | 94.25 | 93.25 | 91.30 | 88.30 | 85.74 | 83.50 | 82.75 | 83.10 | 83.95 | 86.15 |

Table 2.13: labelled F-Score vs dependency relation length

| Model | ROOT | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | >=10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| BERT StateTr | 3046 | 31245 | 14478 | 7565 | 4153 | 2461 | 1681 | 1185 | 933 | 808 | 5415 |
| BERT StateCLSTr | 3046 | 31409 | 14473 | 7553 | 4131 | 2406 | 1641 | 1153 | 923 | 832 | 5403 |
| BERT StateTr+G2GTr | 3047 | 31240 | 14457 | 7572 | 4171 | 2465 | 1688 | 1188 | 941 | 811 | 5390 |
| BERT StateTr+G2CLSTr | 3046 | 31249 | 14447 | 7537 | 4143 | 2453 | 1701 | 1193 | 953 | 814 | 5434 |
| BERT StateTr+G2GTr+C | 3047 | 31304 | 14430 | 7514 | 4137 | 2449 | 1693 | 1182 | 951 | 830 | 5433 |
| Gold bins | 3046 | 31126 | 14490 | 7551 | 4155 | 2508 | 1698 | 1195 | 953 | 821 | 5427 |

Table 2.14: Size of each bin based on dependency length

### 2.16.2 Distance to Root

| Model | ROOT | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | >=10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| BERT StateTr | 96.40 | 91.35 | 91.00 | 91.45 | 91.80 | 91.50 | 92.10 | 91.75 | 91.90 | 90.06 | 93.06 |
| BERT StateCLSTr | 95.80 | 90.55 | 90.20 | 90.25 | 90.70 | 90.65 | 91.10 | 89.95 | 89.20 | 88.24 | 87.69 |
| BERT StateTr+G2GTr | 97.10 | 92.70 | 92.10 | 92.40 | 92.05 | 92.05 | 92.55 | 91.99 | 92.39 | 90.49 | 94.54 |
| BERT StateTr+G2CLSTr | 96.50 | 91.60 | 91.30 | 91.55 | 91.65 | 91.55 | 92.10 | 91.90 | 91.10 | 89.93 | 93.48 |
| BERT StateTr+G2GTr+C | 96.95 | 92.10 | 91.45 | 91.55 | 91.70 | 91.45 | 92.35 | 91.75 | 92.59 | 89.50 | 91.77 |

Table 2.15: labelled F-Score vs distance to root

| Model | ROOT | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | >=10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| BERT StateTr | 3046 | 16081 | 15965 | 12999 | 9301 | 6488 | 3964 | 2242 | 1343 | 740 | 801 |
| BERT StateCLSTr | 3046 | 15963 | 15798 | 12793 | 9222 | 6419 | 3974 | 2331 | 1358 | 827 | 1239 |
| BERT StateTr+G2GTr | 3047 | 16024 | 15889 | 12875 | 9415 | 6463 | 3995 | 2349 | 1347 | 743 | 823 |
| BERT StateTr+G2CLSTr | 3046 | 16064 | 15975 | 12971 | 9327 | 6488 | 3963 | 2279 | 1316 | 708 | 833 |
| BERT StateTr+G2GTr+C | 3047 | 16079 | 15947 | 13020 | 9419 | 6481 | 3930 | 2259 | 1274 | 701 | 813 |
| Gold bins | 3046 | 16002 | 16142 | 13064 | 9403 | 6411 | 3923 | 2298 | 1298 | 702 | 681 |

Table 2.16: Size of each bin based on distance to root

### 2.16.3 Sentence Length

| Model | 1-9 | 10-19 | 20-29 | 30-39 | 40-49 | >= 50 |
|---|---|---|---|---|---|---|
| BERT StateTr | 96.1 | 95.6 | 95.0 | 94.4 | 93.9 | 90.2 |
| BERT StateCLSTr | 94.6 | 95.0 | 94.4 | 93.9 | 93.0 | 80.2 |
| BERT StateTr+G2GTr | 95.1 | 95.9 | 95.3 | 94.6 | 94.4 | 91.2 |
| BERT StateTr+G2CLSTr | 94.7 | 95.1 | 94.8 | 94.2 | 93.2 | 89.4 |
| BERT StateTr+G2GTr+C | 94.7 | 95.3 | 95.1 | 94.5 | 93.4 | 86.7 |

Table 2.17: LAS vs. sentence length

### 2.16.4 Dependency Type Analysis

| Type | StateTr+G2GTr | StateTr | StateTr+G2CLSTr |
|---|---|---|---|
| acomp | 68.62 | 58.60 (-31.9%) | 66.04 (-8.2%) |
| advcl | 82.75 | 70.68 (-70.0%) | 81.85 (-5.2%) |
| advmod | 83.85 | 84.40 (3.4%) | 84.35 (3.1%) |
| amod | 92.55 | 92.25 (-4.1%) | 92.30 (-3.4%) |
| appos | 87.85 | 84.94 (-23.9%) | 83.25 (-37.8%) |
| aux | 98.55 | 98.35 (-13.8%) | 98.45 (-6.9%) |
| auxpass | 96.65 | 96.04 (-18.0%) | 95.84 (-24.0%) |
| cc | 90.90 | 90.45 (-4.9%) | 88.80 (-23.1%) |
| ccomp | 89.49 | 81.82 (-73.0%) | 87.56 (-18.4%) |
| conj | 86.45 | 84.70 (-12.9%) | 84.15 (-17.0%) |
| cop | 93.08 | 92.62 (-6.5%) | 91.58 (-21.7%) |
| csubj | 76.94 | 67.93 (-39.0%) | 70.83 (-26.5%) |
| dep | 54.66 | 50.88 (-8.3%) | 51.99 (-5.9%) |
| det | 98.25 | 98.30 (2.9%) | 98.00 (-14.3%) |
| discourse | 15.40 | 15.40 (-0.0%) | 28.60 (15.6%) |
| dobj | 94.85 | 94.10 (-14.6%) | 93.95 (-17.5%) |
| expl | 96.39 | 94.99 (-38.8%) | 96.39 (-0.0%) |
| infmod | 87.38 | 79.19 (-64.9%) | 84.93 (-19.4%) |
| iobj | 88.01 | 90.66 (22.1%) | 84.24 (-31.4%) |
| mark | 95.04 | 95.19 (2.9%) | 94.84 (-4.1%) |
| mwe | 86.46 | 89.71 (24.0%) | 88.36 (14.1%) |
| neg | 95.75 | 94.84 (-21.4%) | 93.78 (-46.2%) |
| nn | 94.25 | 94.10 (-2.6%) | 93.65 (-10.5%) |
| npadvmod | 91.89 | 92.75 (10.5%) | 90.64 (-15.5%) |
| nsubj | 96.35 | 95.55 (-21.9%) | 95.60 (-20.6%) |
| nsubjpass | 95.49 | 92.70 (-61.9%) | 94.08 (-31.1%) |
| num | 95.25 | 94.89 (-7.4%) | 95.15 (-2.0%) |
| number | 92.50 | 90.65 (-24.6%) | 92.10 (-5.3%) |
| parataxis | 69.59 | 62.89 (-22.1%) | 72.10 (8.2%) |
| partmod | 82.11 | 72.82 (-52.0%) | 79.74 (-13.3%) |
| pcomp | 88.12 | 86.49 (-13.7%) | 85.77 (-19.8%) |
| pobj | 97.15 | 96.95 (-7.0%) | 96.60 (-19.3%) |
| poss | 97.60 | 97.15 (-18.7%) | 97.70 (4.2%) |
| possessive | 98.29 | 98.04 (-14.5%) | 98.44 (8.9%) |
| preconj | 85.71 | 84.65 (-7.5%) | 84.65 (-7.5%) |
| predet | 79.34 | 79.34 (-0.0%) | 77.44 (-9.2%) |
| prep | 90.25 | 89.90 (-3.6%) | 89.50 (-7.7%) |
| prt | 84.48 | 83.42 (-6.9%) | 83.10 (-8.9%) |
| punct | 88.45 | 88.05 (-3.5%) | 87.65 (-6.9%) |
| quantmod | 86.97 | 84.40 (-19.7%) | 84.49 (-19.0%) |
| rcmod | 86.84 | 76.38 (-79.5%) | 83.91 (-22.3%) |
| root | 97.15 | 96.40 (-26.3%) | 96.50 (-22.8%) |
| tmod | 86.02 | 86.38 (2.6%) | 85.03 (-7.1%) |
| xcomp | 90.75 | 84.44 (-68.2%) | 89.75 (-10.8%) |

Table 2.18: F-score of StateTr, StateTr+G2GTr, and StateTr+G2CLSTr models for the dependency types on the development set of WSJ Treebank. Relative error reduction is computed by considering StateTr+G2GTr scores as the reference.

# 3 Recursive Non-Autoregressive Graph-to-Graph Transformer for Dependency Parsing with Iterative Refinement

In this chapter, we introduce Recursive Non-autoregressive Graph-to-Graph Transformer architecture (RNGTr) for the iterative refinement of arbitrary graphs through the recursive application of a non-autoregressive Graph-to-Graph Transformer (defined in Chapter 2) and apply it to syntactic dependency parsing. We demonstrate the power and effectiveness of RNGTr on several dependency corpora, using a refinement model pre-trained with BERT. We also introduce Syntactic Transformer (SynTr), a non-recursive parser similar to our refinement model. RNGTr can improve the accuracy of a variety of initial parsers on 13 languages from the Universal Dependencies Treebanks, English and Chinese Penn Treebanks, and the German CoNLL2009 corpus, even improving over the new state-of-the-art results achieved by SynTr, significantly improving the state-of-the-art for all corpora tested.

## 3.1 Introduction

Self-attention models, such as Transformer (Vaswani et al., 2017), have been hugely successful in a wide range of natural language processing (NLP) tasks, especially when combined with language-model pre-training, such as BERT (Devlin et al., 2019). These architectures contain a stack of self-attention layers which can capture long-range dependencies over the input sequence, while still representing its sequential order using absolute position encodings. Alternatively, Shaw et al. (2018) proposes to define sequential order with relative position encodings, which are input to the self-attention functions. In Chapter 2, we extended this sequence input method to the input of arbitrary graph relations via the self-attention mechanism, and combined it with an attention-like function for graph relation prediction, resulting in Graph-to-Graph Transformer architecture (G2GTr).

The G2GTr architecture could be used to predict all the edges of a graph in parallel, but such predictions are non-autoregressive. They thus cannot fully model the interactions between edges. For sequence prediction, this problem has been addressed with non-autoregressive iterative refinement (Awasthi et al., 2019; Lee et al., 2018b; Lichtarge et al., 2018; Novak et al., 2016). Interactions between different positions in the string are modelled by conditioning on a previous version of the same string.

In this chapter, we propose a new graph prediction architecture which takes advantage of the full graph-to-graph functionality of G2GTr to apply a G2GTr model to refine the output graph recursively. This architecture predicts all edges of the graph in parallel, and is therefore non-autoregressive, but can still capture any between-edge dependency by conditioning on the previous version of the graph, like an auto-regressive model.

This proposed Recursive Non-autoregressive Graph-to-Graph Transformer (RNGTr) architecture has three components. First, an initialisation model computes an initial graph, which can be any given model for the task, even a trivial one. Second, a G2GTr model takes the previous graph as input and predicts each edge of the target graph. Third, a decoding algorithm finds the best graph given these edge predictions. The second and third components are applied recursively to do iterative refinement of the output graph until some stopping criterion is met. The final output graph is the graph output by the final decoding step.

The RNG Transformer architecture can be applied to any task with a sequence or graph as input and a graph over the same set of nodes as output. We evaluate RNGTr on syntactic dependency parsing because it is a difficult structured prediction task, state-of-the-art initial parsers are extremely competitive, and there is little previous evidence that non-autoregressive models (as in graph-based dependency parsers) are not sufficient for this task. We aim to show that capturing correlations between dependencies with non-autoregressive iterative refinement results in improvements, even in the challenging case of state-of-the-art dependency parsers.

The evaluation demonstrates improvements with several initial parsers, including previous state-of-the-art dependency parsers, and the empty parse. We also introduce a strong Transformer-based dependency parser pre-trained with BERT (Devlin et al., 2019), called Syntactic Transformer (SynTr), using it both for our initial parser and as the basis of our refinement model. Results on 13 languages from the Universal Dependencies Treebanks (Nivre et al., 2018), English and Chinese Penn Treebanks (Xue et al., 2002; Marcus et al., 1993), and German CoNLL 2009 corpus (Hajič et al., 2009) show significant improvements over all initial parsers and the state-of-the-art.[1]

In this chapter, we make the following contributions:

- We propose a novel architecture for the iterative refinement of arbitrary graphs (RNGTr) which combines non-autoregressive edge prediction with conditioning on the complete graph.
- We propose a RNGTr model of syntactic dependency parsing.
- We demonstrate significant improvements over the previous state-of-the-art dependency parsing results on Universal Dependency Treebanks, Penn Treebanks, and the German CoNLL 2009 corpus.

[1]Our implementation is available at: `https://github.com/idiap/g2g-transformer`

## 3.2 Dependency Parsing

Syntactic dependency parsing is a critical component in a variety of natural language understanding tasks, such as semantic role labelling (Marcheggiani and Titov, 2017), machine translation (Chen et al., 2017), relation extraction (Zhang et al., 2018), and natural language inference (Pang et al., 2019). There are several approaches to compute the dependency tree. *Transition-based* parsers predict the dependency graph one edge at a time through a sequence of parsing actions (Zhang and Nivre, 2011; Titov and Henderson, 2007c; Nivre and Scholz, 2004; Yamada and Matsumoto, 2003). As in our approach, *transformation-based* (Satta and Brill, 1996) and *corrective modeling* parsers use various methods (e.g. (Zheng, 2017; Hennig and Köhn, 2017; Torres Martins et al., 2008; Attardi and Ciaramita, 2007; Knight and Graehl, 2005; Hall and Novák, 2005)) to correct an initial parse. We take a graph-based approach to this correction. *Graph-based* parsers (Koo and Collins, 2010; McDonald et al., 2005a; Eisner, 1996) compute scores for every possible dependency edge and then apply a decoding algorithm to find the highest scoring total tree. Typically neural graph-based models consist of two components: an *encoder* which learns context-dependent vector representations for the nodes of the dependency graph, and a *decoder* that computes the dependency scores for each pair of nodes and then applies a decoding algorithm to find the highest-scoring dependency tree.

There are several approaches to capture correlations between dependency edges in graph-based models. In first-order models, such as Maximum Spanning Tree (MST) (McDonald et al., 2005b; i Chu and Liu, 1965; Edmonds, 1967), the score for an edge must be computed without being sure what other edges the model will choose. The model itself only imposes the discrete tree constraint between edges. In higher-order models (Tchernowitz et al., 2016; Ma and Zhao, 2012; Zhang and McDonald, 2012; Carreras, 2007; Koo and Collins, 2010; McDonald and Pereira, 2006), they keep some between-edge information, but require more decoding time.

In this study, we apply first-order models, specifically the MST algorithm, and show that it is possible to keep correlations between edges without increasing the time complexity by recursively conditioning each edge score on a previous prediction of the complete dependency graph.

## 3.3 RNG Transformer

The RNG Transformer architecture is illustrated in Figure 3.1, in this case, applied to the dependency parsing task. The input to a RNGTr model specifies the input nodes $W = (w_1, w_2, \ldots, w_N)$ (e.g. a sentence), and the output is the final graph $G^T$ (e.g. a parse tree) over this set of nodes. The first step is to compute an initial graph of $G^0$ over $W$, which can be done with any model. Then each recursive iteration takes the previous graph $G^{t-1}$ as input and predicts a new graph $G^t$.

The RNGTr model predicts $G^t$ with a novel version of a Graph-to-Graph Transformer. Unlike the proposed model in Chapter 2, this G2GTr model predicts every edge of the graph in a single non-autoregressive step. As previously, the G2GTr first encodes the input graph $G^{t-1}$ in a set of

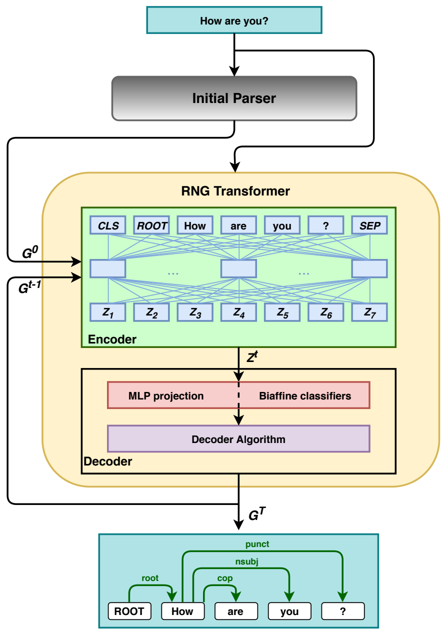


Figure 3.1: The Recursive Non-autoregressive Graph-to-Graph Transformer architecture.

contextualised vector representations $Z = (z_1, z_2, \ldots, z_N)$, with one vector for each node of the graph. The decoder component then predicts the output graph $G^t$ by first computing scores for each possible edge between each pair of nodes and then applying a decoding algorithm to output the highest-scoring complete graph.

The RNGTr model can be formalised in terms of an encoder $\mathrm{E}^{\mathrm{RNG}}$ and a decoder $\mathrm{D}^{\mathrm{RNG}}$:

$$\begin{cases} Z^t = \mathrm{E}^{\mathrm{RNG}}(W, P, G^{t-1}) \\ G^t = \mathrm{D}^{\mathrm{RNG}}(Z^t) \end{cases} \qquad t = 1, \ldots, T \tag{3.1}$$

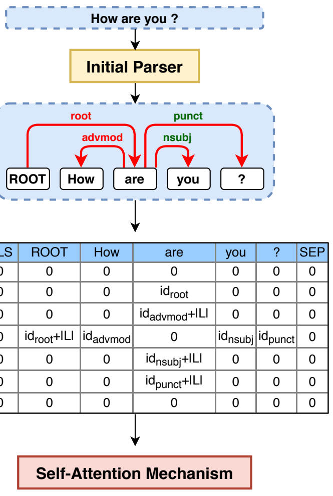
where $W = (w_1, w_2, , \ldots, w_N)$ is the input sequence of tokens, $P = (p_1, p_2, , \ldots, p_N)$ is their associated properties, and $T$ is the number of refinement iterations.

In the case of dependency parsing, $W$ are the words and symbols, $P$ are their part-of-speech tags, and the predicted graph at iteration $t$ is specified as:

$$\begin{aligned} &G^t = \{(i, j, l),\ j = 3, \ldots, N-1\} \\ &\text{where } 2 \le i \le N-1,\ l \in L \end{aligned} \tag{3.2}$$

Each word $w_j$ has one head (parent) $w_i$ with dependency label $l$ from the label set $L$, where the parent can also be the `ROOT` symbol $w_2$ (see Section 3.3.1).

The following sections describe in more detail each element of the proposed RNGTr dependency parsing model.



| | CLS | ROOT | How | are | you | ? | SEP |
|---|---|---|---|---|---|---|---|
| CLS | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ROOT | 0 | 0 | 0 | $id_{root}$ | 0 | 0 | 0 |
| How | 0 | 0 | 0 | $id_{advmod}$+lLl | 0 | 0 | 0 |
| are | 0 | $id_{root}$+lLl | $id_{advmod}$ | 0 | $id_{nsubj}$ | $id_{punct}$ | 0 |
| you | 0 | 0 | 0 | $id_{nsubj}$+lLl | 0 | 0 | 0 |
| ? | 0 | 0 | 0 | $id_{punct}$+lLl | 0 | 0 | 0 |
| SEP | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 3.2: Example of inputting dependency graph to the self-attention mechanism.

### 3.3.1 Encoder

To compute the embeddings $Z^t$ for the nodes of the graph, we use the Graph-to-Graph Transformer architecture proposed in Chapter 2, including similar mechanism to input the previously predicted dependency graph $G^{t-1}$ to the attention mechanism. This graph input allows the node embeddings to include both token-level and relation-level information.

#### Input Embeddings

The RNGTr model receives a sequence of input tokens ($W$) with their associated properties ($P$) and builds a sequence of input embeddings ($X$). For compatibility with BERT's input token representation (Devlin et al., 2019), the sequence of input tokens starts with `CLS` and ends with `SEP` symbols. For dependency parsing, it also adds the `ROOT` symbol to the front of the sentence to represent the root of the dependency tree. To build token representation for a sequence of input tokens, we sum several vectors. For the input words and symbols, we sum the token embeddings of a pre-trained BERT model $\mathrm{EMB}(w_i)$, and learned representations $\mathrm{EMB}(p_i)$ of their Part-of-Speech tags $p_i$. To keep the order information of the initial sequence, we add the position embeddings of pre-trained BERT $F_i$ to our token embeddings. The final input representations are the sum of the position embeddings and the token embeddings:

$$x_i = F_i + \mathrm{EMB}(w_i) + \mathrm{EMB}(p_i), \ i = 1, 2, ..., N \tag{3.3}$$

#### Self-Attention Mechanism

Conditioning on the previously predicted output graph $G^{t-1}$ is made possible by inputting relation embeddings to the self-attention mechanism. This edge input method was initially proposed by Shaw et al. (2018) for relative position encoding, and extending to unlabelled dependency graphs in Chapter 2. We use it to input labelled dependency graphs, by adding relation label embeddings to both the value function and the attention weight function.

Transformers have multiple layers of self-attention, each with multiple heads. The RNGTr architecture uses the same architecture as BERT (Devlin et al., 2019) but changes the functions used by each attention head. Given the token embeddings $X$ at the previous layer and the input graph $G^{t-1}$, the values $A=(a_1,\ldots,a_N)$ computed by an attention head are:

$$a_i = \sum_j \alpha_{ij}(x_j \boldsymbol{W^V} + r_{ij}^{t-1}\boldsymbol{W_2^L}) \tag{3.4}$$

where $r_{ij}^{t-1}$ is a one-hot vector that represents the labelled dependency relation between $i$ and $j$ in the graph $G^{t-1}$. As shown in the matrix in Figure 3.2, each $r_{ij}^{t-1}$ specifies both the label and the direction of the relation ($id_{label}$ for $i \rightarrow j$ versus $id_{label} + |L|$ for $i \leftarrow j$, where $|L|$ is the number of dependency labels), or specifies NONE (as 0). $\boldsymbol{W_2^L} \in R^{(2|L|+1)\times d}$ are the learned relation embeddings. The attention weights $\alpha_{ij}$ are a Softmax applied to the attention function:

$$\begin{aligned}\alpha_{ij} &= \frac{\exp(e_{ij})}{\sum \exp(e_{ij})} \\ e_{ij} &= \frac{(x_i \boldsymbol{W^Q})(x_j \boldsymbol{W^K} + \mathrm{LN}(r_{ij}^{t-1}\boldsymbol{W_1^L}))}{\sqrt{d}}\end{aligned} \tag{3.5}$$

where $\boldsymbol{W_1^L} \in R^{(2|L|+1)\times d}$ are different learned relation embeddings. LN($\cdot$) is the layer normalisation function, used for better convergence.

Equations (3.4) and (3.5) constitute the mechanism by which each iteration of refinement can condition on the previous graph. Instead of the more common approach of hard-coding some attention heads to represent a relation (e.g. Ji et al. (2019)), all attention heads can learn for themselves how to use the information about relations.

### 3.3.2 Decoder

The decoder uses the token embeddings $Z^t$ produced by the encoder to predict the new graph $G^t$. It consists of two components, a scoring function, and a decoding algorithm. The graph found by the decoding algorithm is the output graph $G^t$ of the decoder. Here we propose components for dependency parsing.

#### Scoring Function

We first produce four distinct vectors for each token embedding $z_i^t$ from the encoder by passing it through four feed-forward layers.

$$\begin{aligned}
z_i^{t,(arc-dep)} &= \mathrm{MLP}^{(arc-dep)}(z_i^t) \\
z_i^{t,(arc-head)} &= \mathrm{MLP}^{(arc-head)}(z_i^t) \\
z_i^{t,(rel-dep)} &= \mathrm{MLP}^{(rel-dep)}(z_i^t) \\
z_i^{t,(rel-head)} &= \mathrm{MLP}^{(rel-head)}(z_i^t)
\end{aligned} \tag{3.6}$$

where the MLP's are all one-layer feed-forward networks with LeakyReLU activation functions.

These token embeddings are used to compute probabilities for every possible dependency relation, both unlabelled and labelled, similarly to Dozat and Manning (2016). The distribution of the unlabelled dependency graph is estimated using, for each token $i$, a Biaffine classifier over possible heads $j$ applied to $z_i^{t,(arc-dep)}$ and $z_j^{t,(arc-head)}$. Then for each pair $i, j$, the distribution over labels given an unlabelled dependency relation is estimated using a Biaffine classifier applied to $z_i^{t,(rel-dep)}$ and $z_j^{t,(rel-head)}$.

#### Decoding Algorithms

The scoring function estimates a distribution over graphs, but the RNGTr architecture requires the decoder to output a single graph $G^t$. Choosing this graph is complicated by the fact that the scoring function is non-autoregressive. Thus the estimate consists of multiple independent components, and thus there is no guarantee that every graph in this distribution is a valid dependency graph.

We take two approaches to this problem, one for intermediate parses $G^t$ and one for the final dependency parse $G^T$. To speed up each refinement iteration, we ignore this problem for intermediate dependency graphs. We build these graphs by simply applying argmax independently to find the head of each node. This may result in graphs with loops, which are not trees, but this does not seem to cause problems for later refinement iterations.[2] For the final output dependency tree, we use the maximum spanning tree algorithm, specifically the Chu-Liu/Edmonds algorithm (Chi, 1999; Edmonds, 1967), to find the highest scoring valid dependency tree. This is necessary to avoid problems when running the evaluation scripts. The asymptotic complexity of the full model is determined by the complexity of this algorithm.[3]

### 3.3.3 Training

The RNG Transformer model is trained separately on each refinement iteration. Standard gradient descent techniques are used, with cross-entropy loss for each edge prediction. Error is not backpropagated across iterations of refinement, because no continuous values are being passed from one iteration to another, only a discrete dependency tree.

[2]We leave to future work the investigation of different decoding strategies that keep both speed and well-formedness for the intermediate predicted graphs.

[3]The Tarjan variation (Karger et al., 1995) of Chu-Liu/Edmonds algorithm computes the highest-scoring tree in $O(n^2)$ for dense graphs, which is the case here.

**Stopping Criterion.** In the RNG Transformer architecture, the refinement of the predicted graph can be done an arbitrary number of times, since the same encoder and decoder parameters are used at each iteration. In the experiments below, we place a limit on the maximum number of iterations. But sometimes the model converges to an output graph before this limit is reached, simply copying this graph during later iterations. During training, to avoid multiple iterations where the model is trained to simply copy the input graph, the refinement iterations are stopped if the new predicted dependency graph is the same as the input graph. At test time, we also stop computation in this case, but the output of the model is not affected.

## 3.4 Initial Parsers

The RNGTr architecture requires a graph $G^0$ to initialise the iterative refinement. We consider several initial parsers to produce this graph. To leverage previous work on dependency parsing and provide a controlled comparison to the state-of-the-art, we use parsing models from the recent literature as both baselines and initial parsers. To evaluate the importance of the initial parse, we also consider a setting where the initial parse is empty, so the first complete dependency tree is predicted by the RNGTr model itself. Finally, the success of our RNGTr dependency parsing model leads us to propose an initial parsing model with the same design, so that we can control for the parser design in measuring the importance of the RNG Transformer's iterative refinement.

**SynTr model.** We call this initial parser the Syntactic Transformer (SynTr) model. It is the same as one iteration of the RNGTr model shown in Figure 3.1 and defined in Section 3.3, except that there is no graph input to the encoder. Analogously to (3.1), $G^0$ is computed as:

$$\begin{cases} Z^0 = \mathrm{E}^{\mathrm{SYNTR}}(W, P) \\ G^0 = \mathrm{D}^{\mathrm{SYNTR}}(Z^0) \end{cases} \tag{3.7}$$

where $\mathrm{E}^{\mathrm{SYNTR}}$ and $\mathrm{D}^{\mathrm{SYNTR}}$ are the SynTr encoder and decoder, respectively. For the encoder, we use the Transformer architecture and initialise with pre-trained parameters of BERT. The token embeddings of the final layer are used for $Z^0$. For the decoder, we use the same scoring function as described in Section 3.3.2, and apply Chu-Liu/Edmonds decoding algorithm (Chi, 1999; Edmonds, 1967) to find the highest scoring tree.

This SynTr parsing model is very similar to the UDify parsing model proposed by Kondratyuk and Straka (2019). One difference which seems to be important for the results reported in Section 3.6.2 is in the way BERT token segmentation is handled. When BERT segments a word into sub-words, UDify seems only to encode the first segment, whereas SynTr encodes all segments and only decodes with the first segment, as discussed in Section 3.5.3. Also, UDify decodes with an attention-based mixture of encoder layers, whereas SynTr only uses the last layer.

## 3.5 Experimental Setup

### 3.5.1 Datasets

To evaluate our models, we apply them on several kinds of datasets, namely Universal Dependency (UD) Treebanks, Penn Treebanks, and the German CoNLL 2009 Treebank. For our evaluation on Universal Dependency Treebanks (UD v2.3) (Nivre et al., 2018), we select languages based on the criteria proposed in de Lhoneux et al. (2017), and adapted by Smith et al. (2018). This set contains several languages with different language families, scripts, character set sizes, morphological complexity, and training sizes and domains. For our evaluation of Penn Treebanks, we use the English and Chinese Penn Treebanks (Xue et al., 2002; Marcus et al., 1993). For English, we use the same setting as defined in Chapter 2. For Chinese, we apply the same setup as described in Chen and Manning (2014), including the use of gold PoS tags. For our evaluation on the German Treebank of the CoNLL 2009 shared task (Hajič et al., 2009), we apply the same setup as defined in Kuncoro et al. (2016). Following Hajič et al. (2009); Nivre et al. (2018), we keep punctuation for evaluation on the UD Treebanks and the German corpus and remove it for the Penn Treebanks (Nilsson and Nivre, 2008).

### 3.5.2 Baseline Models

For UD Treebanks, we compare to several baseline parsing models. We use the monolingual parser proposed by Kulmizev et al. (2019), which uses BERT (Devlin et al., 2019) and ELMo (Peters et al., 2018) embeddings as additional input features. In addition, we compare to the multilingual multi-task models proposed by Kondratyuk and Straka (2019) and Straka (2018). UDify (Kondratyuk and Straka, 2019) is a multilingual multi-task model. UDPipe (Straka, 2018) is one of the winners of CoNLL 2018 Shared Task (Zeman et al., 2018). For a fair comparison, we report the scores of UDPipe from Kondratyuk and Straka (2019) using gold segmentation. UDify is on average the best performing of these baseline models, so we use it as one of our initial parsers in the RNGTr model.

For Penn Treebanks and the German CoNLL 2009 corpus, we compare our models with previous state-of-the-art transition-based, and graph-based models, including the Biaffine parser (Dozat and Manning, 2016), which includes the same decoder as our model. We also use the Biaffine parser as an initial parser for the RNGTr model.

### 3.5.3 Implementation Details

The encoder is initialised with pre-trained BERT (Devlin et al., 2019) models with 12 self-attention layers. All hyper-parameters are provided in Appendix 3.8.

Since the wordpiece tokeniser (Wu et al., 2016) of BERT differs from that used in the dependency corpora, we apply the BERT tokeniser to each corpus word and input all the resulting sub-words

| Model | UAS | LAS |
|---|---|---|
| SynTr | 75.62 | 70.04 |
| SynTr+RNGTr (T=1) | 76.37 | 70.67 |
| SynTr+RNGTr (T=3) w/o *stop* | 76.33 | 70.61 |
| SynTr+RNGTr (T=3) | 76.29 | 70.84 |
| UDify Kondratyuk and Straka (2019) | 72.78 | 65.48 |
| UDify+RNGTr (T=1) | 74.13 | 68.60 |
| UDify+RNGTr (T=3) w/o *stop* | 75.68 | 70.32 |
| UDify+RNGTr (T=3) | 75.91 | 70.66 |

Table 3.1: Dependency parsing scores for different variations of the RNG Transformer model on the development set of UD Turkish Treebank (IMST).

| Language | Train Size | Mono [1] | Multi UDPipe | Multi UDify | Multi+Mono UDify+RNGTr | Mono SynTr | Mono SynTr+RNGTr | Mono Empty+RNGTr |
|---|---|---|---|---|---|---|---|---|
| Arabic | 6.1K | 81.8 | 82.94 | 82.88 | **85.93** (+17.81%) | **86.23** | **86.31** (+0.58%) | **86.05** |
| Basque | 5.4K | 79.8 | 82.86 | 80.97 | 87.55 (+34.57%) | 87.49 | **88.2** (+5.68%) | **87.96** |
| Chinese | 4K | 83.4 | 80.5 | 83.75 | 89.05 (+32.62%) | 89.53 | **90.48** (+9.08%) | 89.82 |
| English | 12.5K | 87.6 | 86.97 | 88.5 | 91.23 (+23.74%) | **91.41** | **91.52** (+1.28%) | 91.23 |
| Finnish | 12.2K | 83.9 | 87.46 | 82.03 | **91.87** (+54.76%) | **91.80** | **91.92** (+1.46%) | **91.78** |
| Hebrew | 5.2K | 85.9 | 86.86 | 88.11 | 90.80 (+22.62%) | **91.07** | **91.32** (+2.79%) | 90.56 |
| Hindi | 13.3K | 90.8 | 91.83 | 91.46 | 93.94 (+29.04%) | 93.95 | **94.21** (+4.3%) | 93.97 |
| Italian | 13.1K | 91.7 | 91.54 | 93.69 | 94.65 (+15.21%) | **95.08** | **95.16** (+1.62%) | **94.96** |
| Japanese | 7.1K | 92.1 | 93.73 | 92.08 | **95.41** (+42.06%) | **95.66** | **95.71** (+1.16%) | **95.56** |
| Korean | 4.4K | 84.2 | 84.24 | 74.26 | **89.12** (+57.73%) | **89.29** | **89.45** (+1.5%) | **89.1** |
| Russian | 48.8K | 91.0 | 92.32 | 93.13 | **94.51** (+20.09%) | **94.60** | **94.47** (-2.4%) | 94.31 |
| Swedish | 4.3K | 86.9 | 86.61 | 89.03 | 92.02 (+27.26%) | 92.03 | **92.46** (+5.4%) | **92.40** |
| Turkish | 3.7K | 64.9 | 67.56 | 67.44 | 72.07 (+14.22%) | **72.52** | **73.08** (+2.04%) | 71.99 |
| Average | - | 84.9 | 85.81 | 85.18 | 89.86 | 90.05 | 90.33 | 89.98 |

Table 3.2: Labelled attachment scores on UD Treebanks for monolingual ([1] (Kulmizev et al., 2019) and SynTr) and multilingual (UDPipe (Straka, 2018) and UDify (Kondratyuk and Straka, 2019)) baselines, and the refined models (+RNGTr) pre-trained with BERT (Devlin et al., 2019). The relative error reduction from RNGTr refinement is shown in parentheses. Bold scores are not significantly different from the best score in that row (with $\alpha = 0.01$).

to the encoder. For the input of dependency relations, each dependency between two words is specified as a relationship between their first sub-words. We also input a new relationship between each non-first sub-word and its associated first sub-word as its head. For the prediction of dependency relations, only the encoder embedding of the first sub-word of each word is used by the decoder.[4] The decoder predicts each dependency as a relation between the first sub-words of the corresponding words. Finally, for proper evaluation, we map the predicted sub-word heads and dependents to their original word positions in the corpus.

[4]In preliminary experiments, we found that predicting dependencies using the first sub-words achieves better or similar results compared to using the last sub-word or all sub-words of each word.

| Model | Type | English (PTB) UAS | English (PTB) LAS | Chinese (CTB) UAS | Chinese (CTB) LAS | German (CoNLL) UAS | German (CoNLL) LAS |
|---|---|---|---|---|---|---|---|
| Chen and Manning (2014) | T | 91.8 | 89.6 | 83.9 | 82.4 | - | - |
| Dyer et al. (2015) | T | 93.1 | 90.9 | 87.2 | 85.7 | - | - |
| Ballesteros et al. (2016) | T | 93.56 | 91.42 | 87.65 | 86.21 | 88.83 | 86.10 |
| Cross and Huang (2016) | T | 93.42 | 91.36 | 86.35 | 85.71 | - | - |
| Weiss et al. (2015) | T | 94.26 | 92.41 | - | - | - | - |
| Andor et al. (2016) | T | 94.61 | 92.79 | - | - | 90.91 | 89.15 |
| G2GTr (defined in Chapter 2) | T | 96.11 | 94.33 | - | - | - | - |
| Ma et al. (2018) | T | 95.87 | 94.19 | 90.59 | 89.29 | 93.65 | 92.11 |
| Fernández-González and Gómez-Rodríguez (2019) | T | 96.04 | 94.43 | - | - | - | - |
| Kiperwasser and Goldberg (2016) | G | 93.1 | 91.0 | 86.6 | 85.1 | - | - |
| Wang and Chang (2016) | G | 94.08 | 91.82 | 87.55 | 86.23 | - | - |
| Cheng et al. (2016) | G | 94.10 | 91.49 | 88.1 | 85.7 | - | - |
| Kuncoro et al. (2016) | G | 94.26 | 92.06 | 88.87 | 87.30 | 91.60 | 89.24 |
| Ma and Hovy (2017) | G | 94.88 | 92.98 | 89.05 | 87.74 | 92.58 | 90.54 |
| Ji et al. (2019) | G | 95.97 | 94.31 | - | - | - | - |
| Li et al. (2020)+ELMo | G | 96.37 | 94.57 | 90.51 | 89.45 | - | - |
| Li et al. (2020)+BERT | G | 96.44 | 94.63 | 90.89 | 89.73 | - | - |
| Biaffine Dozat and Manning (2016) | G | 95.74 | 94.08 | 89.30 | 88.23 | 93.46 | 91.44 |
| Biaffine+RNGTr | G | 96.44 | 94.71 | 91.85 | 90.12 | 94.68 | 93.30 |
| SynTr | G | **96.60** | **94.94** | 92.42 | 90.67 | **95.11** | **93.98** |
| SynTr+RNGTr | G | **96.66** | **95.01** | **92.98** | **91.18** | **95.28** | **94.02** |

Table 3.3: Comparison of our models to previous SOTA models on English (PTB) and Chinese (CTB5.1) Penn Treebanks, and German CoNLL 2009 shared task treebank. "T" and "G" specify "Transition-based" and "Graph-based" models. Bold scores are not significantly different from the best score in that column (with $\alpha = 0.01$).

## 3.6 Results and Discussion

After some initial experiments to determine the maximum number of refinement iterations, we report the performance of the RNG Transformer model on the UD treebanks, Penn treebanks, and German CoNLL 2009 treebank.[5] The RNGTr models perform substantially better than previously proposed models on every dataset, and RNGTr refinement improves over its initial parser for almost every dataset. We also perform various analyses to understand these results better.

### 3.6.1 The Number of Refinement Iterations

Before conducting a large number of experiments, we investigate how many iterations of refinement are useful, given the computational costs of additional iterations. We evaluate different variations of our RNG Transformer model on the Turkish Treebank (Table 3.1).[6] We use both SynTr and UDify as initial parsers. The SynTr model significantly outperforms the UDify model, so the errors are harder to correct by adding the RNGTr model (2.67% for SynTr versus 15.01%

[5]The number of parameters and run times of each model on the UD and Penn Treebanks are provided in Appendix 3.9.

[6]We choose the Turkish Treebank because it is a low-resource Treebank and there are more errors in the initial parse for RNGTr to correct.

for UDify of relative error reduction in LAS after integration). In both cases, three iterations of refinement achieve more improvement than one iteration, but not by a large enough margin to suggest the need for additional iterations. The further analysis reported in Section 3.6.5 supports the conclusion that, in general, additional iteration would neither help nor hurt accuracy. The results in Table 3.1 also show that it is better to include the stopping strategy described in Section 3.3.3. In subsequent experiments, we use three refinement iterations with the stopping strategy, unless mentioned otherwise.

### 3.6.2 UD Treebank Results

Results for the UD treebanks are reported in Table 3.2. We compare our models with previous state-of-the-art results (both trained mono-lingually and multi-lingually), based on labelled attachment score.[7]

The results with RNGTr refinement demonstrate the effectiveness of the RNGTr model at refining an initial dependency graph. First, the UDify+RNGTr model achieves significantly better LAS performance than the UDify model in all languages. Second, although the SynTr model significantly outperforms previous state-of-the-art models on all these UD Treebanks,[8] the SynTr+RNGTr model achieves further significant improvement over SynTr in four languages, and no significant degradation in any language. Of the nine languages where there is no significant difference between SynTr and SynTr+RNGTr for the given test sets, RNGTr refinement results in higher LAS in eight languages and lower LAS in only one (Russian).

The improvement of SynTr+RNGTr over SynTr is particularly interesting because it is a controlled demonstration of the effectiveness of the graph refinement method of RNGTr. The only difference between the SynTr model and the final iteration of the SynTr+RNGTr model is the graph inputs from the previous iteration (Equations (3.7) versus (3.1)). By conditioning on the full dependency graph, the SynTr+RNGTr model's final RNGTr iteration can capture any kind of correlation in the dependency graph, including both global and between-edge correlations both locally and over long distances. This result also further demonstrates the generality and effectiveness of the G2GTr architecture for conditioning on graphs (Equations (3.4) and (3.5)).

As expected, we get more improvement when combining the RNGTr model with UDify, because UDify's initial dependency graph contains more incorrect dependency relations for RNGTr to correct. But after refinement, there is surprisingly little difference between the performance of the UDify+RNGTr and SynTr+RNGTr models, suggesting that RNGTr is powerful enough to correct any initial parse. To investigate the power of the RNGTr architecture to correct any initial parse,

[7]Unlabelled attachment scores are provided in Appendix 3.10. All results are computed with the official CoNLL 2018 shared task evaluation script (`https://universaldependencies.org/conll18/evaluation.html`).

[8]In particular, SynTr significantly outperforms UDify, even though they are very similar models. In addition to the model differences discussed in Section 3.4, there are some differences in the way UDify and SynTr models are trained that might explain this improvement, in particular, that UDify is a multi-lingual multi-task model, whereas SynTr is a mono-lingual single-task model.

we also show results for a model with an empty initial parse, Empty+RNGTr. For this model, we run four iterations of refinement (T=4), so that the amount of computation is the same as for SynTr+RNGTr. The Empty+RNGTr model achieves competitive results with the UDify+RNGTr model (i.e. above the previous state-of-the-art), and close to the results for SynTr+RNGTr. This accuracy is achieved despite the fact that the Empty+RNGTr model has half as many parameters as the UDify+RNGtr model and the SynTr+RNGTr model since it has no separate initial parser. These Empty+RNGTr results indicate that RNGTr architecture is a very powerful method for graph refinement.

### 3.6.3 Penn Treebank and German corpus Results

UAS and LAS results for the Penn Treebanks and German CoNLL 2009 Treebank are reported in Table 3.3. We compare to the results of previous state-of-the-art models and SynTr, and we use the RNGTr model to refine both the Biaffine parser (Dozat and Manning, 2016) and SynTr, on all Treebanks.[9]

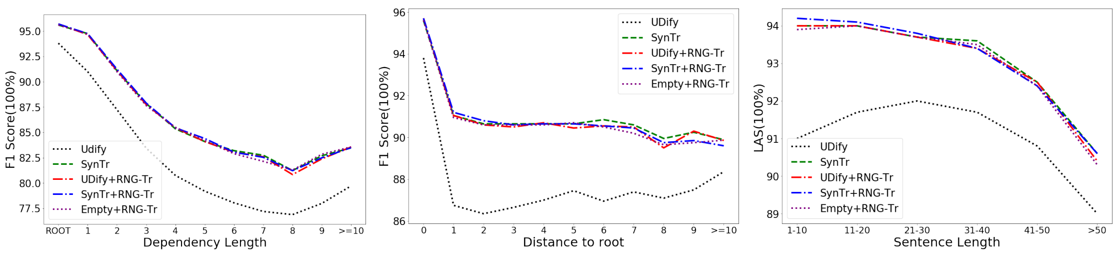


Figure 3.3: Error analysis, on the concatenation of UD Treebanks, of initial parsers (UDify and SynTr), their integration with the RNGTr model, and the Empty+RNGTr model.

Again, the SynTr model significantly outperforms previous state-of-the-art models, with a 5.78%, 9.15%, and 23.7% LAS relative error reduction in English, Chinese, and German, respectively. Despite this level of accuracy, adding RNGTr refinement improves accuracy further under both UAS and LAS. For the Chinese Treebank, this improvement is significant, with a 5.46% LAS relative error reduction. When RNGTr refinement is applied to the output of the Biaffine parser (Dozat and Manning, 2016), it achieves a LAS relative error reduction of 10.64% for the English Treebank, 16.05% for the Chinese Treebank, and 27.72% for the German Treebank. These improvements, even over such strong initial parsers, again demonstrate the effectiveness of the RNGTr architecture for graph refinement.

### 3.6.4 Error Analysis

To better understand the distribution of errors for our models, we follow McDonald and Nivre (2011) and plot labelled attachment scores as a function of dependency length, sentence length and

[9]Results are calculated with the official evaluation script: (`https://depparse.uvt.nl/`). For German, we use `https://ufal.mff.cuni.cz/conll2009-st/eval-data.html`.

distance to root.[10] We compare the distributions of errors made by the UDify (Kondratyuk and Straka, 2019), SynTr, and refined models (UDify+RNGTr, SynTr+RNGTr, and Empty+RNGTr). Figure 3.3 shows the accuracies of the different models on the concatenation of all development sets of UD Treebanks. Results show that applying RNGTr refinement to the UDify model results in a substantial improvement in accuracy across the full range of values in all cases, and little difference in the error profile between the better performing models. In all the plots, the gains from RNGTr refinement are more pronounced for the more difficult cases, where a larger or more global view of the structure is beneficial.

As shown in the leftmost plot of Figure 3.3, adding RNGTr refinement to UDify results in particular gains for the longer dependencies, which are more likely to interact with other dependencies. The middle plot illustrates the accuracy of models as a function of the distance to the root of the dependency tree, which is calculated as the number of dependency relations from the dependent to the root. When we add RNGTr refinement to the UDify parser, we get particular gains for the problematic middle depths, which are neither the root nor leaves. Here, SynTr+RNGTr is also particularly strong on these high nodes, whereas SynTr is particularly strong on low nodes. In the plot by sentence length, the larger improvements from adding RNGTr refinement (both to UDify and SynTr) are for the shorter sentences, which are surprisingly difficult for UDify. Presumably, these shorter sentences tend to be more idiosyncratic, which is better handled with a global view of the structure. (See Figure 3.5 for an example.) In all these cases, the ability of RNGTr to capture any kind of correlation in the dependency graph gives the model a larger and more global view of the correct output structure.

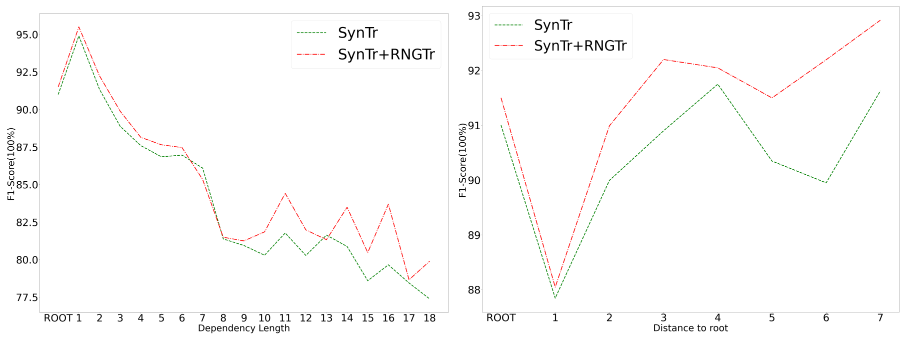


Figure 3.4: Error analysis of SynTr and SynTr+RNGTr models on Chinese CTB Treebank.

To further analyse where RNGTr refinement is resulting in improvements, we compare the error profiles of the SynTr and SynTr+RNGTr models on the Chinese Penn Treebank, where adding RNGTr refinement to SynTr results in significant improvement (see Table 3.3). As shown in Figure 3.4, RNGTr refinement results in particular improvement on longer dependencies (left plot), and on middle and greater depth nodes (right plot), again showing that RNGTr does particularly well on the difficult cases with more interactions with other dependencies.

[10]We use the MaltEval tool (Nilsson and Nivre, 2008) for calculating accuracies in all cases.

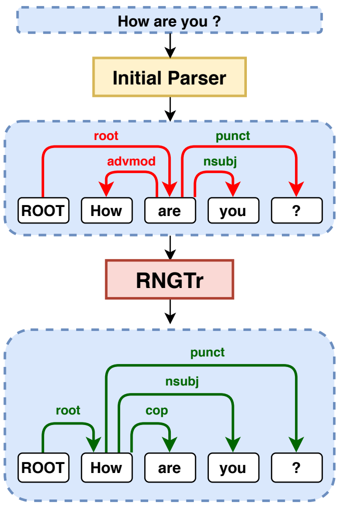


Figure 3.5: The shortest example corrected by UDify+RNGTr in the English UD Treebank.

| Dataset Type | $t = 1$ | $t = 2$ | $t = 3$ |
|---|---|---|---|
| Low-Resource | +13.62% | +17.74% | +0.16% |
| High-Resource | +29.38% | +0.81% | +0.41% |

Table 3.4: Refinement Analysis (LAS relative error reduction) of the UDify+RNGTr model for different refinement steps on the development sets of UD Treebanks.

### 3.6.5 Refinement Analysis

To better understand how the RNG Transformer model is doing refinement, we perform several analyses of the trained UDify+RNGTr model.[11] An example of this refinement is shown in Figure 3.5, where the UDify model predicts an incorrect dependency graph, but the RNGTr model modifies it to build the gold dependency tree.

**Refinements by Iteration.** To measure the accuracy gained from refinement at different iterations, we define the following metric:

$$\mathtt{REL}^t = \mathtt{RER}(\mathtt{LAS}^{t-1}, \mathtt{LAS}^t) \tag{3.8}$$

where $\mathtt{RER}$ is relative error reduction, and $t$ is the refinement iteration. $\mathtt{LAS}^0$ is the accuracy of the initial parser, UDify in this case.

To illustrate the refinement procedure for different dataset types, we split UD Treebanks based on their training set size into "Low-Resource" and "High-Resource" datasets.[12] Table 3.4 shows the

[11]We choose UDify as the initial parser because the RNGTr model makes more changes to the parses of UDify than SynTr, so we can more easily analyse these changes. Results with SynTr as the initial parser are provided in Appendix 3.11.

[12]We consider languages that have training data more than 10k sentences as "High-Resource".

| Dependency Type | $t = 1$ | $t = 2$ | $t = 3$ |
|---|---|---|---|
| goeswith | +57.83% | +0.00% | +2.61% |
| aux | +66.04% | +3.04% | +3.12% |
| cop | +48.17% | +2.21% | +3.01% |
| mark | +44.97% | +2.44% | +0.00% |
| amod | +45.58% | +2.33% | +0.00% |
| det | +34.48% | +0.00% | +2.63% |
| acl | +33.01% | +0.89% | +0.00% |
| xcomp | +33.33% | +0.80% | +0.00% |
| nummod | +28.50% | +0.00% | +1.43% |
| advcl | +29.53% | +1.26% | +0.25% |
| dep | +22.48% | +2.02% | +0.37% |

Table 3.5: Relative F-score error reduction of a selection of dependency types for each refinement step on the concatenation of UD Treebanks (with UDify as the initial parser).

| Tree Type | $t = 1$ | $t = 2$ | $t = 3$ |
|---|---|---|---|
| Non-Projective | +22.43% | +3.92% | +0.77% |
| Projective | +29.6% | +1.13% | +0.0% |

Table 3.6: Relative F-score error reduction of projective and non-projective trees on the concatenation of UD Treebanks (with UDify as the initial parser).

refinement metric ($\texttt{REL}^t$) after each refinement iteration of the UDify+RNGTr model on these sets of UD Treebanks.[13] Every refinement step achieves an increase in accuracy, on both low and high resource languages. But the amount of improvement generally decreases for higher refinement iterations. Interestingly, for languages with less training data, the model cannot learn to make all corrections in a single step but can learn to make the remaining corrections in a second step, resulting in approximately the same total percentage of errors corrected as for high resource languages. In general, different numbers of iterations may be necessary for different datasets, allowing efficiency gains by not performing unnecessary refinement iterations.

**Dependency Type Refinement.** Table 3.5 shows the relative improvement of different dependency types for the UDify+RNGTr model at each refinement step, ranked and selected by the total relative error reduction. A huge amount of improvements is achieved for all these dependency types at the first iteration step, and then we have a considerable further improvement for many of the remaining refinement steps. The later refinement steps are particularly useful for idiosyncratic dependencies which require a more global view of the sentence, such as auxiliary (aux) and copula (cop). A similar pattern of improvements is found when SynTr is used as the initial parser, reported in Appendix 3.11.

[13]For these results we apply MST decoding after every iteration, to allow proper evaluation of the intermediate graphs.

**Refinement by Projectivity.** Table 3.6 shows the relative improvement of each refinement step for projective and non-projective trees. Although the total gain is slightly higher for projective trees, non-projective trees require more iterations to achieve the best results. Presumably, this is because non-projective trees have more complex non-local interactions between dependencies, which requires more refinement iterations to fix incorrect dependencies. This seems to contradict the common belief that non-projective parsing is better done with factorised graph-based models, which do not model these interactions.

## 3.7 Conclusion

In this chapter, we propose a novel model for structured prediction, Recursive Non-autoregressive Graph-to-Graph Transformer (RNG Transformer), to iteratively refine arbitrary graphs. Given an initial graph, RNG Transformer learns to predict a corrected graph over the same set of nodes. Each iteration of refinement predicts the edges of the graph in a non-autoregressive fashion, but conditions these predictions on the entire graph from the previous iteration. This graph conditioning and prediction are made with the Graph-to-Graph Transformer architecture, which can capture complex patterns of interdependencies between graph edges and can exploit BERT (Devlin et al., 2019) pre-training.

We evaluate the RNG Transformer architecture by applying it to the problematic structured prediction task of syntactic dependency parsing. In the process, we also propose a graph-based dependency parser (SynTr), which is the same as one iteration of our RNG Transformer model but without graph inputs. Evaluating on 13 languages of the Universal Dependencies Treebanks, the English and Chinese Penn Treebanks, and the German CoNLL 2009 shared task treebank, our SynTr model already significantly outperforms previous state-of-the-art models on all these treebanks. Even with this powerful initial parser, RNG Transformer refinement almost always improves accuracies, setting new state-of-the-art accuracies for all treebanks. RNG Transformer consistently results in improvement regardless of the initial parser, reaching around the same level of accuracy even when it is given an empty initial parse, demonstrating the power of this iterative refinement method. Error analysis suggests that RNG Transformer refinement is particularly useful for complex interdependencies in the output structure.

The RNG Transformer architecture is a very general and powerful method for structured prediction, which could easily be applied to other NLP tasks. It would especially benefit tasks that require capturing complex structured interdependencies between graph edges, without losing the computational benefits of a non-autoregressive model.

# Appendix

## 3.8 Implementation Details

For better convergence, we use two different optimisers for pre-trained parameters and randomly initialised parameters. We apply bucketed batching, grouping sentences by their lengths into the same batch to speed up the training. Early stopping (based on LAS) is used during training. We use "bert-multilingual-cased" for UD Treebanks.[14] For English Penn Treebank, we use "bert-base-uncased", and for Chinese Penn Treebank, we use "bert-base-chinese". We apply pre-trained weights of "bert-base-german-cased" (Wolf et al., 2019) for German CoNLL shared task 2009. Here is the list of hyper-parameters for RNG Transformer model:

| Component | Specification |
|---|---|
| **Optimiser** | BertAdam |
| Base Learning rate | 2e-3 |
| BERT Learning rate | 1e-5 |
| Adam Betas($b_1$,$b_2$) | (0.9,0.999) |
| Adam Epsilon | 1e-5 |
| Weight Decay | 0.01 |
| Max-Grad-Norm | 1 |
| Warm-up | 0.01 |
| **Self-Attention** | |
| No. Layers | 12 |
| No. Heads | 12 |
| Embedding size | 768 |
| Max Position Embedding | 512 |

| Component | Specification |
|---|---|
| **Feed-Forward layers (arc)** | |
| No. Layers | 2 |
| Hidden size | 500 |
| Drop-out | 0.33 |
| Negative Slope | 0.1 |
| **Feed-Forward layers (rel)** | |
| No. Layers | 2 |
| Hidden size | 100 |
| Drop-out | 0.33 |
| Negative Slope | 0.1 |
| Epoch | 200 |
| Patience | 100 |

Table 3.7: Hyper-parameters for training on all Treebanks. We stop training, if there is no improvement in the current epoch, and the number of the current epoch is bigger than the summation of last checkpoint and "Patience".

[14] `https://github.com/google-research/bert`. For Chinese and Japanese, we use pre-trained "bert-base-chinese" and "bert-base-japanese" models (Wolf et al., 2019) respectively.

## 3.9 Number of Parameters and Run Time Details:

We provide average run times and the number of parameters of each model on English Penn Treebanks, and English UD Treebank. All experiments are computed with a graphics processing unit (GPU), specifically the NVIDIA V100 model.

| Model | No. parameters | Training time (HH:MM:SS) | Evaluation time (seconds) |
|---|---|---|---|
| Biaffine (Dozat and Manning, 2016) | 13.5M | 4:39:18 | 3.1 |
| RNGTr | 206.3M | 24:10:40 | 20.6 |
| SynTr | 206.2M | 6:56:40 | 7.5 |

Table 3.8: Run time details of our models on English Penn Treebank.

| Model | Training time (HH:MM:SS) | Evaluation time (seconds) |
|---|---|---|
| UDify (Kondratyuk and Straka, 2019) | 2:22:47 | 4.0 |
| RNGTr | 8:14:26 | 13.6 |
| SynTr | 1:29:43 | 3.7 |

Table 3.9: Run time details of our models on English UD Treebank.

## 3.10 Unlabelled Attachment Scores for UD Treebanks

| Language | Multi UDPipe | Multi UDify | Multi+Mono UDify+RNGTr | Mono SynTr | Mono SynTr+RNGTr | Mono Empty+RNGTr |
|---|---|---|---|---|---|---|
| Arabic | 87.54 | 87.72 | **89.73**(+16.37%) | **89.89** | **89.94**(+0.49%) | **89.68** |
| Basque | 86.11 | 84.94 | 90.49(+36.85%) | 90.46 | **90.90**(+4.61%) | **90.69** |
| Chinese | 84.64 | 87.93 | 91.04(+25.76%) | 91.38 | **92.47**(+12.64%) | 91.81 |
| English | 89.63 | 90.96 | 92.81(+20.46%) | **92.92** | **93.08**(+2.26%) | 92.77 |
| Finnish | 89.88 | 86.42 | **93.49**(+52.06%) | **93.52** | **93.55**(+0.47%) | **93.36** |
| Hebrew | 89.70 | 91.63 | 93.03(+16.73%) | **93.36** | **93.36**(0.0%) | 92.80 |
| Hindi | 94.85 | 95.13 | 96.44(+26.9%) | 96.33 | **96.56**(+6.27%) | 96.37 |
| Italian | 93.49 | 95.54 | 95.72(+4.04%) | **96.03** | **96.10**(+1.76%) | **95.98** |
| Japanese | 95.06 | 94.37 | **96.25**(+33.40%) | **96.43** | **96.54**(+3.08%) | **96.37** |
| Korean | 87.70 | 82.74 | **91.32**(+49.71%) | **91.35** | **91.49**(+1.62%) | **91.28** |
| Russian | 93.80 | 94.83 | **95.54**(+13.73%) | **95.53** | **95.47**(-1.34%) | **95.38** |
| Swedish | 89.63 | 91.91 | 93.72(+22.37%) | 93.79 | **94.14**(+5,64%) | **94.14** |
| Turkish | 74.19 | 74.56 | 77.74(+12.5%) | **77.98** | **78.50**(+2.37%) | 77.49 |
| Average | 88.94 | 89.13 | 92.10 | 92.23 | 92.46 | 92.16 |

Table 3.10: Unlabelled attachment scores on UD Treebanks for monolingual (SynTr) and multilingual (UDPipe (Straka, 2018) and UDify (Kondratyuk and Straka, 2019)) baselines, and the refined models (+RNGTr), pre-trained with BERT (Devlin et al., 2019). Bold scores are not significantly different from the best score in that row (with $\alpha = 0.01$).

## 3.11 SynTr Refinement Analysis

| Dependency Type | $t = 1$ | $t = 2$ | $t = 3$ |
|---|---|---|---|
| `clf` | +17.60% | +0.00% | +0.00% |
| `discourse` | +9.70% | +0.00% | +0.00% |
| `aux` | +3.57% | +3.71% | +0.00% |
| `case` | +2.78% | +2.86% | +0.00% |
| `root` | +2.27% | +2.33% | +0.00% |
| `nummod` | +2.68% | +1.38% | +0.00% |
| `acl` | +3.74% | +0.29% | +0.00% |
| `orphan` | +1.98% | +1.24% | +0.00% |
| `dep` | +1.99% | +0.80% | +0.00% |
| `cop` | +1.55% | +0.78% | +0.00% |
| `advcl` | +1.98% | +0.25% | +0.00% |
| `nsubj` | +1.07% | +0.54% | +0.00% |

Table 3.11: Relative F-score error reduction, when SynTr is the initial parser, of different dependency types for each refinement step on the concatenation of UD Treebanks, ranked and selected by the total relative error reduction.

| Dataset Type | $t = 1$ | $t = 2$ | $t = 3$ |
|---|---|---|---|
| Low-Resource | 2.46% | 0.09% | 0.08% |
| High-Resource | 0.81% | 0.80% | 0.32% |

(a)

| Tree type | $t = 1$ | $t = 2$ | $t = 3$ |
|---|---|---|---|
| Non-Projective | 5% | 1.63% | 0.13% |
| Projective | 0.6% | 0.61% | 0.13% |

(b)

Table 3.12: Refinement analysis of the SynTr+RNGTr model for different refinement steps. (a) Relative LAS error reduction on the low-resource and high-resource subsets of UD Treebanks. (b) Relative F-score error reduction of projective and non-projective trees on the concatenation of UD Treebanks.

# 4 Syntax-Aware Graph-to-Graph Transformer for Semantic Role Labelling

Recent models have shown that incorporating syntactic knowledge into the semantic role labelling (SRL) task leads to a significant improvement. In this chapter, we propose Syntax-aware Graph-to-Graph Transformer (SynG2G-Tr) model, which encodes the syntactic structure using a novel way to input graph relations as embeddings, directly into the self-attention mechanism of Transformer. This approach adds a soft bias towards attention patterns that follow the syntactic structure but also allows the model to use this information to learn alternative patterns. We evaluate our model on both span-based and dependency-based SRL datasets, and outperform previous alternative methods in both in-domain and out-of-domain settings, on CoNLL 2005 and CoNLL 2009 datasets.

## 4.1 Introduction

The task of semantic role labelling (SRL) provides a shallow representation of the semantics in a sentence, and constructs event properties and relations among relevant words. Traditionally, a syntactic structure was considered a prerequisite for SRL models (Punyakanok et al., 2008; Gildea and Palmer, 2002), but, newer models that leverage deep neural network architectures (Cai et al., 2018; Tan et al., 2017; He et al., 2017; Marcheggiani et al., 2017) have outperformed syntax-aware architectures, without the need for explicit encoding of syntactic structure.

However, recent studies (Zhou et al., 2020a; Strubell et al., 2018; Swayamdipta et al., 2018; He et al., 2017; Marcheggiani and Titov, 2017) have proposed that deep neural network models could benefit from using syntactic information, rather than disregarding it. These studies suggest that incorporating syntax into the model can improve SRL prediction by jointly learning both syntactic and semantic structures (Zhou et al., 2020a), training a self-attention head in Transformer (Vaswani et al., 2017) to attend to each token's syntactic parent (Strubell et al., 2018), or encoding the syntactic structure using graph convolutional networks (Fei et al., 2021; Marcheggiani and Titov, 2017).
In this chapter, we propose a novel method for encoding syntactic knowledge by introducing

Syntax-aware Graph-to-Graph Transformer (SynG2G-Tr) architecture. The model conditions on the sentence's dependency structure and jointly predicts both span-based and dependency-based SRL structures. Following Chapter 2 and Chapter 3, the model inputs graph relations as embeddings incorporated into the self-attention mechanism of Transformer (Vaswani et al., 2017). Different from G2GTr and RNGTr, the self-attention function models the interaction of the graph relations with both the query and key vectors of self-attention mechanism, instead of just the query. We also find that excluding the interaction of graph structure with the value vectors of self-attention does not harm the performance. Furthermore, the architecture uses different types of graphs as the input and output.
We show empirically that our model outperforms previous comparable models. In the in-domain setting, SynG2G-Tr model achieves 88.93 (87.57) F1 score on the CoNLL 2005 dataset, given the predicate (end-to-end), and 91.23 (88.05) F1 on the CoNLL 2009 dataset, given the predicate (end-to-end). In the out-of-domain setting, our model reaches 83.21 (80.53) F1 score on the CoNLL 2005 dataset, given predicate (end-to-end), and 86.43 (81.93) F1 scores on the CoNLL 2009 dataset, given predicate (end-to-end). Our contributions of this chapter are:

- We propose SynG2G-Tr model for encoding the dependency parsing graph in the SRL.
- We evaluate our model on CoNLL 2005 and CoNLL 2009 datasets and outperforms previous comparable models in both in-domain and out-of-domain sets.

## 4.2 Syntax-aware Graph-to-Graph Transformer

The architecture of the SynG2G-Tr model is illustrated in Figure 4.1. The input to the model is the tokenised text ($W = (w_1, w_2, ..., w_N)$), which are the nodes of the input and output graphs, and $N$ is the length of tokenised input. The outputs are the dependency-based ($G_{dep}$) and span-based ($G_{span}$) SRL graphs. The SynG2G-Tr model can be formalised in terms of an encoder $E^{sg2g}$ and decoder $D^{sg2g}$:

$$\begin{cases} Z = \mathrm{E}^{\mathrm{sg2g}}(W, P, G_{syn}) \\ G_{span}, G_{dep} = \mathrm{D}^{\mathrm{sg2g}}(Z) \end{cases} \tag{4.1}$$

Initially, a syntactic parser predicts the dependency graph ($G_{syn}$), and Part-of-Speech (PoS) tags ($P = (p_1, p_2, ..., p_N)$). Then the encoder of SynG2G-Tr ($E^{sg2g}$) encodes both sequences ($W, P$) and the dependency graph ($G_{syn}$) into contextualised representations of graph nodes ($Z$). This representation ($Z$) is then used by the decoder ($D^{sg2g}$) to jointly predict SRL graphs. For the decoder, we follow the same unified scorer and decoder as defined in Zhou et al. (2020a). Further explanation of SRL scorer and decoding mechanism is provided in Appendix 4.6.

The encoder employs an enhanced way of inputting graph relations into the self-attention mechanism of Transformer (Vaswani et al., 2017). Unlike the proposed Graph-to-Graph Transformer (defined in Chapter 2), we modify the self-attention mechanism to have a more comprehensive interaction between graph relations, queries and keys. We also find that excluding the interaction of graph relations with value vectors retains good performance.

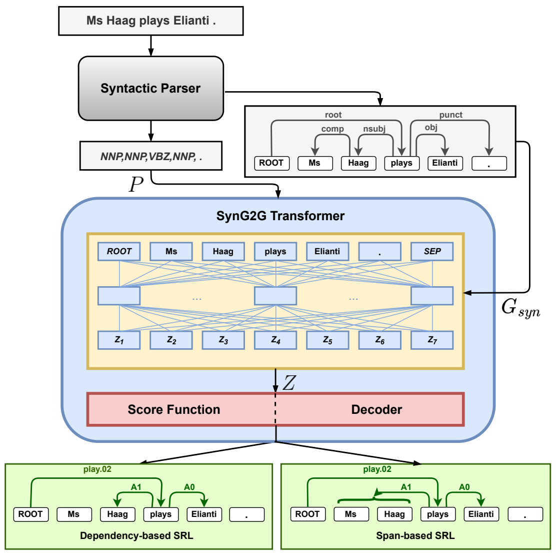


Figure 4.1: The architecture of SynG2G-Tr.

Specifically, given the output of an intermediate embedding layer $X = (x_1, ..., x_N)$, we define the attention mechanism of each head in each layer to take the dependency graph as input. These attention scores ($\alpha_{ij}$) are calculated as a Softmax function over $e_{ij}$ values:

$$e_{ij} = \frac{1}{\sqrt{d}}\Big[x_i \boldsymbol{W^Q}(x_j \boldsymbol{W^K})^T + x_i \boldsymbol{W^Q}(r_{ij}\boldsymbol{W^R})^T + r_{ij}\boldsymbol{W^R}(x_j \boldsymbol{W^K})^T\Big] \tag{4.2}$$

where $\boldsymbol{W^Q}, \boldsymbol{W^K} \in \mathbb{R}^{d_x \times d}$ are learned query and key matrices. $r_{ij} \in R$ is a one-hot vector specifying both the label and direction of the dependency relation between token $i$ and token $j$. [1] $R$ is the matrix of graph relations, derived from the syntactic graph ($G_{syn}$). $\boldsymbol{W^R} \in \mathbb{R}^{(2|L_{syn}|+1)\times d}$ is a matrix of learned relation embeddings. $d$ is the attention head size, and $d_x$ is the hidden size. A sample computation of $R$ matrix from $G_{syn}$ is provided in Appendix 4.7. The second and third terms in Equation 4.2 incorporate the graph information into the self-attention mechanism of Transformer with a soft bias, while the model can still learn other structures, using this encoded graph information. For better efficiency, we share the relation embeddings across multiple attention heads in each layer.
The output of the attention function is the value embedding ($v_i$), which is calculated as:

$$v_i = \sum_j \alpha_{ij}(x_j \boldsymbol{W^V}) \tag{4.3}$$

which, in our model, does not use the graph, and $\boldsymbol{W^V} \in \mathbb{R}^{d_x \times d}$ is the learned value matrix.

**Syntactic Parser.** The dependency parser jointly predicts a sequence of PoS tags and the dependency graph. Specifically, we apply the BERT-based syntactic parser defined in Zhou et al.

[1] $id_{label}$ if $i \rightarrow j$, $id_{label} + |L_{syn}|$ if $j \leftarrow i$ , or NONE; where $|L_{syn}|$ is the label set size.

(2020a), which uses a joint scorer and decoder for dependency and constituency graphs based on Head-driven Phrase Structure Grammar (HPSG) (Zhou and Zhao, 2019). This method has achieved state-of-the-art results in the dependency parsing task. More details can be found in Zhou et al. (2020a).

## 4.3 Related Work

Several approaches have been proposed to improve the performance of SRL models using syntactic information. Roth and Lapata (2016) embed dependency paths, while some researchers (Fei et al., 2021; Munir et al., 2021; Marcheggiani and Titov, 2017) use graph convolutional networks to encode the syntactic structure. Strubell et al. (2018) incorporates a dependency graph by training one attention head of Transformer to attend to syntactic parents for each token, in a multi-task setting. He et al. (2019, 2018b) use syntactic information to guide the argument pruning. Xia et al. (2019) exploit different alternatives e.g. tree-structured GRU and graph features of dependency tree to encode syntactic knowledge. Kasai et al. (2019) apply BiLSTM to tag the text with supertags extracted from dependency parses and feed them into SRL models. Xia et al. (2020) showed that incorporating heterogeneous syntactic knowledge results in significant improvement. Some other work focus on joint learning of both SRL and syntax (Zhou et al., 2020a,b; Cai and Lapata, 2019a,b). Additionally, some approaches discarded the syntax, but achieve impressive results (Shi and Lin, 2019; Peters et al., 2018; He et al., 2018a; Marcheggiani et al., 2017; He et al., 2017; Tan et al., 2017; Zhou and Xu, 2015). Our work is different from previous work since we encode the syntactic graph by directly inputting it as embeddings into the attention mechanism of Transformer, which provides a soft bias. Moreover, both sequences (e.g. tokens and PoS tags) and syntactic graph can be encoded in one general model.

## 4.4 Results and Discussion

**Experimental Setup.** Our models are evaluated on CoNLL 2005 (Carreras and Màrquez, 2005) and CoNLL 2009 (Hajič et al., 2009).[2] For predicate disambiguation, we follow previous work (Marcheggiani and Titov, 2017), and use an off-the-shelf disambiguator from Roth and Lapata (2016). As in previous work, we evaluate in both *end-to-end*, and *given predicate* settings. For a more accurate comparison, we train SynG2G-Tr both with and without BERT initialisation (SynG2G-Tr w/o BERT). The discrepancy between BERT tokenisation and the tokenisation used in the SRL corpora is handled as in Chapter 2. [3] For the syntactic parser, we use the same hyper-parameters as defined in Zhou et al. (2020a). The performance of the syntactic parser is provided in Appendix 4.10.

[2]Implementation details of datasets and SynG2G-Tr model are provided in Appendices 4.8 and 4.9.

[3]For inputting the dependency graph, the relation between two sub-words of different words is defined as the same dependency relation between their corresponding words in the sentence. This means that the relation $(i, j, l_{in})$ is repeated for each sub-word of word $x_i$, and word $x_j$. The same strategy is applied to predicted PoS tags.

| Model | SA | WSJ (in-domain) | | | Brown (out-of-domain) | | |
|---|---|---|---|---|---|---|---|
| | | P | R | F1 | P | R | F1 |
| **end-to-end** | | | | | | | |
| He et al. (2017) | ✗ | 85.0 | 84.3 | 84.6 | 74.9 | 72.4 | 73.6 |
| He et al. (2018a) | ✗ | 81.2 | 83.9 | 82.5 | 69.7 | 71.9 | 70.8 |
| Strubell et al. (2018) | ✓ | 85.53 | 84.45 | <u>84.99</u> | 75.8 | 73.54 | <u>74.66</u> |
| Li et al. (2019) | ✗ | - | - | 83.0 | - | - | - |
| Xia et al. (2019) | ✓ | 84.3 | 83.8 | 84.1 | 73.7 | 72.0 | 72.9 |
| Xia et al. (2020) | ✓ | 83.05 | 84.49 | 84.49 | 73.47 | 74.92 | 74.19 |
| SynG2G-Tr (w/o BERT) | ✓ | 84.48 | 86.46 | **85.45** | 73.92 | 76.65 | **75.26** |
| *+pre-training* | | | | | | | |
| He et al. (2018a) | ✗ | 84.8 | 87.2 | 86.0 | 73.9 | 78.4 | 76.1 |
| Strubell et al. (2018)† | ✓ | 87.13 | 86.67 | <u>86.9</u> | 79.02 | 77.49 | <u>78.25</u> |
| Li et al. (2019) | ✗ | 85.2 | 87.5 | 86.3 | 74.7 | 78.1 | 76.4 |
| SynG2G-Tr | ✓ | 86.86 | 88.3 | **87.57** | 80.01 | 81.07 | **80.53** |
| **given predicate** | | | | | | | |
| Tan et al. (2017) | ✗ | 84.5 | 85.2 | 84.8 | 73.5 | 74.6 | 74.1 |
| He et al. (2018a) | ✗ | - | - | 83.9 | - | - | 73.7 |
| Strubell et al. (2018)† | ✓ | 86.02 | 86.05 | <u>86.04</u> | 76.65 | 76.44 | <u>76.54</u> |
| Ouchi et al. (2018) | ✗ | 84.7 | 82.3 | 83.5 | 76.0 | 70.4 | 73.1 |
| Xia et al. (2020) | ✓ | 85.12 | 85.0 | 85.06 | 76.3 | 75.42 | 75.86 |
| SynG2G-Tr (w/o BERT) | ✓ | 86.46 | 86.56 | **86.50** | 77.73 | 77.18 | **77.45** |
| *+pre-training* | | | | | | | |
| He et al. (2018a) | ✗ | - | - | 87.4 | - | - | 80.4 |
| Ouchi et al. (2018) | ✗ | 88.2 | 87.0 | 87.6 | 79.9 | 77.5 | 78.7 |
| Li et al. (2019) | ✗ | 87.9 | 87.5 | 87.7 | 80.6 | 80.4 | 80.5 |
| Jindal et al. (2020) | ✗ | 87.70 | 88.15 | 87.93 | 81.52 | 81.36 | 81.44 |
| Zhang et al. (2021b) | ✗ | 88.70 | 88.00 | 87.90 | 80.30 | 80.10 | 80.20 |
| Jia et al. (2022) | ✗ | - | - | <u>88.25</u> | - | - | <u>81.90</u> |
| SynG2G-Tr | ✓ | 89.11 | 88.74 | **88.93** | 83.93 | 82.50 | **83.21** |

Table 4.1: Comparing our SynG2G-Tr with previous comparable models on CoNLL 2005 test sets. 'SA' means a syntax-aware model. Scores being boldfaced means that they are significantly better than the second best model, specified by the underline marker.

**CoNLL 2005 Results.**[4] The results for span-based SRL are shown in Table 4.1.[5] Without BERT initialisation, our SynG2G-Tr model outperforms Strubell et al. (2018) (the second best model) in both *end-to-end* and *given-predicate* settings. This highlights the benefit of injecting the graph information into the self-attention mechanism using a soft bias, instead of hard-coding one attention head to attend to the syntactic parent of each token, as used in Strubell et al. (2018). The main reason for this improvement is that the model can still learn other attention patterns in combination with the graph information, which will be described later in this section. When adding BERT initialisation, our SynG2G-Tr model outperforms best previous work by 5.4%/8.8% F1 relative error reduction (RER) on average in both in-domain and out-of-domain evaluation sets, which demonstrates the compatibility of the modified self-attention mechanism of SynG2G-Tr

[4]Results are calculated with official evaluation scripts of CoNLL 2005 (`https://www.cs.upc.edu/~srlconll`).

[5]For a fair comparison, we excluded Li et al. (2021b); Zhou et al. (2020a), as they use information from the constituency graph additional to the dependency tree. Also, to better understand the effect of syntactic information, we exclude Fernández-González (2023); Zhou et al. (2022); Conia and Navigli (2020), as they exploited different scorer and training mechanism for SRL graphs. However, the best setting of SynG2G-Tr model still shows competitive or better results when compared to aforementioned excluded works.

| Model | SA | WSJ (in-domain) | | | Brown (out-of-domain) | | |
|---|---|---|---|---|---|---|---|
| | | P | R | F1 | P | R | F1 |
| **end-to-end** | | | | | | | |
| He et al. (2018b) | ✓ | 83.9 | 82.7 | 83.3 | - | - | - |
| Cai et al. (2018) | ✗ | 84.7 | 85.2 | 85.0 | - | - | 72.5 |
| Li et al. (2019) | ✗ | - | - | 85.1 | - | - | - |
| SynG2G-Tr (w/o BERT) | ✓ | 84.10 | 87.07 | **85.59** | 73.66 | 72.56 | **73.11** |
| *+pre-training* | | | | | | | |
| Li et al. (2019) | ✗ | 84.5 | 86.1 | 85.3 | 74.6 | 73.8 | 74.2 |
| SynG2G-Tr | ✓ | 86.38 | 89.78 | **88.05** | 80.35 | 83.57 | **81.93** |
| **given predicate** | | | | | | | |
| Marcheggiani et al. (2017) | ✗ | 88.7 | 86.8 | 87.7 | 79.4 | 76.2 | 77.7 |
| M&T. 2017 | ✓ | 89.1 | 86.8 | 88.0 | 78.5 | 75.9 | 77.2 |
| He et al. (2018b) | ✓ | 89.7 | 89.3 | 89.5 | 81.9 | 76.9 | 79.3 |
| Cai et al. (2018) | ✗ | 89.9 | 89.2 | 89.6 | 79.8 | 78.3 | 79.0 |
| Cai and Lapata (2019b) | ✓ | 90.5 | 88.6 | 89.6 | 80.5 | 78.2 | 79.4 |
| Kasai et al. (2019) | ✓ | 89.0 | 88.2 | 88.6 | 78.0 | 77.2 | 77.6 |
| SynG2G-Tr (w/o BERT) | ✓ | 89.78 | 90.28 | **90.03** | 81.32 | 82.15 | **81.73** |
| *+pre-training* | | | | | | | |
| Li et al. (2019) | ✗ | 89.6 | 91.2 | 90.4 | 81.7 | 81.4 | 81.5 |
| Kasai et al. (2019) | ✓ | 90.3 | 90.0 | 90.2 | 81.0 | 80.5 | 80.8 |
| Lyu et al. (2019) | ✗ | - | - | 90.99 | - | - | 82.18 |
| Chen et al. (2019) | ✗ | 90.74 | 91.38 | **91.06** | 82.66 | 82.78 | 82.72 |
| He et al. (2019) | ✓ | 90.41 | 91.32 | 90.86 | 86.15 | 86.70 | 86.42 |
| Cai and Lapata (2019a) | ✓ | 91.1 | 90.4 | 90.7 | 82.1 | 81.3 | 81.6 |
| Munir et al. (2021) | ✓ | 91.2 | 90.6 | 90.9 | 83.1 | 82.6 | 82.8 |
| SynG2G-Tr | ✓ | 91.31 | 91.16 | **91.23** | 86.40 | 86.47 | **86.43** |
| **gold syntax** | | | | | | | |
| Fei et al. (2021) | ✓ | 92.5 | 92.5 | 92.5 | 85.6 | 85.3 | 85.4 |
| SynG2G-Tr+Gold | ✓ | 92.71 | 93.37 | **93.03** | 88.27 | 88.31 | **88.29** |

Table 4.2: Comparing our SynG2G-Tr with previous comparable models on CoNLL 2009 test sets. 'SA' means a syntax-aware model. Scores being boldfaced means that they are significantly better than the second best model, specified by the underline marker.

with BERT (Devlin et al., 2019) initialisation.

## CoNLL 2009 Results.[6]

Table 4.2 illustrates the results of dependency-based SRL on the test set of CoNLL 2009 dataset. Without BERT initialisation, SynG2G-Tr significantly outperforms previous work in in-domain and out-of-domain settings. With BERT initialisation, our model significantly outperforms previous work in all settings (except in-domain, *given-predicate*) with 10.3%/25.7% F1 RER in both in-domain and out-of-domain evaluation sets. For a better comparison with Fei et al. (2021) (last setting of Table 4.2), we also employ the gold dependency tree for training and use the predicted dependency graph at the inference time. Our model significantly outperforms Fei et al. (2021), especially on out-of-domain dataset. This shows the benefit of encoding the dependency graph by modifying the self-attention mechanism of Transformer (Vaswani et al., 2017) compared to using graph convolutional network, as in Fei et al. (2021).

[6]Scores are calculated with CoNLL 2009 shared task script (`https://ufal.mff.cuni.cz/conll2009-st/`).

**Further Analysis.** We also analyse the self-attention matrix of SynG2G-Tr model for different heads and layers. Figure 4.3 in Appendix 4.11 demonstrates that the self-attention mechanism of SynG2G-Tr ignores the dependency graph information in the first few layers, and only uses the context-dependent information. However, as it progresses to upper layers, it begins to utilise the graph relation information, as shown in the attention matrix. This highlights the benefit of encoding the dependency graph with a soft bias as the model can still learn different structures in different layers, given this encoded graph information. Furthermore, in Appendix 4.12, we show that removing the interaction of graph embeddings with key vectors results in a performance drop. Additionally, ignoring the interaction of graph relations with both key and query vectors [7] results in a significant drop as well. However, integrating the graph information into Equation 4.3 as stated in Chapter 2 does not improve the performance, and we remove it for better efficiency.

## 4.5 Conclusion

In this chapter, we propose the Syntax-aware Graph-to-Graph Transformer architecture, which effectively incorporates syntactic information by inputting the syntactic dependency graph into the self-attention mechanism of Transformer. The mechanism for inputting graph relation embeddings differs from the original Graph-to-Graph Transformer (defined in Chapter 2 and Chapter 3) in that it models the complete interaction between the dependency relation, query vector and key vector. It also excludes the graph interaction with value vectors while maintaining good performance. We have evaluated our model on CoNLL 2005 and CoNLL 2009 SRL datasets and outperformed previous comparable models. Future studies can apply our model to any NLP task which might benefit from conditioning on the syntactic structure or other graphs.

[7]This leads to a BERT-based syntax-agnostic model, similar to Shi and Lin (2019).

# Appendix

## 4.6 SRL Scorer and Decoder Details

**Scorer.** Inspired by Zhou et al. (2020a), we first define span representation ($s_{ij}$) as the difference between right and left end-points of the span:

$$s_{ij} = s\vec{r}_j - s\overleftarrow{l}_i \tag{4.4}$$

where $s\vec{r}_j$ is defined as $[z_{\vec{j}+1}; \vec{z}_j]$, and $s\overleftarrow{l}_i$ is calculated as $[\overset{\leftrightarrows}{z}_i; z_{\overleftarrow{i}+1}]$. $\overset{\leftrightarrows}{z}_i$ is computed by dividing the output representation of Transformer ($z_i$) in half.
Argument ($a_{ij}$) and predicate ($v_k$) representations are defined as:

$$\begin{aligned} a_{ij} &= \text{ReLU}(\boldsymbol{W^1_{srl}} s_{ij} + b^1_{srl}) \\ v_k &= z_k \end{aligned} \tag{4.5}$$

where $\boldsymbol{W^1_{srl}}$ and $b^1_{srl}$ are learned parameters and ReLU(.) is the Rectified Linear Unit (Nair and Hinton, 2010) function.
We predict semantic roles as defined in Zhou et al. (2020a):

$$\Phi_l(v, a) = \boldsymbol{W^3_{srl}}(\text{LN}(\boldsymbol{W^2_{srl}}[a_{ij}; v_k] + b^2_{srl})) + b^3_{srl} \tag{4.6}$$

where LN(.) is the layer normalisation (Ba et al., 2016) function, and $\boldsymbol{W^2_{srl}}$, $\boldsymbol{W^3_{srl}}$, $b^2_{srl}$, and $b^3_{srl}$ are learned parameters. The semantic role score for a specific label $l_{out}$ is defined as:

$$\Phi_l(v, a, l_{out}) = [\Phi_l(v, a)]_{l_{out}} \tag{4.7}$$

Since the number of predicate-argument pairs is $O(n^3)$, we apply the pruning method proposed in Li et al. (2019); He et al. (2018a) by defining separate scorers for argument and predicate candidates ($\Phi_a$ and $\Phi_v$), and pruning all but the top-ranked arguments and predicates based on their corresponding scores.

**Training.** The model is trained to optimise the probability $P(\hat{y}|W, P, G_{syn})$ of predicate-argument pairs, conditioned on input sequence ($W$), PoS tags ($P$), and predicated dependency graph ($G_{syn}$).

This objective can be factorised as:

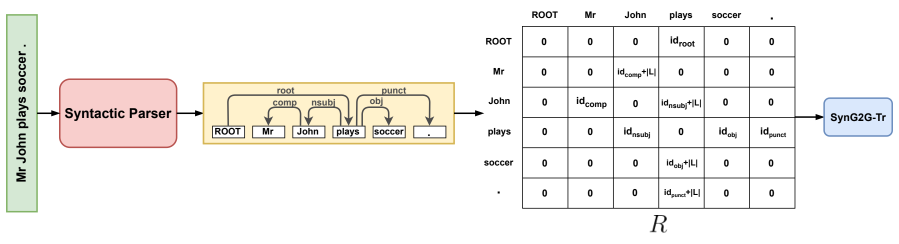
$$
\begin{aligned}
J(\theta) &= \sum_{y \in \Gamma} -log P_\theta(y|W, P, G_{syn}) \\
&= \sum_{\langle v,a,l_{out}\rangle \in \Gamma} -log \frac{\exp(\Phi(v,a,l_{out}))}{\sum_{\hat{l} \in L_{srl}} \exp(\Phi(v,a,\hat{l}))}
\end{aligned}
\tag{4.8}
$$

where $\Phi(v,a,l_{out})$ is defined as $\Phi_v(v) + \Phi_a(a) + \Phi_l(v,a,l_{out})$, and $\theta$ is model parameters. $\Gamma$ is the set of predicate-argument-relation tuples for all possible predicate-argument pairs and either the correct relation or NONE.

**Decoders.** Following Zhou et al. (2020a), we apply a single dynamic programming decoder according to the uniform score following the non-overlapping constraints (Punyakanok et al., 2008).

## 4.7 Sample Computation of Matrix of Graph Relations

Figure 4.2 illustrates an example of creating the matrix of graph relations ($R$), which is then fed to SynG2G-Tr model.



|  | ROOT | Mr | John | plays | soccer | . |
|---|---|---|---|---|---|---|
| ROOT | 0 | 0 | 0 | $id_{root}$ | 0 | 0 |
| Mr | 0 | 0 | $id_{comp}$+\|L\| | 0 | 0 | 0 |
| John | 0 | $id_{comp}$ | 0 | $id_{nsubj}$+\|L\| | 0 | 0 |
| plays | 0 | 0 | $id_{nsubj}$ | 0 | $id_{obj}$ | $id_{punct}$ |
| soccer | 0 | 0 | 0 | $id_{obj}$+\|L\| | 0 | 0 |
| . | 0 | 0 | 0 | $id_{punct}$+\|L\| | 0 | 0 |

Figure 4.2: A sample computation of $R$ matrix for the sentence "Mr John plays soccer.".

## 4.8 Implementation Details and Pre-processing Steps

**CoNLL 2005.** In this shared task (Carreras and Màrquez, 2005) (under LDC license), the focus was on verbal predicates in English. The training data includes sections 2-21 of the Wall Street Journal (WSJ) dataset. Section 24 is considered as the development set, while section 23 is used as the in-domain test set. Three sections of the Brown corpus are used for the out-of-domain dataset. The dataset can be downloaded from here, and pre-processing steps are provided in here.

**CoNLL 2009.** This shared task Hajič et al. (2009) (under LDC license) focused on dependency-based SRL and was created by merging PropBank and NomBank treebanks. We evaluate our models on the English dataset with the same split as the CoNLL 2005 dataset. The dataset and pre-processing steps can be found at here and here. The number of sentences in train and evaluation sets is as follows:

| Train | Dev | Test-WSJ | Test-Brown |
|---|---|---|---|
| 39’832 | 1’334 | 2’399 | 425 |

Table 4.3: The number of sentences for each split of CoNLL 2005 and CoNLL 2009 datasets.

## 4.9 Hyper-parameters Setting

We use *bert-large-whole-word-masking*[8] (345M parameters) for the initialisation of encoder in SynG2G-Tr model. We apply separate optimisers for pre-trained parameters and randomly initialised ones. We use bucket batching, grouping sentences by their lengths to the same batch to speed up the model. Early stopping is used to mitigate over-fitting. In a pre-defined predicate setting, we use different dynamic programming decoders to find SRL graphs, since predicates are not necessarily the same in dependency-based and span-based SRL graphs. For choosing the best hyper-parameters, we use manual tuning to find the base learning rate and BERT learning rate. For other hyper-parameters, we follow previous work (Zhou et al., 2020a). The base learning rate is selected from $\{1e-2, 1e-3, 1.5e-3\}$, and the BERT learning rate is chosen from $\{1e-5, 1.5e-5, 2e-5\}$. So, we train our models with 9 different learning rates to find the best performing model based on the summation of F1 scores of span-based and dependency-based SRL graphs. We use NVIDIA GeForce GTX 1080 Ti for training and evaluating our models. [9] For the dependency parser, we apply the same hyper-parameters as Zhou et al. (2020a). We use the base learning rate of $2e-3$, and the BERT learning rate of $1.5e-5$. Here is the list of hyper-parameters for the SynG2G-Tr model:

| Component | Specification |
|---|---|
| **Optimiser** | BertAdam |
| Base Learning rate | 1.5e-3 |
| BERT Learning rate | 1e-5 |
| Adam Betas($b_1$,$b_2$) | (0.9,0.999) |
| Adam Epsilon | 1e-5 |
| Weight Decay | 0.01 |
| Max-Grad-Norm | 1 |
| Warm-up | 0.001 |
| **Self-Attention** | |
| No. Layers | 24 |
| No. Heads | 16 |
| Embedding size | 1024 |
| Max Position Embedding | 512 |

| Component | Specification |
|---|---|
| **Feed-Forward layers (SRL)** | |
| Span Hidden size | 512 |
| Label Hidden size | 250 |
| **Feed-Forward layers (PoS)** | |
| Hidden size | 250 |
| **Pruning (SRL)** | |
| $\lambda_{verb}$ | 0.6 |
| $\lambda_{span}$ | 0.6 |
| Max No. Span | 300 |
| Max No. Verb | 30 |
| Epoch | 100 |

Table 4.4: Hyper-parameters for training SynG2G-Tr.

[8] `https://github.com/google-research/bert`. Apache License 2.0.

[9] The training time of SynG2G-Tr model is 0h20m40s, and the evaluation time is 0h02m24s.

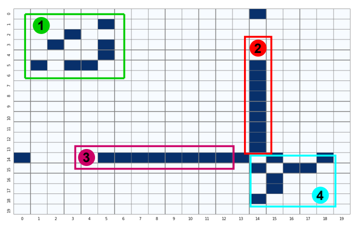
## 4.10 Performance of Syntactic Parser

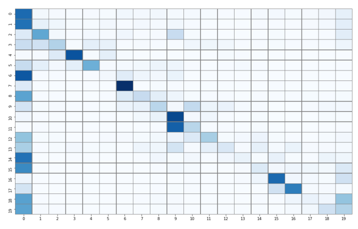
| Section | UAS | LAS | PoS |
|---|---|---|---|
| Development | 96.72 | 94.83 | 96.81 |
| Test | 96.85 | 95.24 | 97.41 |

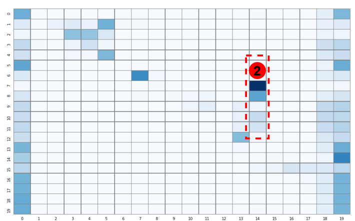
Table 4.5: Labelled and unlabelled attachment scores (LAS/UAS) and PoS accuracy. Sections 22&23 of WSJ Penn Treebanks are used as evaluation and test sets.

## 4.11 Attention Visualisation

Figure 4.3 shows the attention weights for different layers of self-attention in the SynG2G-Tr model (Figure 4.3b-4.3d), alongside the dependency relation matrix (Figure 4.3a). The self-attention matrix includes four patterns. The first layer of the SynG2G-Tr model (Figure 4.3b) ignores the graph relations and learns string-local context information. For the middle layer (Figure 4.3c), attention weights partially use the graph relation pattern. Then, in the last layer (Figure 4.3d), the dependency graph relations are evident in the attention pattern. This demonstrates the benefit of adding the graph information with a soft bias, allowing the model to learn different structures using both local context and graph information. Furthermore, it can be inferred that the last layers of the self-attention mechanism require a global view and between-edge information, while the first few layers learn local context information. More examples are provided in Figures 4.4-4.5-4.6.


(a) Graph Relation Matrix


(b) First Layer


(c) Middle Layer

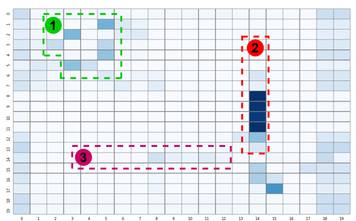
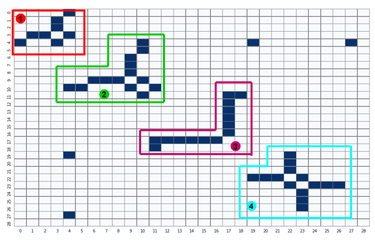
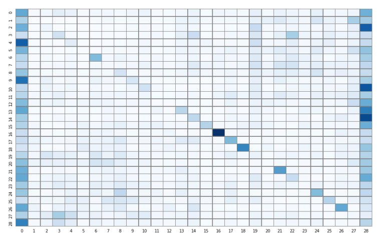
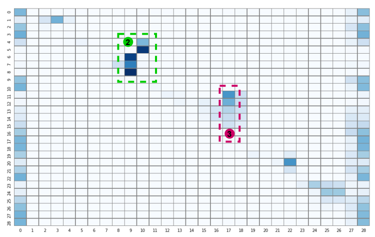

(d) Last Layer

Figure 4.3: The attention weights for the CoNLL 2009 example "`[CLS]` The most troublesome report may be the August merchandise trade deficit due out tomorrow . `[SEP]`". The first figure shows the dependency graph matrix.


(a) Graph Relation Matrix


(b) First Layer


(c) Middle Layer

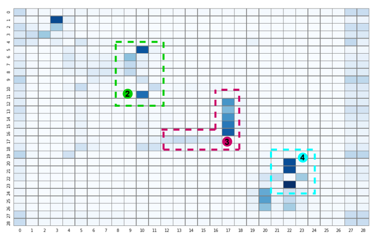
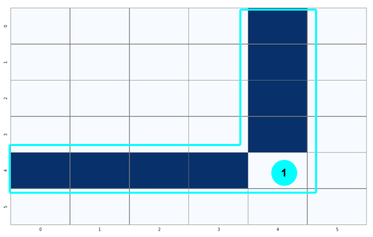
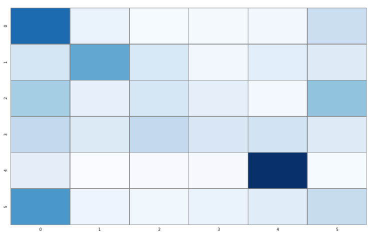

(d) Last Layer

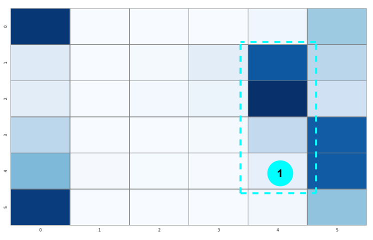
Figure 4.4: The attention weights for a specific example "`[CLS]` The consensus view expects a 0.4 % increase in the September CPI after a flat reading in August . `[SEP]`" on CoNLL 2009 dataset. The first figure shows the dependency graph matrix.

## 4.12 Ablation Study

In Table 4.6, we analyse the interaction of the dependency graph with key and query vectors in the attention mechanism, as defined in Equation 4.2. Excluding the key interaction results in


(a) Graph Relation Matrix


(b) First Layer


(c) Middle Layer

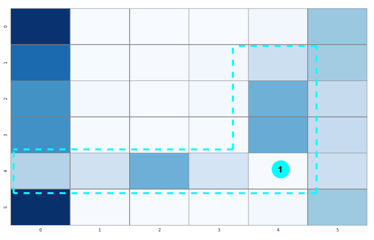
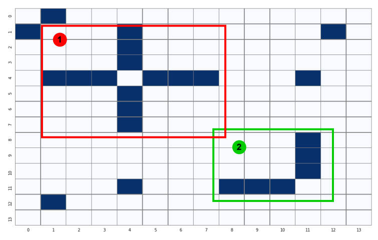
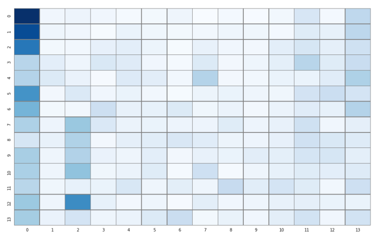
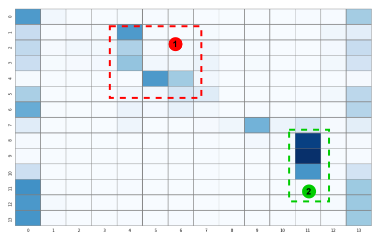

(d) Last Layer

Figure 4.5: The attention weights for a specific example "[CLS] Candid Comment [SEP]" on CoNLL 2009 dataset. The first figure shows the dependency graph matrix.


(a) Graph Relation Matrix


(b) First Layer


(c) Middle Layer

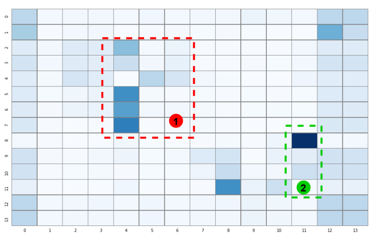

(d) Last Layer

Figure 4.6: The attention weights for a specific example "[CLS] Let 's make that 1929 , just to be sure . [SEP]" on CoNLL 2009 dataset. The first figure shows the dependency graph matrix.

a similar attention mechanism as defined in Chapter 2. This SynG2G-Tr-*key* model achieves similar results compared to the SynG2G-Tr model on the WSJ test dataset given the predicate, but the SynG2G-Tr model outperforms it on all other settings, including both types of out-of-domain datasets, confirming that key interaction is a critical part of the SynG2G-Tr model.
When both key and query interactions are excluded from the SynG2G-Tr model (SynG2G-Tr-*key-query*), it has significantly lower performance than the SynG2G-Tr model in all settings.

This demonstrates the impact of encoding the graph relation embeddings in the self-attention mechanism of Transformer (Vaswani et al., 2017) model.
We also evaluate adding the interaction of graph relations with value vectors to the SynG2G-Tr model, as defined in Chapter 2. The SynG2G-Tr+*value* model achieves similar or worse results compared to the SynG2G-Tr model. So, we exclude this interaction to speed up the modified attention mechanism.

| Model | CoNLL 2005 | | | CoNLL 2009 | | |
|---|---|---|---|---|---|---|
| | Dev | WSJ | Brown | Dev | WSJ | Brown |
| *end-to-end* | | | | | | |
| SynG2G-Tr *-key -query* | 86.65 | 87.08 | 79.40 | 86.40 | 87.26 | 81.12 |
| SynG2G-Tr *-key* | 86.82 | 87.27 | 80.33 | 86.85 | 87.50 | 81.51 |
| SynG2G-Tr | 87.08 | 87.57 | 80.53 | 87.13 | 88.05 | 81.93 |
| SynG2G-Tr +*value* | 87.17 | 87.45 | 80.40 | 86.92 | 87.95 | 82.03 |
| *given predicate* | | | | | | |
| SynG2G-Tr *-key -query* | 87.93 | 88.52 | 82.56 | 90.16 | 90.68 | 85.72 |
| SynG2G-Tr *-key* | 88.03 | 88.91 | 82.90 | 90.31 | 91.22 | 86.28 |
| SynG2G-Tr | 88.17 | 88.93 | 83.21 | 90.66 | 91.23 | 86.43 |
| SynG2G-Tr +*value* | 88.15 | 88.78 | 83.10 | 90.48 | 91.15 | 86.41 |

Table 4.6: Model comparison of SynG2G-Tr and other variants, by F1 score on CoNLL 2005 and CoNLL 2009 datasets. The SynG2G-Tr-*key-query* model is the same as the syntax-agnostic BERT model.

# 5 What Do Compressed Multilingual Machine Translation Models Forget?

Recently, very large pre-trained models achieve state-of-the-art results in various natural language processing (NLP) tasks, but their size makes it more challenging to apply them in resource-constrained environments. Compression techniques allow to drastically reduce the size of the models and therefore their inference time with negligible impact on top-tier metrics. However, the general performance averaged across multiple tasks and/or languages may hide a drastic performance drop on under-represented features, which could result in the amplification of biases encoded by the models. In this chapter, we assess the impact of compression methods on Multilingual Neural Machine Translation models (MNMT) for various language groups, gender, and semantic biases by extensive analysis of compressed models on different machine translation benchmarks, i.e. FLORES-101, MT-Gender, and DiBiMT. We show that the performance of under-represented languages drops significantly, while the average BLEU metric only slightly decreases. Interestingly, the removal of noisy memorisation with compression leads to a significant improvement for some medium-resource languages. Finally, we demonstrate that compression amplifies intrinsic gender and semantic biases, even in high-resource languages.[1]

## 5.1 Introduction

Over the recent years, pre-trained Transformer (Vaswani et al., 2017) models have reached a substantial improvement in a variety of Natural Language Processing (NLP) tasks. This improvement mostly comes from increasing their parameter size (Zhang et al., 2022; Brown et al., 2020; Fan et al., 2020a; Devlin et al., 2019) which escalates the cost of training (Patterson et al., 2021; Strubell et al., 2019; Yang et al., 2019), and hurts the memory footprint and latency at inference (Wang et al., 2022; Fan et al., 2020a; Dai et al., 2019). Specially in Neural Machine Translation (NMT) task, massively MNMT models (Zhang et al., 2020a; Aharoni et al., 2019; Fan et al., 2020a; Tang et al., 2020) demonstrated promising results. They have been shown particularly interesting for low-resource languages which benefit a lot from knowledge

[1] We release our implementation at `https://github.com/alirezamshi/bias-compressedMT`.

transfer. On the other hand, it has also been observed that the *curse of multilinguality* may hurt the performance in high-resource languages. The strategy employed to overcome this problem (Goyal et al., 2021a; Aharoni et al., 2019; Fan et al., 2020a) is to scale up the number of parameters, thus attaining state-of-the-art performance in both high and low-resource languages.

Consequently, efficient inference with these very large models has become a crucial problem. This challenge can be overcome through model compression, e.g. knowledge distillation (Li et al., 2021a; Wang et al., 2021; Sanh et al., 2019; Kim and Rush, 2016), pruning (Zhang et al., 2021a; Behnke and Heafield, 2020; Michael H. Zhu, 2018; Frankle and Carbin, 2019), and quantisation (Yao et al., 2022; Yang et al., 2022; Tao et al., 2022; Bondarenko et al., 2021; Kim et al., 2021a; Wu et al., 2020; Xu et al., 2018). These methods can be applied with a little loss in top-line metrics, while reducing the memory-footprint, and enhancing inference time. However, recent work (Ahia et al., 2021; Xu et al., 2021; Li et al., 2021a; Renduchintala et al., 2021; Hooker et al., 2020) has demonstrated that under-represented features can suffer from a drastic decrease in performance which is not necessarily reflected by global (aggregated) metrics. In multilingual NMT, the overall metrics are often reported as an average across all the language pairs, where the performance between individual language pairs can vary a lot. Therefore it is even more critical to understand what would be the exact impact of compression on multilingual NMT models, beyond the aggregated metrics.

In this chapter, we illustrate the impacts of applying compression methods to massively multilingual NMT models, that are pre-trained in a great number of languages in several domains. To the best of our knowledge, this is the first attempt to analyze how compression impacts massively multilingual models. We hope it could be a starting point to bringing a comprehensive understanding between fairness and compression in multilingual NMT models. In this study, we concentrate on *light* compression techniques, specifically post-training quantisation and magnitude pruning without any further fine-tuning.[2] We exploit the recent and largest MNMT model, M2M-100 (Fan et al., 2020a) that covers 100 languages and contains nearly 12B parameters and analyse the impact of compression on different language pairs evaluated on FLORES-101 benchmark (Goyal et al., 2021b) (covering 101 languages). We also consider MT-Gender (Stanovsky et al., 2019) and DiBiMT (Campolungo et al., 2022) benchmarks allowing us to assess different types of biases that could be present in the data and MNMT model. To sum up, our contributions of this chapter are as follows:

- We conduct extensive analysis on the effects of *light* compression methods for massively multilingual NMT models.
- On FLORES-101 (Goyal et al., 2021b), we discover that while the overall performance is barely impacted by the compression, a subset of language pairs corresponding to under-represented languages during training suffers an extreme drop in performance.
- Also, we observe an important improvement for some language pairs after the compression.

[2]The reason is that fine-tuning MNMT models is extremely computationally demanding.

We hypothesize that this is due to the removal of noisy memorisation.

- We show that the compression amplifies gender and semantic biases, hidden in MNMT models across several high-resource languages by evaluating on MT-Gender, and DiBiMT benchmarks.

In section 5.2, we describe *light* compression methods we rely on, and MNMT model. Section 5.3 presents our experimental setup and evaluation benchmarks. Section 5.4 shows the analysis of the impact of the compression for NMT benchmarks.

## 5.2 Model and Compression Techniques

### 5.2.1 M2M-100 Model

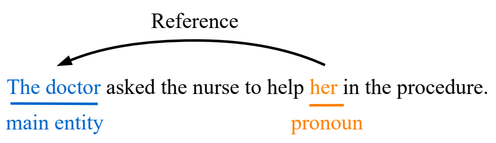
We assume that potential biases, discovered after the compression are mostly related to the training data, than the model architecture, as previous work (Hooker et al., 2020) demonstrated for the image classification task.
So, we use M2M-100 (Fan et al., 2020a), as it is the best performing and the largest massively multilingual MT model, which covers more than 10K language directions, including a great number of low and medium-resource language pairs. Other previous work (Tang et al., 2020; Aharoni et al., 2019) cover fewer languages, especially from low and medium-resource languages, and have worse results compared to M2M-100.
M2M-100 is trained on large-scale multilingual corpora (Schwenk et al., 2021; El-Kishky et al., 2020) with a novel data mining procedure, that uses language similarities. The biggest model introduced consists of 24 encoder, and 24 decoder Transformer (Vaswani et al., 2017) layers. Using several scaling techniques, it is trained with nearly 12B parameters. We refer to Fan et al. (2020a) for more details. In all our experiments, we exploit the largest M2M-100 model.

### 5.2.2 Light Compression Techniques

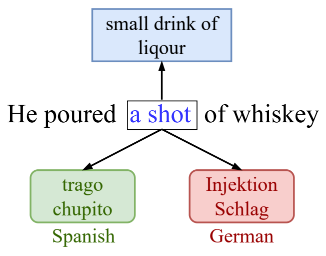
Compression techniques without any further fine-tuning are defined as *light* compression methods. We do not fine-tune the compressed models due to the massive computation cost, as we have to fine-tune the model for all language pairs to provide a fair comparison. [3] We discuss our methods in the following paragraphs.

**Magnitude Pruning.** is a popular technique for both memory footprint reduction and inference speed-up. It reduces the model size by removing redundant nodes that do not contribute to the resulting performance. It usually achieves comparable results with state-of-the-art models with further fine-tuning (Menghani, 2021; Ahia et al., 2021; Michael H. Zhu, 2018; Gale et al., 2019).

[3]Additionally, the exact and original training data is required to alleviate the additional bias added by fine-tuning, but M2M-100 authors do not provide the exact data e.g. back-translation.



(a) MT-Gender example: for a correct translation, system will have to link English pronoun 'her' to 'doctor'.



(b) DiBiMT Example. German instance contains wrong word senses, while Spanish one is correct.

Figure 5.1: Samples of MT-Gender (Stanovsky et al., 2019) and DiBiMT (Campolungo et al., 2022) benchmarks.

In this work, we apply post-training magnitude pruning for each layer of Transformer (including Embedding layers). Given $\Theta_l$ as the parameters of Transformer layer $l$ and $p$ as the sparsity ratio, the output of the pruning function is $\Theta'_l$ where $p$ percentage of weights sets to zero.[4]

**Post-Training Quantisation.** Recent work applies post-training, and training-aware quantisation to pre-trained machine translation and language models (Wei et al., 2022b; Menghani, 2021; Liang et al., 2021; Bondarenko et al., 2021; Wu et al., 2020), and achieves promising results while lowering the inference latency, and the model size. In this work, we exploit the post-training quantisation method proposed by Wu et al. (2020), converting all weights and activations from 32-bit floating-point values to an 8-bit fixed-point integer. Specifically, it quantises linear layers input and weights, matrix multiplications, and the residual summations for Transformer (Vaswani et al., 2017).

## 5.3 Experimental Setup

### 5.3.1 Evaluation Benchmarks

We analyse our compressed models on three different NMT benchmarks. We exploit FLORES-101 (Goyal et al., 2021b) to study the model behaviour based on the amount of available resources for each language. MT-Gender (Stanovsky et al., 2019) is used to study the impact of compression on gender bias. Finally, we evaluate on DiBiMT (Campolungo et al., 2022) to illustrate the compression effect on semantic biases.

[4]Preliminary experiments showed that pruning based on Transformer layer results in a better performance than other alternatives e.g. separate pruning of self-attention and feed-forward layers. The comparison is provided in Appendix 5.7.

**FLORES-101.** is a many-to-many NMT evaluation benchmark, including sentences extracted from English Wikipedia. It is translated into 101 languages by human translators, enabling 10,100 language directions to be evaluated. We evaluate our models on `devtest` subset of the FLORES-101 (Goyal et al., 2021b) benchmark. This benchmark provides test sets comparable across all the language pairs, and thus allows us to assess to what extent each language pair gets impacted by the compression techniques.

**MT-Gender.** (Stanovsky et al., 2019) is an English-centric multilingual NMT benchmark for evaluating gender bias in multiple target languages: Arabic, Ukrainian, Hebrew, Russian, Italian, French, Spanish, and German. The method relies on automatic alignment and morphological analysis, without the need for gold translations.[5] An example is shown in Figure 5.1a. Later, Kocmi et al. (2020) extends the benchmark by adding Czech and Polish languages. We choose MT-Gender as it covers more languages compared to other existing MT gender bias benchmarks (Savoldi et al., 2022; Renduchintala et al., 2021; Bentivogli et al., 2020).

**DiBiMT.** is the first fully manually-crafted NMT benchmark for evaluating word sense disambiguation on five high-resource languages: Chinese, German, Italian, Russian, and Spanish (Campolungo et al., 2022), where the source language is English. Besides, they propose several bias evaluation metrics to compare different models (defined in Section 5.4.3). As shown in Figure 5.1b, given English source sentence, specific word ($w_i$) with associated synset ($\sigma$), and language $L$, set of `GOOD`, and `BAD` translation candidates include sentences that do and do not contain set of correct translation of $\sigma$ in language $L$, respectively. More details can be found in Campolungo et al. (2022).

### 5.3.2 Implementation Details

We use pre-trained M2M-100 12B model.[6] For quantisation, we use Mean Squared Error (MSE) calibration. For weights, we use default per-channel calibration. In FLORES-101, we use SentencePiece BLEU (spBLEU) score[7] for the evaluation, as it is shown to be fair for the multilingual comparison (Goyal et al., 2021b). Additionally, we use character $n$-gram F-score (ChrF) (Popović, 2015) [8] metric to compare compressed models with M2M-100 model. We evaluate our compressed models on language pairs in which M2M-100 12B model (Fan et al., 2020a) has reasonable[9] performance. This leaves us with 3,763 language directions. All experiments are computed on 2 NVIDIA A100-40GB GPUs.

[5]For each instance, the main entity is attached to a pronoun, and the side entity attempts to distort the translation. With the use of automatic alignment and morphological analysis, the translated gender is extracted.

[6]`https://github.com/pytorch/fairseq/tree/main/examples/m2m_100`

[7]It uses SentencePiece tokeniser with 256K tokens and then BLEU is computed: `https://github.com/facebookresearch/flores`

[8]sacrebleu 1.5.1 (Post, 2018) with ChrF3.

[9]Specifically, we choose language pairs, in which M2M-100 12B model has a spBLEU score higher than 12. More details are provided in Appendix 5.8.

| Resource Type | Criterion | No. Languages |
|---|---|---|
| Very-Low | $\|L\| \leq 100k$ | 16 |
| Low | $100k < \|L\| \leq 1M$ | 40 |
| Medium | $1M < \|L\| \leq 100M$ | 38 |
| High | $100M < \|L\|$ | 7 |

Table 5.1: Distribution of lang. in FLORES-101 based on amount of available data to/from English ($|L|$).

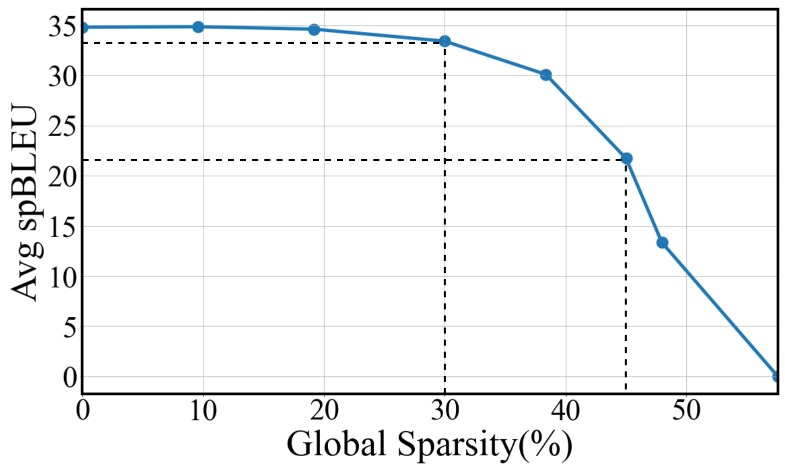


Figure 5.2: Average spBLEU score for different sparsity ratios on 9 FLORES-101 language pairs, selected from all pairwise combinations of "low", "medium", and "high" language resource categories.

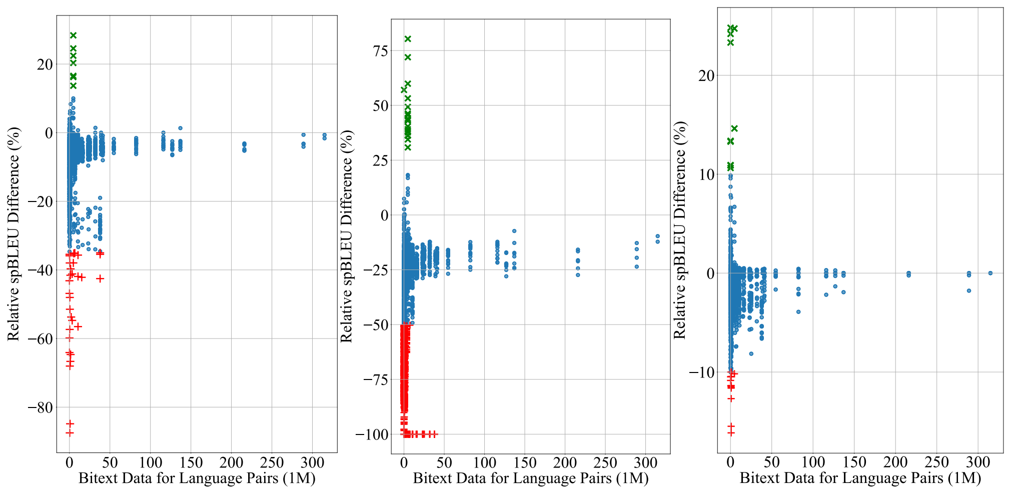
## 5.4 Results and Discussion

### 5.4.1 Compression Impact Across Languages

**Language Resource Type.** The true amount of available training data for a language is difficult to estimate, as it relies both on the quality and quantity of the data. Inspired by Goyal et al. (2021b), we classify languages into four categories, based on the amount of available data to/from English. The distribution of language resource types is illustrated in Table 5.1.

**Magnitude pruning: Sparsity Ratio ($p$) Selection.** Figure 5.2 shows the average spBLEU score of different sparsity ratios for a subset of language pairs.[10] Based on this preliminary analysis, we decide to analyse the model behaviour for two sparsity ratios, 30% which is the maximum sparsity ratio for which the compressed model mostly keeps the performance, and 45% for which the performance starts to drop drastically. Therefore, we evaluate the pruned models on sparsity ratios of 30%, and 45% for further experiments.

[10]We choose nine language pairs covering all pairwise combinations of "low", "medium", and "high" language categories. A list of this subset is provided in Appendix 5.9.



(a) Pruned 30% Model (b) Pruned 45% Model (c) Quantised Model

Figure 5.3: Relative spBLEU difference (%) between the compressed models and M2M-100 model based on the amount of available Bitext data with English ($\rho_{x,y}$). Green points ("×") are language pairs with significant improvement. Red points ("+") correspond to language pairs with a drastic performance drop.

| Model | Memory size | Avg spBLEU | drop(%) |
|---|---|---|---|
| M2M-100 | 1× | **22.44** | - |
| Pruned 30% M2M-100 | 0.7× | **20.95** | 6.6 |
| Pruned 45% M2M-100 | 0.55× | 15.12 | 32.6 |
| Quantised M2M-100 | 0.25× | **22.31** | 0.6 |

Table 5.2: Memory size and average spBLEU score of M2M-100, and compressed models on FLORES-101.

### Main Results

Table 5.2[11] illustrates memory footprint and spBLEU scores on FLORES-101 dataset averaged over 3.7k language pairs retained for analysis. Pruned 30% model suffers from a slight drop in performance, while quantisation mostly preserves the same average spBLEU score. Both quantised and pruned 30% models reduce the memory footprint by 75% and 30%, respectively. The performance of 45% pruned model drops significantly. In what follows, we check the behavior of each language pair after compression along different criteria.

**Amount of Bitext Data.** Figure 5.3 shows the relative spBLEU performance of compressed models for each language pair $(x, y)$ compared to the M2M-100. The X-axis corresponds to the amount of bitext data with English defined as $\rho_{x,y} = min(\rho_x, \rho_y)$ where $\rho_x$ is the amount of Bitext data with English for language $x$. For pruned 30% model, while the average spBLEU score drops

[11]We did not report actual inference time as implementation of compression techniques is highly dependent on the device.

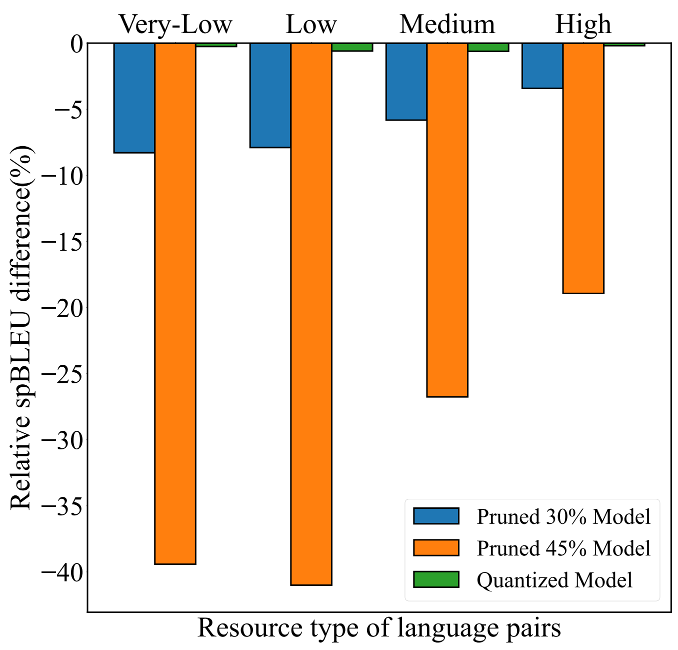


Figure 5.4: Relative spBLEU difference (%) between the compressed models and M2M-100 model grouped by the resource type of language pairs.

by 6.63% (shown in Table 5.2), there is a subset of language pairs that drops drastically (shown as "+"). Interestingly, there is a subset of language pairs that get significantly improved after compression (shown as "×"). For pruned 45% model, there is also a subset of languages with more than 50% drop in performance, while the average spBLEU degradation is 32.62%. For the quantised model which preserves almost the same average spBLEU, we see that there is also a set of languages suffering from a significant drop, and others being significantly improved. The behaviour of compressed models in these specific language pairs is further studied in Section 5.4.1 and 5.4.1, respectively.

**Resource Type.** We study the performance of the compressed models based on the resource category of language pairs, which is defined as the category of $\rho_{x,y}$ for a pair $x \rightarrow y$. Figure 5.4 demonstrates the relative spBLEU drop for each category of the compressed models. For pruning 30%, the relative spBLEU drop is inversely proportional to the amount of training data for different categories, which confirms that pruning disproportionately impacts the performance of under-represented language pairs, while the average performance is near to the base M2M-100 model (as shown in Table 5.2). For quantisation, we see a much smaller decrease in all language categories. Furthermore, we show that the resource type of the target language is more crucial than the source language,[12] meaning that the performance of language pairs with "low" and "very-low" target languages drops drastically after the compression.

[12]Results are provided in Appendix 5.10.

| Model | Off-T(%) *base* | Off-T(%) *comp* | Total No. |
|---|---|---|---|
| Pruned 30% | 5.9 | 13.7(**+7.8**) | 1,521 |
| Pruned 45% | 6.4 | 30.3(**+23.9**) | 10,314 |
| Quantised | 5.2 | 17.5(**+12.3**) | 268 |

Table 5.3: Percentage of off-target translations for M2M-100 (*base*), and compressed models (*comp*). Last column is the total number of losing sentences (both on- and off-targets) for each compressed model.

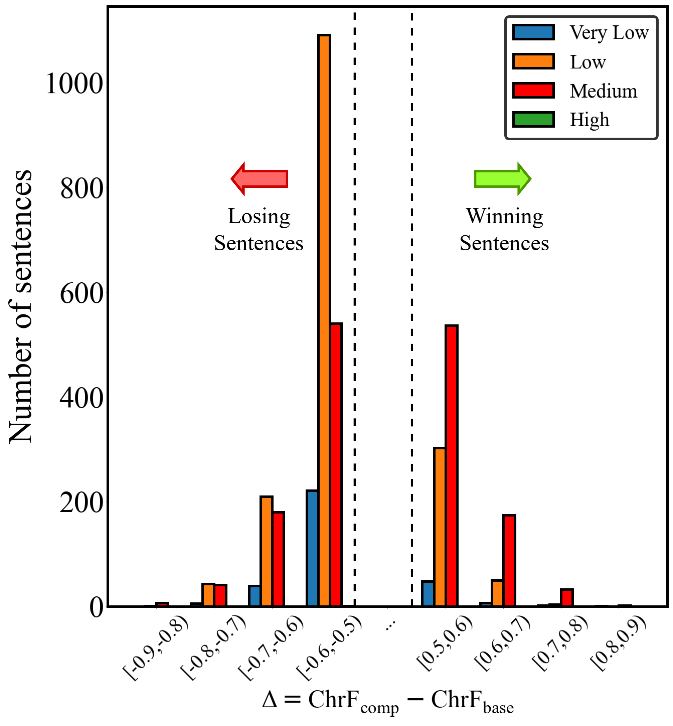


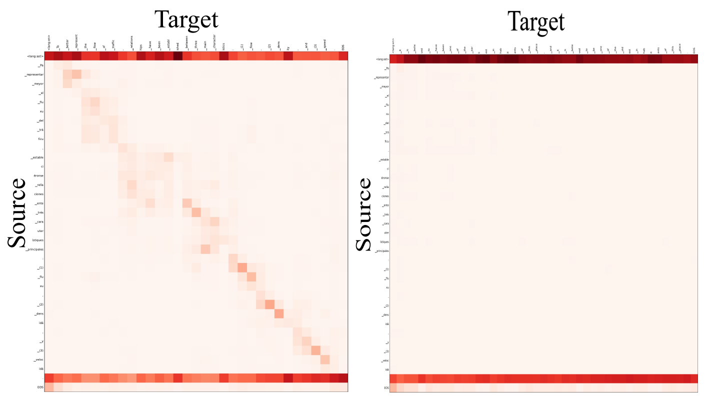
Figure 5.5: Absolute number of sentences in each language pair category for different Δ bins.

**ChrF Difference.** For more fine-grained analysis, we perform sentence-level ChrF (Popović, 2015)[13] evaluation. We define $\Delta = \text{ChrF}_{\text{comp}} - \text{ChrF}_{\text{base}}$ where $\text{ChrF}_{\text{comp}}$ and $\text{ChrF}_{\text{base}}$ correspond to ChrF of compressed and baseline models, respectively. Sentences with Δ close to zero are less impacted by compression, while those further away from zero are the most impacted (either positively or negatively) by compression. We define *Losing Pairs* as a set of instances where $\Delta < -0.5$, and *Winning Pairs* as a set of instances where $\Delta > 0.5$. Thus, identified samples could be seen as an adaptation of *Compression-Identified Exemplars* introduced by (Hooker et al., 2019) for the case of translation. Figure 5.5[14] plots the distribution of sentences from different language pair groups along with the different Δ bins for these two subsets. [15]

[13]ChrF demonstrates better correlation with human judgements at sentence-level.

[14]The normalised distribution by the number of instances in each language pair category is provided in Appendix 5.11.

[15]Figure 5.5 belongs to Pruned 30% model. Complete ChrF calculation (including $-0.5 < \Delta < 0.5$) of compressed models for different bins are provided in Appendix 5.11.



(a) M2M-100 Model (b) Compressed Model

| | |
|---|---|
| **Reference** | To better represent traffic flow, relationships have been established between the three main characteristics: (1) flow, (2) density, and (3) velocity. |
| **M2M-100** | To better represent the flow of traffic, relationships have been established between three main characteristics: (1) flow, (2) density, and (3) speed. |
| **Compressed** | It is believed to have been one of the earliest inhabitants of this place, and it is believed to be one of the oldest inhabitants of this place. |

(c) Reference and output translations of M2M-100, and compressed models.

Figure 5.6: Cross-attention matrices of an on-target losing sentence for the M2M-100 model, and pruned 30% model. Output translations show the hallucination for the compressed model. Source language is Asturian.

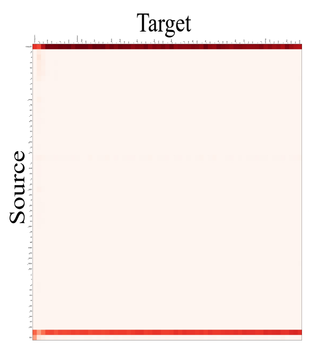
In the following, we comprehensively analyse the behaviour of the model for *Losing Pairs*, and *Winning Pairs*.[16]

### Analysis of Losing Pairs

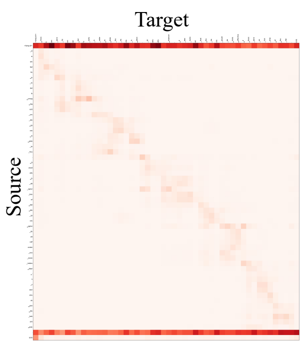
As shown in Figure 5.5 (left side), losing pairs belong to very-low, low, and medium-resource languages, that are mostly under-represented subsets during training.[17] We manually inspected some of the translations from the losing pairs sets and we have identified two main reasons for the drop in performance which are *off-target translations* (translation in the wrong target language) and *hallucinations*. In what follows we attempt to quantify these two phenomena.

[16]During the preliminary analysis we have identified languages for which M2M-100 training data contains two different scripts (e.g. Cyrillic and Latin), while FLORES-101 dataset provides one script for the evaluation. To fairly analyse the effect of compression, we exclude sentences that refer to these languages. A list of them is provided in Appendix 5.12.

[17]Normalised distribution in Appendix 5.11 follows same trend.



(a) M2M-100 Model



(b) Compressed Model

| | |
|---|---|
| **Reference** | Crossties were introduced fairly early to hold the tracks in place. Gradually, however, it was realised that tracks would be more efficient if they had a stip of iron on the top. |
| **M2M-100** | Cucumbers Zucchini Summer Squash Carrots Kale Radishes Broccoli Rosemary Basil Pole Beans Peas Arugula Bibb Lettuce Cutting Lettuces Potatoes |
| **Compressed** | Crossbars were inserted fairly early in order to keep the tracks in place. Gradually, however, it was realized that the tracks would be more effective if there were an iron strip at the top. |

(c) Reference and output translations of M2M-100, and compressed models.

Figure 5.7: Cross-attention matrices of a winning sentence for the M2M-100 model, and pruned 30% model. Output translations show the hallucination for M2M-100 model. Source language is Afrikaans.

**Off-Target.** We use FastText language identifier (Joulin et al., 2016a,b) to predict the languages of reference and the translated sentences. Table 5.3 shows the total number of losing sentences and percentage of off-target translations for both baseline and compressed models.[18] As the sparsity increases, the compressed model predicts more off-target translations (7.8% and 23.9% increase from baseline). Quantisation also increases the percentage of off-target translation by 12.3%.

**Hallucinations.** It refers to the case, in which a model generates an output unrelated to the source sentence. Lee et al. (2018a) have shown that the cases of hallucinations have different cross-attention matrices. Figure 5.6 shows an example of cross-attention matrices for a losing sentence,

[18]We exclude sentences where the predicted reference language ids are not matched with gold reference languages.

| Model | $\lambda$ | No. On-Target sents |
|---|---|---|
| Pruned 30% | 2.95 | 1,312 |
| Pruned 45% | 3.01 | 7,192 |
| Quantised | 1.96 | 221 |

Table 5.4: Total number of on-target (excluding off-target translations) sentences and relative alignment ($\lambda$) metric on losing pair subset.

where the translation of the compressed model is considered as a hallucination. As expected, translated tokens ignore the alignment with the source sequence. To quantitatively analyse the hallucination effect on all on-target losing sentences (excluding off-target translations), we define the relative alignment metric as:

$$\lambda = \frac{\mathrm{var}_{\mathrm{comp}}}{\mathrm{var}_{\mathrm{base}}} \tag{5.1}$$

where var is defined as:

$$\begin{cases} \mathrm{var} = \frac{1}{|I|.|J|} \sum_{i \in I} \sum_{j \in J} \alpha_{i,j} (\mu_i - j)^2 \\ \mu_i = \sum_{j \in J} j.\alpha_{i,j} \end{cases} \tag{5.2}$$

where $I$ and $J$ correspond to sequences of source and target languages, respectively; $\alpha_{i,j}$ is the attention weight, where we use the average attention over all layers and all attention heads. Inspired by Kim et al. (2021b); Vig and Belinkov (2019), the variance (var) is high for cases where the target sequence pays attention to a very small subset of source tokens (hallucination), while it is low when the cross-attention matrix is near to the diagonal matrix (approximation of perfect alignment matrix). Table 5.4 displays the relative alignment ($\lambda$) metric for different compressed models. As the metric is higher than "1" for compressed models, it confirms that target translations of compressed models contain more hallucinated sentences. Lastly, we provide a list of the most affected language pairs in Appendix 5.13 for further studies.

### Analysis of Winning Pairs

When manually inspecting some examples from the translation of winning pairs, we realize that a lot of them are matching cases where the baseline model generates hallucinations, while the compressed model generates acceptable translations, as shown in Figure 5.7. We recall that in Figure 5.5, most of the winning pairs (right side) belong to medium-resource languages[19], which include a moderate amount of training instances, and could contain some poorly aligned parallel sentences. Raunak et al. (2021) connects the phenomenon of hallucination to the corpus-level noise and suggests that it could also be amplified by back-translation (used for data augmentation to training M2M-100 model). Therefore, the compression seems to remove the memorisation of noisy samples, which is more important for medium-resource languages, thus fixing some of the

[19]Normalised distribution in Appendix 5.11 shows the same behaviour.

| Model | $\lambda$ | Total No. |
|---|---|---|
| Pruned 30% M2M-100 | 0.42 | 863 |
| Pruned 45% M2M-100 | 0.15 | 1,455 |
| Quantised M2M-100 | 0.52 | 308 |

Table 5.5: The relative alignment ($\lambda$) metric for different compressed models on winning pairs subset.

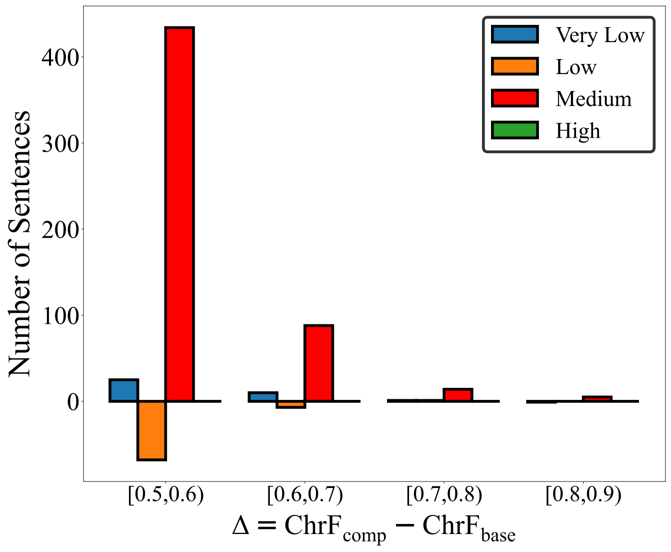


Figure 5.8: Number of sentences in winning pairs, added to each language category after increasing the sparsity from 30% to 45%.

cases of hallucination. In Table 5.5, we compute the total number of winning sentences, and the relative alignment metric ($\lambda$) for compressed models and M2M-100 model. As $\lambda$ is lower than "1", it confirms that the compression removes the noisy memorisation of medium-resource languages, and benefits the generalisation of the model. Ahia et al. (2021) made a similar observation in the case of bilingual MT models. Interestingly, the number of winning sentences increases as the model gets sparser (1,455 vs. 863). Figure 5.8 shows that new sentences mostly belong to medium-resource languages. Finally, a list of most winning language pairs is provided in Appendix 5.13.

### 5.4.2 Gender Bias Analysis

We evaluate M2M-100 and our compressed models on MT-Gender benchmark (Kocmi et al., 2020; Stanovsky et al., 2019). Inspired by Boito et al. (2022), we use a fairness metric to compare the behavior of compressed models on male and female subsets:

$$\psi = \frac{f_m - f_f}{f_m + f_f} \tag{5.3}$$

| Model | $\psi$ (%) | $\psi^*$ (%) |
|---|---|---|
| Original M2M-100 | 17.36 | 16.51 |
| Pruned 30% M2M-100 | 21.65 (**+24.7**) | 19.52 (**+18.25**) |
| Pruned 45% M2M-100 | 29.03 (**+67.2**) | 20.8 (**+25.9**) |
| Quantised M2M-100 | 18.24 (+5.1) | 15.53 (-5.8) |

Table 5.6: Average fairness metrics over languages of MT-Gender (Stanovsky et al., 2019). Numbers in parentheses are the relative score differences between a specific compressed model and M2M-100 model.

where $f_m$, and $f_f$ refer to F1 scores of male and female, respectively. if $\psi$ is near zero, then the model is not biased toward any gender, however, $\psi$ values of +1 or -1 mean that the model is highly biased toward male or female, respectively. We extend the fairness metric to pro- and anti-stereotypical subsets as follows:[20]:

$$\psi^* = |\psi_{anti} - \psi_{pro}| \tag{5.4}$$

where $\psi_{pro}$, and $\psi_{anti}$ belong to the fairness metric of pro- and anti-stereotypical sections. Intuitively, if the model has different behaviors in pro- and anti-stereotypical subsets, then it results in increasing the absolute difference of $\psi_{anti}$ and $\psi_{pro}$.[21] Average fairness metrics over 10 languages are illustrated in Table 5.6. Increasing the sparsity ratio results in a more biased model as both $\psi$ and $\psi^*$ relatively increase +67.2%, and +25.9%. Quantisation has less effect on the gender bias as both $\psi$ and $\psi^*$ negligibly change after applying it. Detailed results for each language are provided in Appendix 5.15. Interestingly, pruning 30% highly increases the gender bias even for high-resource languages e.g. French and German, while spBLEU is almost the same after the compression.

### 5.4.3 Word Sense Disambiguation Benchmark

In this section, we analyze the impact of the compression on semantic biases by evaluating our models on a multilingual word sense disambiguation benchmark. We first detail metrics used in Campolungo et al. (2022) to measure semantic biases.

**Notation.** Given a specific word ($w_i$), $l_{w_i}$ is defined as (lemmatisation, Part-of-Speech tag) pair. $\Pi_L(l_{w_i}) = \{\sigma_1, ..., \sigma_n\}$ is the ordered list of synsets according to WordNet's sense frequency (Miller et al., 1990) in language $L$. For instance, it is built as {the act of firing, photograph, drink, ...} for noun *shot* in English. $C_{l_{w_i}}(\sigma)$ is the index of synset ($\sigma$) in $\Pi_L(l_{w_i})$.

**SFII.** is calculated as the error rate averaged over $C_{l_{w_i}}(\sigma)$ for different positions and words $w_i$. Intuitively, it measures the sensitivity of the model when predicting a sense concerning its corresponding index in $\Pi_L(l_{w_i})$.

[20]Pro-stereotypical sentences refer to samples that context and occupation match (e.g. The carpenter stopped the housekeeper and helped her.) while anti-stereotypical subset contains sentences that context and occupation do not match.

[21]Proposed metrics are different than simple absolute score difference of Kocmi et al. (2020), more details in Appendix 5.14.

| Model | SFII | SPDI | MFS | MFS$^+$ | AVG |
|---|---|---|---|---|---|
| Baseline | 77.6 | 71.6 | 52.8 | 87.6 | 72.4 |
| Pruned 30% | 76.4 | 72.2 | 52.9 | 87.8 | 72.4 |
| Pruned 45% | 80.2 | 74.8 | 53.4 | 87.8 | 74.1 |
| Quantised | 79.5 | 74 | 53.7 | 88.8 | 74 |

Table 5.7: The average semantic bias metrics over languages of DiBiMT (Campolungo et al., 2022). Last column is the average score of bias metrics for each model.

**SPDI.** is computed as the average error rate based on polysemy degrees of synsets.
**MFS.** measures how often the model chooses more frequent senses than the correct one. Given $C_{l_{w_i}}(\sigma)$ for a synset, it is increased once the model predicts a synset ($\sigma'$) with $C_{l_{w_i}}(\sigma') < C_{l_{w_i}}(\sigma)$.
**MFS$^+$.** It is similar to the MFS metric, but it increases when $C_{l_{w_i}}(\sigma')$ equals to 1. Since metrics are based on the error rate, the lower values show that the model is less biased.
Table 5.7 demonstrates the semantic bias scores, averaged over all languages in DiBiMT (Campolungo et al., 2022).[22] The last column is the average of semantic bias metrics for each model. According to the average bias score, quantised and pruned 45% models amplify the bias metric by 1.6, and 1.7 points on average, compared to M2M-100, respectively. It confirms that the compression amplifies the semantic bias while keeping almost the same BLEU performance, especially for the quantisation (average BLEU scores are shown in Table 5.2).

## 5.5 Related Work

The first connection between compression and bias amplification has been made by Hooker et al. (2019, 2020) in the case of image classification. The same authors proposed an approach to find a subset of the dataset which contains samples that have disproportionately high errors after the compression. There is also recent work that analyses the effect of compression on pre-trained language models (Ogueji et al., 2022; Xu et al., 2021; Lauscher et al., 2021; Li et al., 2021a). Notably, de Vassimon Manela et al. (2021) demonstrated a higher gender bias in compressed pre-trained language models. Concerning NMT, Renduchintala et al. (2021) demonstrated that optimisation of inference speed up may result in gender bias amplification. To the best of our knowledge, this work is the first in-depth study of the impact of compression on massively multilingual models. We hope our findings would encourage further research on this topic.

## 5.6 Conclusion

We demonstrate the impacts of applying compression methods to the massively Multilingual Machine Translation models by evaluating compressed models on FLORES-101 (Goyal et al., 2021b), gender bias benchmark (Stanovsky et al., 2019), and word sense disambiguation benchmark (Campolungo et al., 2022). We show that while average BLEU drops negligibly, the performance of under-represented language pairs drops drastically. By analysing the attention

[22]Detailed results are provided in Appendix 5.16.

patterns, we showed that sparsity improves the performance of some medium-resource language pairs by removing the noisy memorisation, resulting in less hallucinations in the target translations. By evaluating our compressed models on gender bias and word sense disambiguation benchmarks, we show that the compression amplifies the intrinsic gender and semantic biases, even in high-resource language pairs. We hope our findings could be a starting point to consider the fairness aspects when compressing multilingual models.

# Appendix

## 5.7 Magnitude Pruning Strategy

Figure 5.9 shows the performance of pruned models with different pruning strategies. Results illustrate that pruning based on Transformer-layer is slightly better than pruning based on each module of the model, and separate pruning for self-attention and feed-forward Transformer layers.

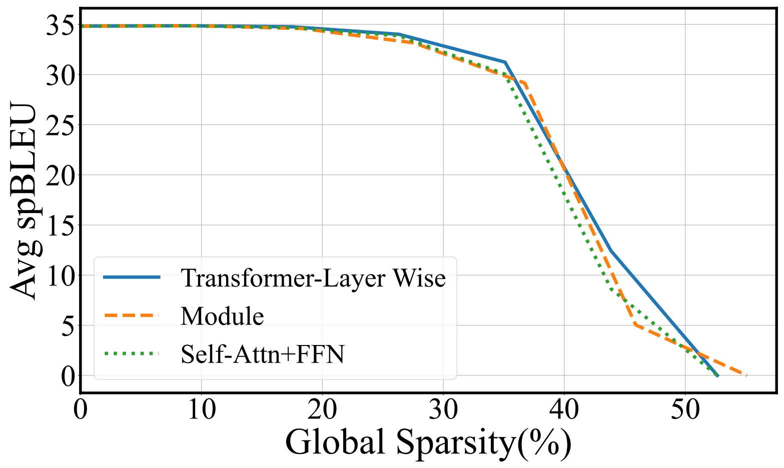


Figure 5.9: Average spBLEU score of different magnitude pruning strategies on 9 FLORES-101 language pairs, defined in Appendix 5.9.

## 5.8 Selection of Language Pairs in FLORES-101

Figure 5.10 shows the distribution of different language pair categories (defined in Table 6.1) based on spBLEU score of M2M-100 12B model (Fan et al., 2020a). We use 12 spBLEU as the threshold, which is approximately the average score over the median of different language pair categories.

Table 5.8 illustrates the number of language pairs in each category after the filtering.

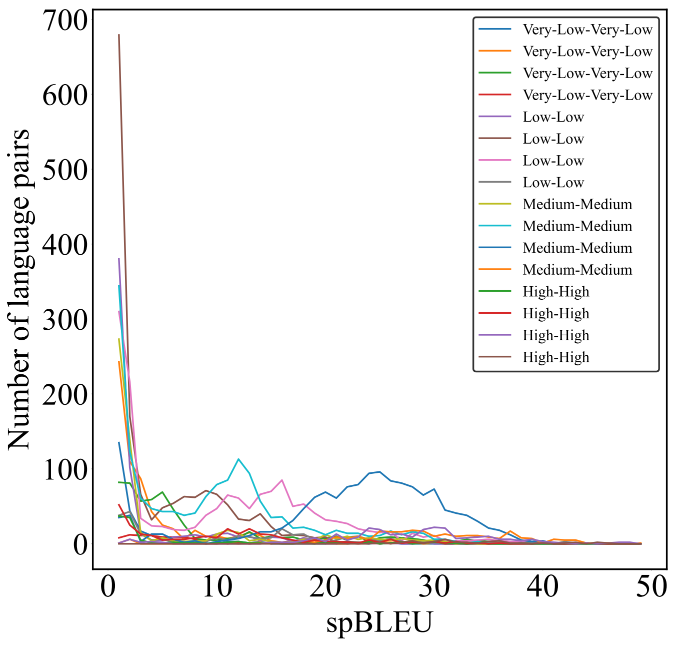


Figure 5.10: Histogram of number of language pairs based on spBLEU score for different language pair categories.

| Source \ Target | Very-Low | Low | Medium | High |
|---|---|---|---|---|
| Very-Low | 10 | 51 | 157 | 33 |
| Low | 58 | 164 | 643 | 143 |
| Medium | 108 | 440 | 1,277 | 257 |
| High | 23 | 103 | 252 | 39 |

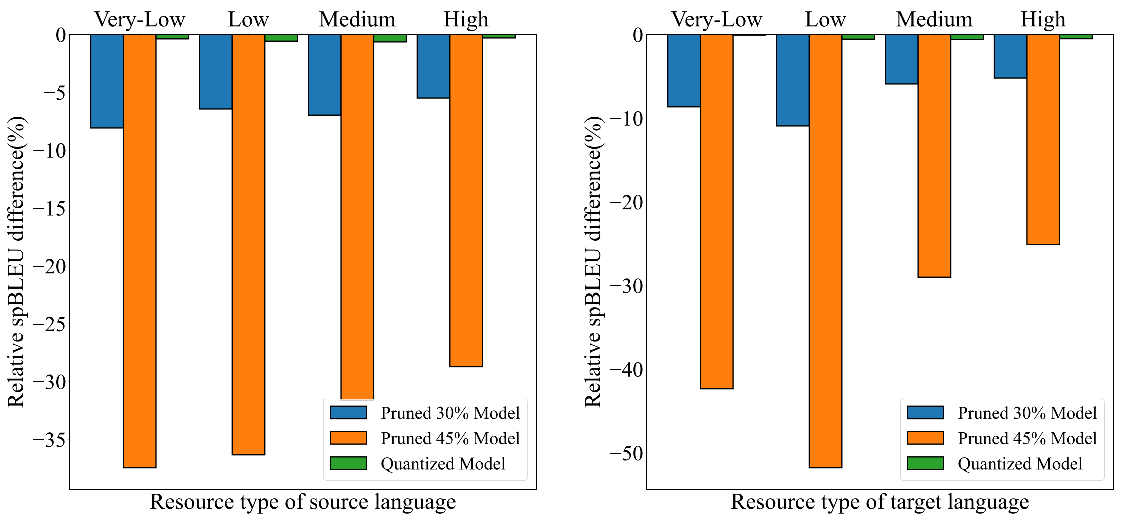
Table 5.8: Number of language pairs in each category after the filtering.

## 5.9 Language Pairs for Selection of Sparsity Ratio

| Language Pair | Resource-Type | M2M-100 spBLEU |
|---|---|---|
| Bosnian-Afrikaans | low-to-low | 29.9 |
| Afrikaans-Bulgarian | low-to-medium | 37.3 |
| Afrikaans-French | low-to-high | 41.5 |
| Catalan-Asturian | medium-to-low | 29.7 |
| Danish-Bulgarian | medium-to-medium | 37.8 |
| Swedish-Spanish | medium-to-high | 27.5 |
| French-Afrikaans | high-to-low | 30.9 |
| Spanish-Swedish | high-to-medium | 27.5 |
| English-French | high-to-high | 51.3 |

Table 5.9: Subset of language pairs used to compute average spBLEU score of Figure 5.2. M2M-100 model achieves reasonable performance for all selected pairs as shown in the last column.

## 5.10 Relative spBLEU based on Resource Type of Target and Source

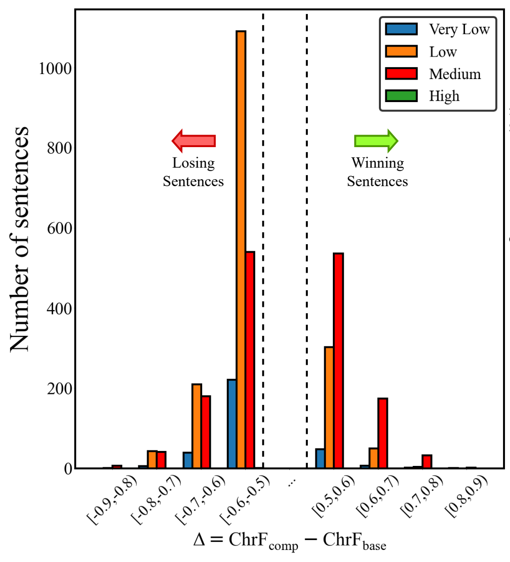


(a) Source Resource Type (b) Target Resource Type

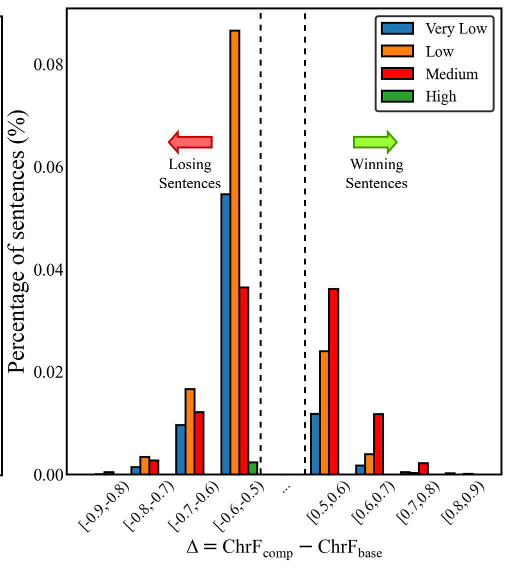
Figure 5.11: Relative spBLEU difference (%) between compressed models and M2M-100 model grouped by the resource type of source or target languages.

## 5.11 ChrF Difference Analysis

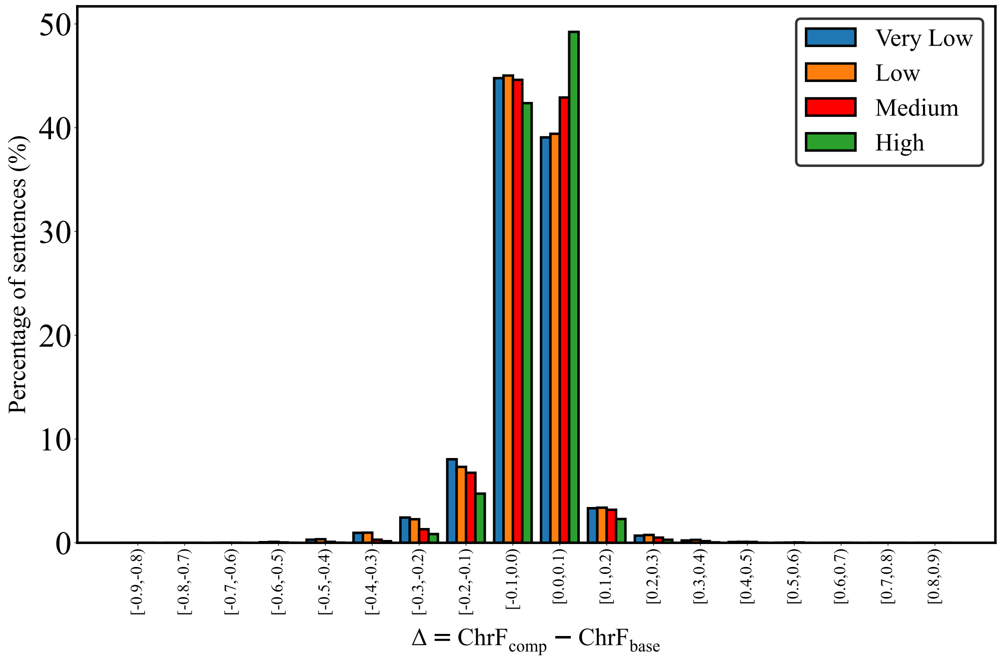
### 5.11.1 Pruned 30% Model



(a) Absolute number of sentences.



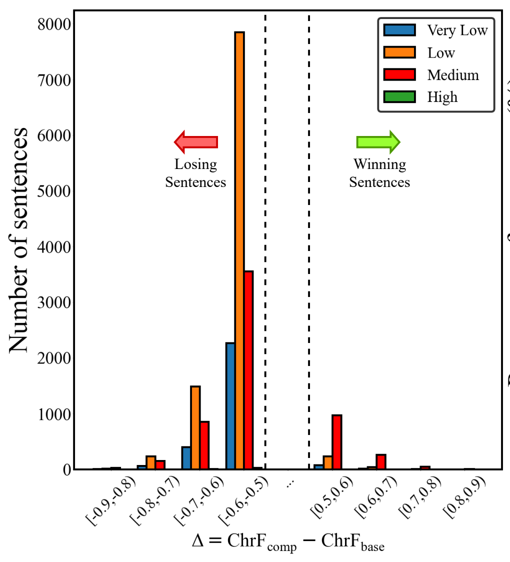
(b) Normalised distribution of sentences.



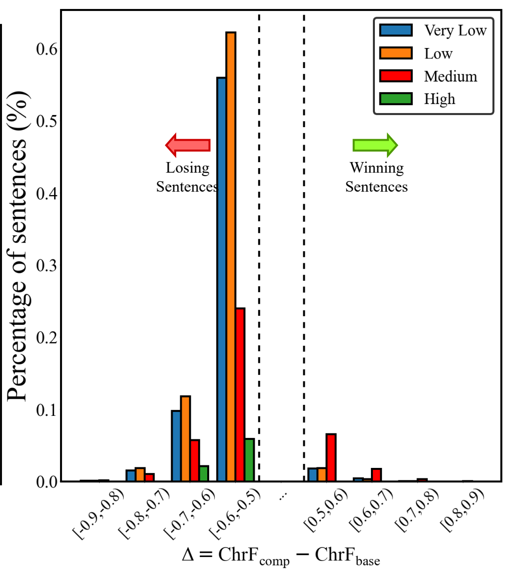
(c) Normalised distribution of sentences in each bin for different categories.

Figure 5.12: ChrF analysis of pruned 30% M2M-100 model.

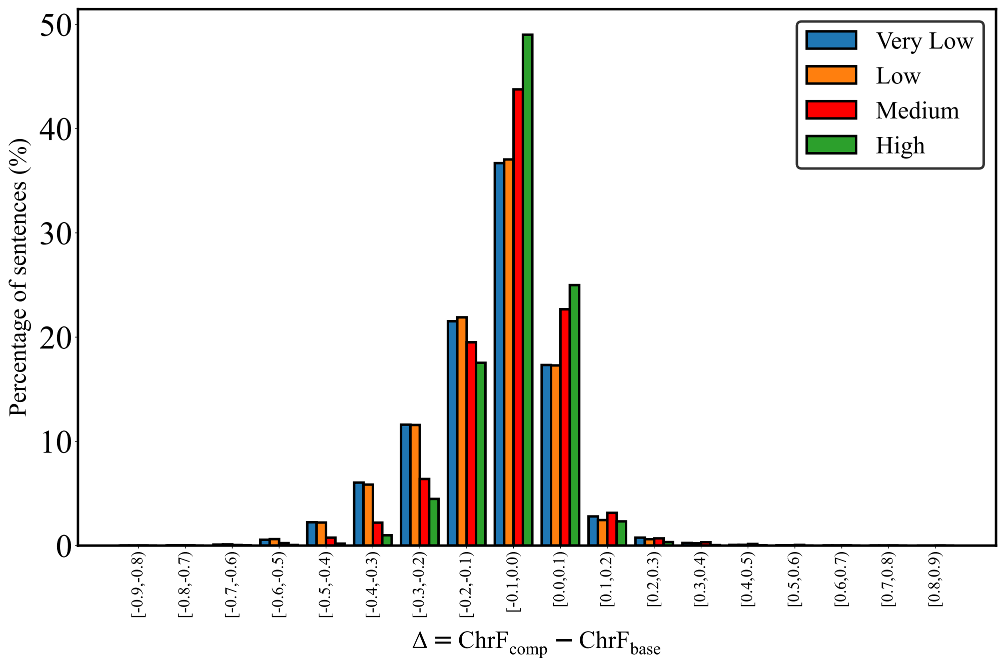
### 5.11.2 Pruned 45% Model



(a) Absolute number of sentences.



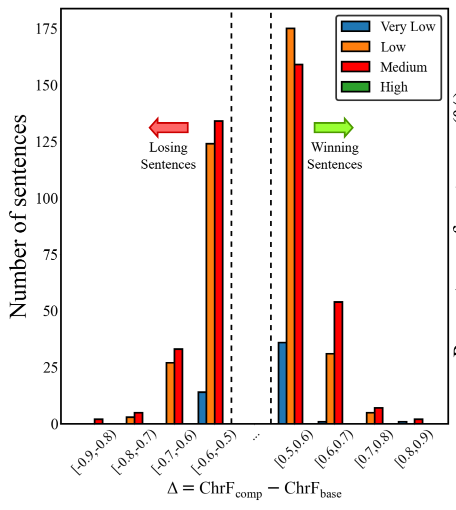
(b) Normalised distribution of sentences.



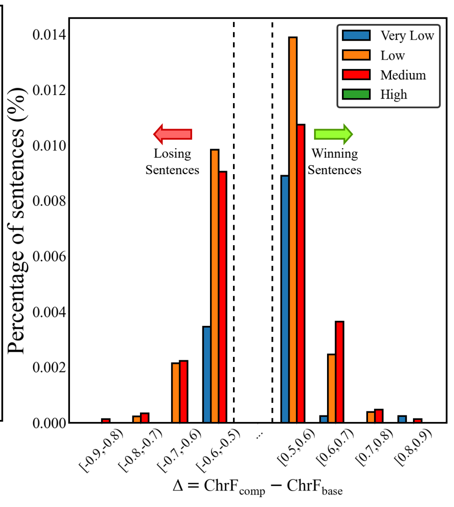
(c) Normalised distribution of sentences in each bin for different categories.

Figure 5.13: ChrF analysis of pruned 45% M2M-100 model.

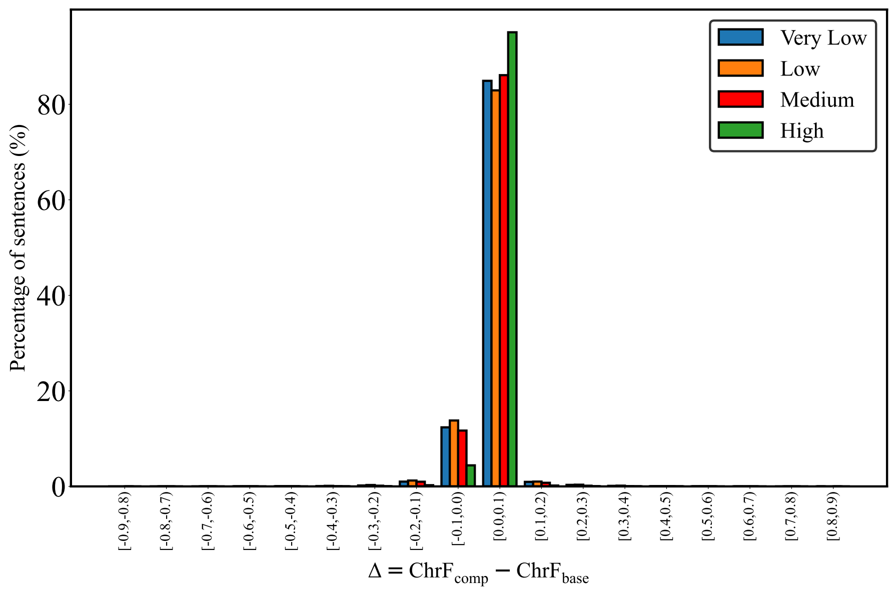
### 5.11.3 Quantised Model



(a) Absolute number of sentences.



(b) Normalised distribution of sentences.



(c) Normalised distribution of sentences in each bin for different categories.

Figure 5.14: ChrF analysis of quantised M2M-100 model.

## 5.12 Languages with Two Scripts in M2M-100 Training

| ISO | Language |
|---|---|
| sr | Serbian |
| cy | Welsh |
| az | Azerbaijani |
| uz | Uzbek |
| ja | Japanese |
| bn | Bengali |
| lo | Lao |
| zh | Chinese |

Table 5.10: Languages for which M2M-100 training data contains two scripts, while FLORES-101 provides one script for the evaluation.

## 5.13 Most Affected Language Pairs After Compression

Language pairs are selected, if both quantisation and pruning have significant effect on them (based on spBLEU performance shown in Figure 5.3).

| Source | Target |
|---|---|
| Catalan | Cebuano |
| Latvian | Igbo |
| Arabic | Igbo |
| Danish | Xhosa |
| French | Zulu |

(a) Most losing language pairs

| Source | Target |
|---|---|
| Latvian | Vietnamese |
| Bulgarian | Latvian |
| Arabic | Urdu |
| Thai | Vietnamese |
| Latvian | Italian |

(b) Most winning language pairs

Table 5.11: Most affected language pairs after the compression.

## 5.14 Proposed Metrics for MT-Gender Benchmark

Equation 5.3 considers the range of F1 scores for female and male subsets, while the simple difference between F1 scores does not reflect the range of F1 scores. The range is crucial since a model with the same F1 score difference but higher individual F1 scores should have a lower fairness score, as lied in Equation 5.3.
We also believe equation 5.4 is a better metric than the simple difference between accuracies of the model in pro-stereotypical and anti-stereotypical subsets since it again considers the range of scores, and ignores missed translations and wrongly aligned genders. Additionally, it exactly reflects the difference in the behavior of the model in these two subsets. If the compressed model has a contrary performance in pro- and anti-stereotypical subsets, e.g. amplifying the bias in the anti-stereotypical subset more than the pro-stereotypical one or decreasing the bias more in one subset, then $\psi*$ becomes higher. We suggest using Equation 5.3 and Equation 5.4 for comparing models on MT-Gender benchmark (Kocmi et al., 2020; Stanovsky et al., 2019).

## 5.15 MT-Gender Results per Language

| Model | $\psi$ (%) | $\psi^*$ (%) |
|---|---|---|
| Original M2M-100 | 21.01 | 15.09 |
| Pruned 30% M2M-100 | 20.71 | 16.87 |
| Pruned 45% M2M-100 | 28.58 | 17.33 |
| Quantised M2M-100 | 18.07 | 12.55 |

(a) Arabic

| Model | $\psi$ (%) | $\psi^*$ (%) |
|---|---|---|
| Original M2M-100 | 39.02 | 11.39 |
| Pruned 30% M2M-100 | 45.19 | 7.15 |
| Pruned 45% M2M-100 | 45.56 | 18.54 |
| Quantised M2M-100 | 40.93 | 2.54 |

(b) Ukrainian

| Model | $\psi$ (%) | $\psi^*$ (%) |
|---|---|---|
| Original M2M-100 | 7.98 | 20.09 |
| Pruned 30% M2M-100 | 10.38 | 16.30 |
| Pruned 45% M2M-100 | 8.89 | 2.75 |
| Quantised M2M-100 | 10.39 | 21.26 |

(c) Hebrew

| Model | $\psi$ (%) | $\psi^*$ (%) |
|---|---|---|
| Original M2M-100 | 29.06 | 3.93 |
| Pruned 30% M2M-100 | 29.10 | 2.30 |
| Pruned 45% M2M-100 | 30.28 | 8.08 |
| Quantised M2M-100 | 32.65 | 8.74 |

(d) Russian

| Model | $\psi$ (%) | $\psi^*$ (%) |
|---|---|---|
| Original M2M-100 | 22.46 | 2.03 |
| Pruned 30% M2M-100 | 30.17 | 13.81 |
| Pruned 45% M2M-100 | 48.59 | 4.61 |
| Quantised M2M-100 | 24.71 | 2.6 |

(e) Italian

| Model | $\psi$ (%) | $\psi^*$ (%) |
|---|---|---|
| Original M2M-100 | 13.86 | 28.71 |
| Pruned 30% M2M-100 | 29.03 | 40.20 |
| Pruned 45% M2M-100 | 38.44 | 32.83 |
| Quantised M2M-100 | 15.43 | 25.86 |

(f) French

| Model | $\psi$ (%) | $\psi^*$ (%) |
|---|---|---|
| Original M2M-100 | 5.77 | 15.72 |
| Pruned 30% M2M-100 | 4.89 | 14.62 |
| Pruned 45% M2M-100 | 22.53 | 34.01 |
| Quantised M2M-100 | 6.01 | 15.11 |

(g) Spanish

| Model | $\psi$ (%) | $\psi^*$ (%) |
|---|---|---|
| Original M2M-100 | 6.48 | 16.93 |
| Pruned 30% M2M-100 | 13.16 | 26.83 |
| Pruned 45% M2M-100 | 22.14 | 18.12 |
| Quantised M2M-100 | 6.23 | 14.96 |

(h) German

| Model | $\psi$ (%) | $\psi^*$ (%) |
|---|---|---|
| Original M2M-100 | 18.20 | 39.01 |
| Pruned 30% M2M-100 | 21.82 | 42.60 |
| Pruned 45% M2M-100 | 25.95 | 45.01 |
| Quantised M2M-100 | 18.24 | 38.42 |

(i) Polish

| Model | $\psi$ (%) | $\psi^*$ (%) |
|---|---|---|
| Original M2M-100 | 7.91 | 12.14 |
| Pruned 30% M2M-100 | 11.65 | 14.43 |
| Pruned 45% M2M-100 | 19.31 | 27.23 |
| Quantised M2M-100 | 9.78 | 13.26 |

(j) Czech

Table 5.12: MT-Gender (Stanovsky et al., 2019; Kocmi et al., 2020) results for M2M-100 12B (Fan et al., 2020a), and compressed models.

## 5.16 Detailed DiBiMT Results

| Model | SFII | SPDI | MFS | MFS$^+$ | Avg |
|---|---|---|---|---|---|
| Original M2M-100 | 89.14 | 80.59 | 41.8 | 92.59 | 76.03 |
| Pruned 30% M2M-100 | 87.32 | 80.56 | 39.55 | 93.04 | 75.11 |
| Pruned 45% M2M-100 | 86.78 | 82.9 | 39.93 | 92.41 | 75.50 |
| Quantised M2M-100 | 88.86 | 81.26 | 43.32 | 92.51 | 76.48 |

(a) Chinese

| Model | SFII | SPDI | MFS | MFS$^+$ | Avg |
|---|---|---|---|---|---|
| Original M2M-100 | 80 | 71.61 | 60.63 | 89.76 | 75.5 |
| Pruned 30% M2M-100 | 78.96 | 73.79 | 61.44 | 88.56 | 75.68 |
| Pruned 45% M2M-100 | 81.28 | 77.05 | 62.5 | 91.67 | 78.12 |
| Quantised M2M-100 | 82.32 | 74.42 | 61.07 | 91.22 | 77.25 |

(b) German

| Model | SFII | SPDI | MFS | MFS$^+$ | Avg |
|---|---|---|---|---|---|
| Original M2M-100 | 75.99 | 70.53 | 61.23 | 88.41 | 74.04 |
| Pruned 30% M2M-100 | 75.91 | 71.86 | 60.92 | 87.74 | 74.10 |
| Pruned 45% M2M-100 | 83.38 | 75.08 | 62.22 | 86.67 | 76.83 |
| Quantised M2M-100 | 81.73 | 75.81 | 63.33 | 88.33 | 77.3 |

(c) Italian

| Model | SFII | SPDI | MFS | MFS$^+$ | Avg |
|---|---|---|---|---|---|
| Original M2M-100 | 68.16 | 66.42 | 47.06 | 83.82 | 66.36 |
| Pruned 30% M2M-100 | 68.2 | 64.73 | 48.21 | 87.18 | 67.08 |
| Pruned 45% M2M-100 | 70.92 | 66.41 | 50 | 85.29 | 68.15 |
| Quantised M2M-100 | 68.16 | 69.03 | 44.19 | 86.51 | 66.97 |

(d) Russian

| Model | SFII | SPDI | MFS | MFS$^+$ | Avg |
|---|---|---|---|---|---|
| Original M2M-100 | 75.08 | 68.92 | 53.44 | 83.61 | 70.26 |
| Pruned 30% M2M-100 | 71.58 | 70.26 | 54.58 | 82.71 | 69.78 |
| Pruned 45% M2M-100 | 78.39 | 72.46 | 52.33 | 83.15 | 71.58 |
| Quantised M2M-100 | 76.45 | 69.72 | 56.88 | 85.63 | 72.17 |

(e) Spanish

Table 5.13: DiBiMT (Campolungo et al., 2022) evaluation for M2M-100 12B (Fan et al., 2020a), and compressed models.

# 6 SMaLL-100: Introducing Shallow Multilingual Machine Translation Model for Low-Resource Languages

In recent years, multilingual machine translation models have achieved promising performance on low-resource language pairs by sharing information between similar languages, thus enabling zero-shot translation. To overcome the "curse of multilinguality", these models often opt for scaling up the number of parameters, which makes their use in resource-constrained environments challenging. In this chapter, we introduce *SMaLL-100*, a distilled version of the M2M-100 (12B) model, a massively multilingual machine translation model covering 100 languages. We train SMaLL-100 with uniform sampling across all language pairs and therefore focus on preserving the performance of low-resource languages. We evaluate SMaLL-100 on different low-resource benchmarks: FLORES-101, Tatoeba, and TICO-19 and demonstrate that it outperforms previous massively multilingual models of comparable sizes (200-600M) while improving inference latency and memory usage. Additionally, our model achieves comparable results to M2M-100 (1.2B), while being 3.6× smaller and 4.3× faster at inference.[1]

## 6.1 Introduction

Neural Machine Translation (NMT) systems are usually trained on datasets consisting of millions of parallel sentences, thus still performing poorly on low-resource languages, i.e., languages without a large amount of training data. Over the past few years, previous work has proposed several approaches to improve the quality of translations in low-resource languages, e.g., Multilingual Neural Machine Translation (MNMT) models (Tang et al., 2021; Goyal et al., 2021b; Fan et al., 2020a; Johnson et al., 2017), back-translation (Edunov et al., 2018; Sennrich et al., 2016) and unsupervised machine translation (Garcia et al., 2021; Ko et al., 2021). Massively MNMT models are particularly interesting for low-resource languages as they benefit the most from knowledge transfer from related languages (Arivazhagan et al., 2019). However, it is also seen that *curse of multilinguality* hurts the performance of high-resource languages. So, previous work attempted to increase the model size to maintain the translation performance in

[1]The code and pre-trained SMaLL-100 model is available at `https://github.com/alirezamshi/small100`.

both high and low-resource languages. This makes the use of these massively MNMT models challenging in real-world resource-constrained environments. To overcome this problem, we propose SMaLL-100, a **S**hallow **M**ultilingual M**a**chine Translation Model for **L**ow-Resource **L**anguages covering 100 languages, which is a distilled alternative of M2M-100 (12B) (Fan et al., 2020a), the most recent and biggest available multilingual NMT model. We focus on very-low and low-resource language pairs as there is no reasonable-size universal model that achieves acceptable performance over a great number of low-resource languages. We do so by training SMaLL-100 on a perfectly balanced dataset.[2] While this leads to lower performance on the high-resource languages, we claim that this loss is easily recoverable through further fine-tuning. We evaluate SMaLL-100 on different low-resource benchmarks, e.g., FLORES-101 (Goyal et al., 2021b), Tatoeba (Tiedemann, 2020), and TICO-19 (Anastasopoulos et al., 2020). To summarise, the contributions of this chapter are as follows:

- We propose SMaLL-100, a shallow multilingual NMT model, focusing on low-resource language pairs.
- We evaluate SMaLL-100 on several low-resource NMT benchmarks.
- We show that our model significantly outperforms previous multilingual models of comparable size while being faster at inference. Additionally, it achieves comparable results with M2M-100 (1.2B) model, with 4.3× faster inference and a 3.6× smaller size.
- While SMaLL-100 reaches 87.2% performance of the 12B teacher model, we show that this gap can be closed with a few fine-tuning steps both for low and high-resource languages.

## 6.2 Model and Training

### 6.2.1 SMaLL-100 Architecture

It has been shown by Kasai et al. (2021) that deep encoder / shallow decoder architectures can achieve good translation quality while being significantly faster at inference. Berard et al. (2021) have confirmed that this is also valid for multilingual NMT. Here, we use a 12-layer Transformer encoder (Vaswani et al., 2017) and 3-layer decoder. Table 6.8 in the Appendix 6.8 reports further details of the SMaLL-100 architecture. Different from M2M-100 model, we use language codes in the encoder side, as it is shown to perform better with shallow decoder architectures (Berard et al., 2021).

### 6.2.2 Training Strategy

SMaLL-100 is trained with a combination of two loss functions: a standard Cross Entropy loss (CE) and a Knowledge Distillation loss (KD). Given source sequence $X$ and gold target

[2]All language pairs have the same sampling probability, regardless of their training data size.

translation $Y = (y_0, ..., y_m)$, the CE loss is calculated as:

$$\mathcal{L}_{ce} = -\sum_{j=0}^{m} \sum_{z=1}^{|K|} \mathbb{1}\{y_j = z\} \log\ p(y_j = z | y_{<j}, X, \theta_S) \tag{6.1}$$

where $|K|$ is the target vocabulary size, $\mathbb{1}$ is the indicator function, and $\theta_S$ is the model parameters. $p()$ is the conditional probability function.

We additionally use a word-level distillation loss, which is the Kullback–Leibler divergence between the output distributions of the student and teacher models (Hu et al., 2018). Specifically, it is calculated as:

$$\begin{aligned} \mathcal{L}_{kd} = -\sum_{j=0}^{m} \sum_{z=1}^{|K|} & q(y_j = z | y_{<j}, X, \theta_T) \\ & \times \log\ p(y_j = z | y_{<j}, X, \theta_S) \end{aligned} \tag{6.2}$$

where $\theta_T$ is parameters of the teacher model. $q()$ is the conditional probability of the teacher model. The total loss is computed as:

$$\mathcal{L}_{total} = \mathcal{L}_{ce} + \alpha \mathcal{L}_{kd} \tag{6.3}$$

where $\alpha$ is a trainable parameter.

### 6.2.3 Training Data

Our training data includes sentences from CCMatrix (Schwenk et al., 2021) and CCAligned (El-Kishky et al., 2020) datasets, which are part of the training data used by Fan et al. (2020a) to train the M2M-100 models. As our goal is to maintain the performance of low-resource languages, we balance the training data across all language pairs; specifically, 100K sentence pairs are sampled for each language pair.[3] As a result, our training data contains nearly 456M parallel sentences, which is less than 6% of the original data on which M2M-100 (Fan et al., 2020a) was trained. We use the same languages as M2M-100.

## 6.3 Experimental Setup

### 6.3.1 Evaluation Benchmarks

**FLORES-101.** is a multilingual NMT benchmark, containing 3,001 sentences from different domains, that are derived from English Wikipedia. Sentences are translated into 101 languages by human translators (Goyal et al., 2021b). It mostly includes low and medium-resource languages. We use `devtest` subset for the evaluation.

[3] For language pairs with less than 100K sentence pairs, we repeat their data. We randomly select 100K sentences for language pairs with more than 100K training sentences.

| Resource Type | Criteria |
|---|---|
| Very-Low | $\|K\| \leq 100K$ |
| Low | $100K < \|K\| \leq 1M$ |
| Medium | $1M < \|K\| \leq 100M$ |
| High | $100M < \|K\|$ |

Table 6.1: The criteria to split languages into different resource categories. $|K|$ is the amount of training data to/from English.

**Tatoeba.** is a crowd-sourced collection of user-provided translations in different languages (Tiedemann, 2020). We choose a subset of languages from `test` set of Tatoeba Challenge,[4] which are covered by M2M-100.

**TICO-19.** was created during the COVID-19 pandemic (Anastasopoulos et al., 2020). It contains sentences from 36 languages in the medical domain, including 26 low-resource languages. We evaluate on languages which are covered by M2M-100 (Fan et al., 2020a).

Inspired by Goyal et al. (2021b), we split the languages based on the amount of available training sentences aligned with English into 4 different categories: Very-Low (VL), Low (L), Medium (M), and High-resource (H). As the true amount of training data is both dependent on quality and quantity of parallel sentences, Goyal et al. (2021b) suggested to estimate it by computing the number of bitext data aligned with English, that is calculated from statistics of OPUS corpora (Tiedemann, 2012). Table 6.1 illustrates the criteria for choosing the category of different languages. More details about the distribution of language pair categories in each benchmark are provided in Appendix 6.7.

### 6.3.2 Baselines

**M2M-100.** is a recent many-to-many NMT model covering 100 languages. Fan et al. (2020a) provide 3 variants with respectively 418M, 1.2B, and 12B parameters. We compare against these 3 variants.
**FLORES-124.** is an extension of M2M-100, covering additional 24 languages. Training data of the additional languages is derived from OPUS (Tiedemann, 2012). Goyal et al. (2021b) provide two models with 175M and 615M parameters. We use both models as baselines.
**FineTuned-100.** uses the same architecture as defined in Section 6.2, but KD loss ($\mathcal{L}_{kd}$) is not used for training. For a fair comparison, it is trained for the same number of steps as SMaLL-100 model.

[4]`https://github.com/Helsinki-NLP/Tatoeba-Challenge`

| | | | Language Direction | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | params | Speed | VL2VL | VL2L | VL2M | VL2H | L2VL | L2L | L2M | L2H | M2VL | M2L | H2VL | H2L | AVG |
| **FLORES-101** | | | | | | | | | | | | | | | |
| FLORES-124 | 175M | 5.3× | 3.3 | 3.4 | 6.0 | 7.8 | 3.7 | 3.1 | 6.9 | 8.8 | 6.9 | 5.2 | 8.1 | 6.0 | 5.8 |
| M2M-100 | 418M | 3.1× | 4.3 | 3.7 | 7.8 | 9.4 | 5.4 | 3.4 | 9.1 | 11.3 | 9.9 | 5.8 | 11.4 | 6.6 | 7.3 |
| FLORES-124 | 615M | 2.9× | 5.1 | 5.1 | 9.2 | 11.2 | 5.8 | 4.7 | 10.6 | 13.1 | 10.3 | 7.6 | 11.5 | 8.5 | 8.6 |
| *Finetuned-100* | 330M | 7.8× | 6.1 | 5.4 | 8.7 | 11.3 | 5.7 | 4.1 | 9.0 | 11.8 | 10.4 | 6.8 | 13.0 | 8.0 | 8.4 |
| *SMaLL-100* | 330M | 7.8× | $\underline{7.9}$ | $\underline{7.0}$ | 10.3 | 12.6 | 8.4 | $\underline{6.1}$ | 11.6 | 14.3 | $\underline{13.7}$ | $\underline{9.0}$ | $\underline{16.7}$ | $\underline{10.2}$ | $\underline{10.7}$ |
| M2M-100 | 1.2B | 1.8× | 6.7 | 6.1 | $\underline{10.8}$ | $\underline{12.8}$ | $\underline{8.7}$ | $\underline{6.1}$ | $\underline{13.0}$ | $\underline{15.9}$ | 13.6 | 8.8 | 15.4 | 9.7 | 10.6 |
| M2M-100 | 12B | 1× | **8.7** | **8.8** | **11.9** | **13.7** | **11.7** | **9.7** | **15.4** | **18.2** | **16.5** | **12.6** | **18.7** | **13.9** | **13.3** |
| **Tatoeba** | | | | | | | | | | | | | | | |
| FLORES-124 | 175M | 5.3× | - | 7.6 | 15.7 | 10.1 | 4.6 | 5.3 | 11.5 | 10.8 | 14.0 | 10.2 | 6.4 | 7.5 | 9.4 |
| M2M-100 | 418M | 3.1× | - | 7.4 | 19.7 | 12.3 | 5.9 | 5.3 | 13.8 | 13.2 | 14.9 | 11.7 | 7.7 | 9.0 | 10.9 |
| FLORES-124 | 615M | 2.9× | - | **9.1** | 19.4 | 11.4 | 6.9 | $\underline{7.6}$ | 12.7 | 13.7 | 14.4 | 13.3 | 8.0 | 9.7 | 11.4 |
| *Finetuned-100* | 330M | 7.8× | - | 4.0 | 21.1 | $\underline{14.4}$ | 7.7 | 5.2 | 15.3 | 14.2 | 14.0 | 12.1 | 8.9 | 8.3 | 11.4 |
| *SMaLL-100* | 330M | 7.8× | - | 4.6 | $\underline{22.1}$ | **16.4** | $\underline{8.7}$ | 7.0 | $\underline{16.7}$ | 15.8 | 16.3 | $\underline{14.5}$ | $\underline{10.6}$ | $\underline{11.2}$ | $\underline{13.1}$ |
| M2M-100 | 1.2B | 1.8× | - | $\underline{8.8}$ | 19.5 | 13.1 | $\underline{8.7}$ | 7.2 | 16.3 | $\underline{17.0}$ | $\underline{17.2}$ | 13.4 | **10.7** | 11.1 | 13.0 |
| M2M-100 | 12B | 1× | - | 8.6 | **23.5** | 13.1 | **9.8** | **10.2** | **17.8** | **17.9** | **18.5** | **15.2** | **10.7** | **13.2** | **14.4** |
| **TICO-19** | | | | | | | | | | | | | | | |
| FLORES-124 | 175M | 5.3× | 4.6 | 5.5 | 8.1 | 11.5 | 4.4 | 5.6 | 9.7 | 12.2 | 3.9 | 8.0 | 4.2 | 8.7 | 7.2 |
| M2M-100 | 418M | 3.1× | 4.0 | 5.5 | 9.8 | 13.7 | 4.2 | 5.7 | 11.6 | 14.9 | 4.1 | 8.8 | 5.3 | 9.4 | 8.1 |
| FLORES-124 | 615M | 2.9× | 4.6 | 7.4 | 11.5 | 16.4 | 4.8 | 7.6 | 12.9 | 16.7 | 4.4 | 10.7 | 4.4 | 11.5 | 9.4 |
| *Finetuned-100* | 330M | 7.8× | 6.1 | 7.2 | 11.9 | 17.4 | 5.5 | 6.1 | 12.1 | 15.2 | $\underline{6.4}$ | 9.0 | $\underline{9.5}$ | 10.3 | 9.7 |
| *SMaLL-100* | 330M | 7.8× | **7.8** | $\underline{8.8}$ | $\underline{13.3}$ | $\underline{19.0}$ | **8.0** | 8.5 | $\underline{14.3}$ | 17.8 | **8.3** | $\underline{11.5}$ | **11.3** | $\underline{12.7}$ | $\underline{11.8}$ |
| M2M-100 | 1.2B | 1.8× | 5.4 | 8.2 | 13.2 | 18.9 | 6.0 | $\underline{8.7}$ | 14.0 | $\underline{19.2}$ | 5.2 | $\underline{11.5}$ | 6.1 | 12.5 | 10.8 |
| M2M-100 | 12B | 1× | $\underline{6.4}$ | **10.9** | **15.4** | **20.6** | $\underline{7.8}$ | **11.9** | **16.6** | **21.4** | $\underline{6.4}$ | **15.4** | 8.7 | **16.4** | **13.1** |

Table 6.2: Average spBLEU performance on FLORES-101, Tatoeba, and TICO-19 benchmarks for different language pair categories, defined in Appendix 6.7. FLORES-101 results are computed on language pairs where M2M-100 12B has spBLEU scores higher than 3 to avoid polluting the analysis with meaningless scores. The first and second columns give the model size and speed-up ratios compared to M2M-100 (12B). Last column is the average spBLEU performance over all mentioned language directions. The best scores are shown in bold, and the second best results are shown with underline.

### 6.3.3 Implementation Details

SMaLL-100 contains nearly 330M parameters with 12 encoder and 3 decoder Transformer layers.[5] It is trained for 30 days on 16 TESLA V100-32GB GPUs,[6] with a batch size of 1K tokens and accumulated gradients over 9 batches. We implement our model using fairseq repository.[7] We use `last-checkpoint`[8] of M2M-100 (12B) for the teacher model. For decoding, the beam search of 5 is applied. All hyper-parameters regarding the architecture and optimisation strategy are provided in Appendix 6.8.
For a faster convergence, we first fine-tune SMaLL-100 for 150k steps without distillation ($\mathcal{L}_{kd}$). Then, it is trained with both losses for 756K steps (nearly 1 epoch). For evaluation, we use SentencePiece BLEU (spBLEU), as it is shown to be a fair metric in multilingual settings (Goyal

[5]It is initialised with M2M-100 (418M), using its first 3 decoder layers for the initialisation of the student's decoder.

[6]4 GPUs are used for the training of the student model. 12 GPUs are utilised to do the model parallelism of the teacher model.

[7]`https://github.com/facebookresearch/fairseq`

[8]`https://github.com/facebookresearch/fairseq/tree/main/examples/m2m_100`

et al., 2021b).[9] We use the same tokeniser and dictionary as M2M-100.

## 6.4 Results and Discussion

### 6.4.1 Low-Resource NMT Benchmarks

Table 6.2 shows the average spBLEU performance on FLORES-101, Tatoeba, and TICO-19 test sets for different categories of language directions.[10] SMaLL-100 outperforms all the models with comparable sizes while being smaller and faster at inference. Specifically, it outperforms M2M-100 418M both in terms of performance (+3.1 spBLEU) and inference speed (2.5× faster). We believe that Finetuned-100 outperforms M2M-100 418M for low-resource languages thanks to finetuning on the balanced dataset. The higher performance of SMaLL-100 compared to Finetuned-100 across all benchmarks shows the benefit of KD loss which allows to distill knowledge from the teacher model. Additionally, SMaLL-100 achieves competitive results with M2M-100 (1.2B), while being 3.6× smaller and 4.3× faster at inference. Compared to the biggest M2M-100 model (12B), SMaLL-100 loses nearly 1.7 spBLEU but is 36× smaller and 7.8× faster. Regarding medium and high-resource language pairs (as shown in Appendix 6.9.1), SMaLL-100 achieves better or similar performance compared to M2M-100 (418M) and FLORES-124 (615M), while it contains fewer parameters and is faster at the evaluation time. It under-performs for some medium and high-resource language pairs compared to the teacher model (M2M-100 12B), which could be easily recovered, as we describe in the remaining section.

### 6.4.2 Recovering Teacher Model Performance

To go further, we demonstrate that SMaLL-100 can easily recover the performance of the teacher model with just a few fine-tuning steps, both for low and high-resource language pairs. For comparison, we fine-tune M2M-100 (418M) model with the same number of steps.
Table 6.3 reports spBLEU performance for several language pairs, alongside the number of fine-tuning steps, required by SMaLL-100 model to reach M2M-100 (12B) performance.[11] We see that SMaLL-100 achieves better performance than M2M-100 (12B) after a few training steps on low-resource language pairs. For high-resource language pairs, SMaLL-100 is fine-tuned for 20K steps to reach the performance of M2M-100 (12B) model. Additionally, fine-tuned SMaLL-100 significantly outperforms fine-tuned M2M-100 (418M) model on low and medium-resource languages. This confirms that SMaLL-100 could be a powerful and lightweight initialisation model for training on different language pairs.

[9]It utilises a SentencePiece tokeniser with 256K tokens: `https://github.com/facebookresearch/flores`

[10]Complete spBLEU calculations of different language pairs on tested NMT benchmarks are provided in Appendix 6.9. Speed is calculated on 2 TESLA V100-32GB GPUs with a batch size of 1 sentence over a subset of FLORES-101 devtest, containing nearly 10K sentences from all language pairs.

[11]More dataset and implementation details of these fine-tuning experiments are provided in Appendix 6.10.

| | | Fine-tuned | | | |
|---|---|---|---|---|---|
| Language pair | Language Type | M2M-100 (418M) | SMaLL-100 | steps | M2M-100 (12B) |
| Cebuano-English | Low | 18.6 | **29.8** | 0.5K | 27.7 |
| English-Igbo | Low | 9.2 | **15.7** | 1.5K | 14.9 |
| English-Malayalam | Low | 16.7 | **20.8** | 3K | 20.6 |
| Georgian-Russian | Medium | 6.9 | **13.1** | 0.5K | 10.1 |
| English-Italian | High | 33.3 | 33.4 | 20K | **33.5** |
| French-Italian | High | 31.3 | 31.5 | 20K | **32.0** |
| Italian-Spanish | High | 26.6 | 26.8 | 20K | **27.0** |

Table 6.3: spBLEU performance of fine-tuned SMaLL-100 and M2M-100 (418M) for the specified step, and M2M-100 (12B) on FLORES-101 devtest. The "step" column is the number of training steps required to reach M2M-100 (12B) performance. The type of each language pair is defined as the minimum of source and target language categories.

## 6.5 Related Work

**Compression and Distillation.** Over the past few years, pre-trained models lead to significant improvement by increasing the parameter size (Zhang et al., 2022; Fan et al., 2020a; Raffel et al., 2019), which makes it challenging to use them in the resource-constraint environment. Previous work use several compression techniques e.g. knowledge distillation (Li et al., 2021a; Kim and Rush, 2016), pruning (Zhang et al., 2021a; Behnke and Heafield, 2020), and quantisation (Tao et al., 2022; Yao et al., 2022) to provide a reasonable-size model, while keeping the performance.

**Multilingual NMT.** It provides a single model to translate between any pair of languages, which significantly improves performance on low-resource languages thanks to knowledge transfer (Haddow et al., 2021). Several works (Berard et al., 2021; Fan et al., 2020a; Platanios et al., 2018; Firat et al., 2016; Dong et al., 2015) propose to include both language-specific, and language-independent parameters in MNMT models. Recently, massively MNMT models (Zhang et al., 2020a; Fan et al., 2020a; Neubig and Hu, 2018; Arivazhagan et al., 2019; Aharoni et al., 2019) have been proposed to translate between more than 100 languages. However, these models usually contain a huge number of parameters to maintain performance in both high and low-resource languages. Different from the previous work, we introduce SMaLL-100, which outperforms previous models with comparable size in low-resource language directions, while achieving better speed and being smaller.

## 6.6 Conclusion

In this chapter, we presented SMaLL-100 model, a shallow multilingual NMT model, focusing on low-resource languages. We evaluated our model on different NMT benchmarks. SMaLL-100 significantly outperforms multilingual models of comparable size on all of the tested benchmarks (FLORES-101, Tatoeba, TICO-19) and is much faster at inference. It also achieves competitive results with M2M-100 1.2B (Fan et al., 2020a), while being 4.3× faster at inference and 3.6× smaller. Compared to M2M-100 (12B), the biggest available MNMT model, SMaLL-100 loses nearly 1.7 spBLEU on average but it is significantly faster (7.8×) and smaller (36×), which makes it a good fit for resource-constrained settings. Additionally, we show that SMaLL-100 can achieve similar performance as M2M-100 (12B) with just a few steps of fine-tuning on specific language pairs.

# Appendix

## 6.7 Details of Evaluation benchmarks

### 6.7.1 Resource-type of Languages in Evaluated Datasets

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| af | Low | lg | Very-Low | lt | Medium | sn | Low | gl | Medium |
| am | Low | ka | Medium | luo | Low | sd | Very-Low | gd | Low |
| ar | Medium | de | High | lb | Medium | sk | Medium | ht | Low |
| hy | Low | el | Medium | mk | Medium | sl | Medium | su | Low |
| as | Very-Low | gu | Low | ms | Low | so | Low | ln | Very-Low |
| ast | Low | ha | Low | ml | Low | ku | Low | ilo | Low |
| az | Low | he | Medium | mt | Medium | es | High | mg | Medium |
| be | Very-Low | hi | Medium | mr | Low | sw | Low | tn | Very-Low |
| bn | Medium | hu | Medium | mi | Low | sv | Medium | br | Medium |
| bs | Low | is | Medium | mn | Low | tg | Low | ns | Very-Low |
| bg | Medium | ig | Low | ne | Very-Low | ta | Low | si | Medium |
| my | Low | id | Medium | nso | Very-Low | te | Low | yi | Low |
| ca | Medium | ga | Low | no | Medium | th | Medium | fy | Medium |
| ceb | Low | it | High | ny | Low | tr | Medium | sq | Medium |
| zh | Medium | ja | Medium | oc | Very-Low | uk | Medium | ss | Very-Low |
| hr | Very-Low | jv | Medium | or | Very-Low | umb | Low | fr | High |
| cs | Medium | kea | Very-Low | om | Low | ur | Low | ff | Very-Low |
| da | Medium | kam | Very-Low | ps | Low | uz | Very-Low | lo | Low |
| nl | Medium | kn | Low | fa | Medium | vi | Medium | lv | Medium |
| en | High | kk | Low | pl | Medium | cy | Low | ru | High |
| et | Medium | km | Low | pt | High | wo | Very-Low | sr | Medium |
| tl | Very-Low | ko | Medium | pa | Low | xh | Low | zu | Low |
| fi | Medium | ky | Low | ro | Medium | yo | Low | ba | Low |

Table 6.4: ISO-639 code and resource type of languages used in evaluated NMT benchmarks.

### 6.7.2 FLORES-101

We use `devtest` subset of FLORES-101 for the evaluation. To better compare different models, we exclude evaluation of language pairs, in which the spBLEU performance of M2M-100 12B (Fan et al., 2020a) model is below 3. This gives 5,934 language directions for the comparison.

Table 6.5 shows the distribution of different categories of language pairs.

| | VL2VL | VL2L | VL2M | VL2H | L2VL | L2L | L2M | L2H | M2VL | M2L | M2M | M2H | H2VL | H2L | H2M | H2H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No. lang. pairs | 44 | 200 | 388 | 81 | 144 | 645 | 960 | 186 | 181 | 992 | 1330 | 259 | 35 | 190 | 257 | 42 |

Table 6.5: Distribution of resource categories for different language directions on FLORES-101 (Goyal et al., 2021b).

### 6.7.3 Tatoeba Challenge

We use the `test` subset data, provided by Tiedemann (2020)[12] to evaluate all models. We choose a subset of dataset that includes languages which are covered by M2M-100 (Fan et al., 2020a) model. This brings 1,844 language pairs for the evaluation. The distribution of different language pair categories is shown in Table 6.6.

| | VL2L | VL2M | VL2H | L2VL | L2L | L2M | L2H | M2VL | M2L | M2M | M2H | H2VL | H2L | H2M | H2H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No. lang. pairs | 7 | 30 | 37 | 7 | 34 | 144 | 113 | 30 | 144 | 632 | 237 | 37 | 113 | 237 | 42 |

Table 6.6: Distribution of resource categories for different language directions on Tatoeba Challenge.

### 6.7.4 TICO-19

We use the evaluation benchmark provided by Anastasopoulos et al. (2020)[13] to compare all models in multilingual medical domain. We utilise language pairs that are included in M2M-100 (Fan et al., 2020a) model. This gives us 650 language pairs for the evaluation. Table 6.7 shows the number of language pairs in different resource types of language directions.

| | VL2VL | VL2L | VL2M | VL2H | L2VL | L2L | L2M | L2H | M2VL | M2L | M2M | M2H | H2VL | H2L | H2M | H2H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No. lang. pairs | 12 | 48 | 24 | 16 | 48 | 132 | 72 | 48 | 24 | 72 | 30 | 24 | 16 | 48 | 24 | 12 |

Table 6.7: Number of language pairs for different categories of language directions on TICO-19.

## 6.8 Hyper-Parameters for Architecture and Optimisation

[12]`https://github.com/Helsinki-NLP/Tatoeba-Challenge`
[13]`https://tico-19.github.io/`

| Hyper-Parameter | Specification | Hyper-Parameter | Specification |
|---|---|---|---|
| Encoder Layer | 12 | Scheduler | inverse-sqrt |
| Decoder Layer | 3 | Optimizer | Adam |
| Encoder Emb dim. | 1024 | Clip Norm | 1.0 |
| Decoder Emb dim | 1024 | Learning Rate | 1e-4 |
| Encoder FFN Emb dim. | 4096 | Warmup init LR | 1e-07 |
| Decoder FFN Emb dim. | 4096 | Warmup Updates | 40K |
| Number of attn heads | 16 | Adam Betas | 0.9,0.98 |
| Attention Dropout | 0.1 | Adam eps | 1e-6 |
| Share Encoder/Decoder Emb. | True | Label Smoothing | 0.1 |
| FP16 | True | Dropout | 0.1 |
| Loss Scalar | 2.0 | Max Tokens (per GPU) | 1,000 |

Table 6.8: List of hyper-parameters used for the architecture, and optimisation.

## 6.9 spBLEU Results

### 6.9.1 spBLEU of Remaining Language Directions

| | | | Language Direction | | | |
|---|---|---|---|---|---|---|
| Model | params | Speed | M2M | M2H | H2M | H2H |
| **FLORES-101** | | | | | | |
| FLORES-124 | 175M | 5.3× | 13.7 | 18.0 | 16.8 | 23.0 |
| M2M-100 | 418M | 3.1× | 18.1 | 23.0 | 21.6 | 27.8 |
| FLORES-124 | 615M | 2.9× | 19.3 | 24.1 | 22.7 | 28.9 |
| *Finetuned-100* | 330M | 7.8× | 16.4 | 21.2 | 19.7 | 26.0 |
| *SMaLL-100* | 330M | 7.8× | 19.3 | 24.2 | 22.6 | 28.8 |
| M2M-100 | 1.2B | 1.8× | 22.3 | 28 | 25.8 | 32.7 |
| M2M-100 | 12B | 1× | 23.9 | 29.5 | 27.6 | 34.3 |
| **Tatoeba** | | | | | | |
| FLORES-124 | 175M | 5.3× | 25.2 | 28.1 | 24.5 | 35.3 |
| M2M-100 | 418M | 3.1× | 29.6 | 34.0 | 29.6 | 41.6 |
| FLORES-124 | 615M | 2.9× | 31.7 | 35.5 | 31.0 | 43.0 |
| *Finetuned-100* | 330M | 7.8× | 28.3 | 34.2 | 28.1 | 39.2 |
| *SMaLL-100* | 330M | 7.8× | 31.9 | 36.3 | 31.1 | 42.4 |
| M2M-100 | 1.2B | 1.8× | 33.8 | 39.0 | 34.2 | 47.4 |
| M2M-100 | 12B | 1× | 33.1 | 39.0 | 34.2 | 48.7 |
| **TICO19** | | | | | | |
| FLORES-124 | 175M | 5.3× | 15.1 | 20.3 | 17.7 | 28.1 |
| M2M-100 | 418M | 3.1× | 20.6 | 26.6 | 23.6 | 33.1 |
| FLORES-124 | 615M | 2.9× | 20.2 | 27.0 | 23.3 | 33.9 |
| *Finetuned-100* | 330M | 7.8× | 19.7 | 24.4 | 23.4 | 31 |
| *SMaLL-100* | 330M | 7.8× | 21.7 | 27.4 | 25.2 | 33.7 |
| M2M-100 | 1.2B | 1.8× | 21.1 | 30.2 | 24.8 | 37.7 |
| M2M-100 | 12B | 1× | 24.8 | 33.1 | 28.3 | 39.4 |

Table 6.9: Average spBLEU performance of different models on FLORES-101, Tatoeba, and TICO-19 benchmarks for medium and high-resource language pair categories, defined in Section 6.4. The FLORES-101 results are computed on language pairs where M2M-100 12B has spBLEU scores higher than 3 to avoid polluting the analysis with meaningless scores. The first and second columns give the model size and speed-up ratios compared to M2M-100 (12B).

### 6.9.2 FLORES-101

[illegible]

Table 6.10: spBLEU performance of last checkpoint of SMaLL-100 model on language pairs of FLORES-101.

### 6.9.3 Tatoeba

[illegible]

Table 6.11: spBLEU performance of last checkpoint of SMaLL-100 model on language pairs of Tatoeba.

### 6.9.4 TICO19

| src \ tgt | am | ar | bn | my | zh | en | tl | fr | lg | ha | hi | id | km | ln | ms | mr | ne | ps | fa | ru | so | es | sw | ta | ur | zu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| am | -1 | 7.5 | 12.7 | 1.5 | 7.6 | 18.3 | 11.3 | 13.7 | 2.4 | 6.9 | 16.2 | 14.2 | 5.5 | 2.4 | 14.4 | 5.9 | 6.1 | 7.3 | 11.8 | 10.6 | 0.5 | 15.7 | 13.7 | 6 | 10.4 | 5.2 |
| ar | 7.8 | -1 | 16.8 | 1.5 | 12.3 | 29.9 | 3.8 | 23.6 | 1.5 | 7.4 | 24.9 | 26 | 9.8 | 1.8 | 23.9 | 4.2 | 0.3 | 6.4 | 20.5 | 18.8 | 0.4 | 29.5 | 21.3 | 1.7 | 13.3 | 2.5 |
| bn | 9.7 | 15.5 | -1 | 4.4 | 13.8 | 33.8 | 23.9 | 21.6 | 3.9 | 10.7 | 31.1 | 27.9 | 12.6 | 4.4 | 27.3 | 11.3 | 16.4 | 10.4 | 21.1 | 19.4 | 1.6 | 27.9 | 22.7 | 9.9 | 18.3 | 9.9 |
| my | 6.7 | 6.3 | 11.4 | -1 | 7.9 | 16.6 | 14.2 | 11.2 | 3.2 | 7.9 | 14.7 | 14.3 | 7.9 | 2.8 | 14.8 | 5.6 | 8 | 7.9 | 11.5 | 10.2 | 1.3 | 14.4 | 13.7 | 6.1 | 11 | 7.1 |
| zh | 8.6 | 15.6 | 18.3 | 4 | -1 | 27.5 | 20.6 | 20.9 | 3.6 | 9.3 | 24.3 | 26.5 | 10.6 | 4 | 24.5 | 8.5 | 6.8 | 9.2 | 19.9 | 19 | 1.4 | 26.6 | 20.9 | 5.3 | 15.2 | 9 |
| en | 11.1 | 25.2 | 26.8 | 7 | 19.6 | -1 | 35.6 | 36.3 | 5.6 | 15.4 | 39.3 | 47 | 17.9 | 3.6 | 44.7 | 12.1 | 19 | 11.8 | 29.9 | 29.7 | 2.1 | 47.5 | 32.4 | 9.3 | 20.8 | 15.7 |
| tl | 10.1 | 15.2 | 21.4 | 6 | 14.6 | 45.4 | -1 | 26.1 | 5.7 | 14 | 28.5 | 33.9 | 14.7 | 6 | 33.2 | 10.1 | 14.6 | 10.6 | 21.9 | 22.5 | 1.6 | 33.3 | 26.2 | 7 | 17 | 14.6 |
| fr | 8.5 | 18.4 | 17.8 | 3.8 | 13.9 | 35.1 | 22.3 | -1 | 3.5 | 9.7 | 26.1 | 30.7 | 11.6 | 3.9 | 27.6 | 8.3 | 2.6 | 7.9 | 20.8 | 22.5 | 1.1 | 34.3 | 21.3 | 1.7 | 14.3 | 8.8 |
| lg | 4.1 | 3.1 | 6.5 | 2.6 | 4.2 | 14.1 | 10.8 | 8.5 | -1 | 7.4 | 7.2 | 10.2 | 3.9 | 4.6 | 9.8 | 3.9 | 4.5 | 4 | 5.7 | 8.7 | 1.5 | 11.3 | 9.7 | 2.4 | 5.2 | 6.8 |
| ha | 5.9 | 6.4 | 10.3 | 3.2 | 6.4 | 20.7 | 16.8 | 12.6 | 3.9 | -1 | 13.5 | 16.2 | 8.1 | 4.8 | 16 | 5 | 6.8 | 7.2 | 11.7 | 11.7 | 1.4 | 16.6 | 14.7 | 4.3 | 9.8 | 8.3 |
| hi | 10.5 | 18 | 25.4 | 4.6 | 15.3 | 40.8 | 26.3 | 25 | 4.2 | 11.4 | -1 | 31.9 | 13.5 | 4.5 | 31 | 13.5 | 20.4 | 11 | 23.6 | 22.1 | 1.6 | 32.6 | 25.5 | 10.4 | 21.4 | 10.5 |
| id | 9.7 | 19.5 | 22.2 | 4.8 | 17.4 | 43 | 22.9 | 28.6 | 4.7 | 12.7 | 30.2 | -1 | 14.3 | 5.4 | 39.1 | 7.4 | 0.7 | 10.3 | 25.6 | 25 | 1.9 | 36.5 | 27.5 | 3.9 | 17.5 | 11.7 |
| km | 7.6 | 10.8 | 14.6 | 3.2 | 10 | 25 | 19.4 | 17 | 3.8 | 10.4 | 20 | 22.5 | -1 | 4.1 | 22.2 | 6.8 | 9.4 | 8.8 | 15.9 | 14.7 | 1 | 21.5 | 18.9 | 6.6 | 13.1 | 8.6 |
| ln | 3.6 | 3.4 | 5.9 | 2.1 | 4.2 | 12 | 9.2 | 8.1 | 4.2 | 6.1 | 6.8 | 10.9 | 3 | -1 | 10.1 | 3.4 | 3.4 | 4.3 | 5.9 | 8.2 | 1.2 | 10.7 | 9.4 | 2 | 5.2 | 6.3 |
| ms | 10.1 | 19.2 | 22.6 | 5 | 16.3 | 44.2 | 28 | 27.6 | 4.8 | 13.2 | 30.8 | 41.4 | 14.7 | 5.5 | -1 | 8.9 | 1.1 | 10.6 | 25.2 | 23.8 | 1.9 | 35.2 | 28.2 | 6.6 | 17.8 | 12.4 |
| mr | 7.9 | 10.8 | 17.7 | 2.6 | 9.8 | 23.9 | 18.3 | 16.4 | 2.6 | 8.2 | 24.1 | 19.2 | 9.5 | 2.8 | 19.7 | -1 | 12.3 | 8 | 15.8 | 14.1 | 0.9 | 20.4 | 16.4 | 8 | 14.9 | 7.3 |
| ne | 9.5 | 8.6 | 21.9 | 4 | 12.2 | 33.1 | 22.1 | 19.1 | 3.9 | 9.9 | 31.7 | 20.8 | 11.1 | 4.3 | 24.9 | 11.4 | -1 | 10.5 | 14.8 | 17.3 | 1.7 | 25.1 | 20.4 | 8.4 | 17.4 | 9.5 |
| ps | 8 | 10.9 | 16.2 | 3.4 | 9.9 | 23.4 | 18.3 | 15.5 | 3.4 | 8.9 | 20.7 | 19.4 | 10 | 3.8 | 19.2 | 7.8 | 11.7 | -1 | 17.1 | 13.9 | 1.4 | 20.1 | 17.1 | 7.6 | 14.7 | 8 |
| fa | 8.8 | 18.2 | 19.3 | 3.3 | 14.6 | 32 | 10.8 | 23.5 | 3.6 | 10.1 | 26 | 29 | 11.8 | 4 | 26.8 | 6.2 | 0.9 | 9.5 | -1 | 20.9 | 1.2 | 29.7 | 22.6 | 3.5 | 15.8 | 7 |
| ru | 9.1 | 18.2 | 20.2 | 3.6 | 15.7 | 33.1 | 23 | 25.9 | 4 | 10.5 | 26.4 | 30 | 11.8 | 4.6 | 26.8 | 9.4 | 12.9 | 9.3 | 22 | -1 | 1.4 | 32.7 | 22.6 | 4.6 | 15.9 | 10.5 |
| so | 1.1 | 0.2 | 1.1 | 0.9 | 0.3 | 2.1 | 3.4 | 1 | 1.6 | 3.3 | 1.8 | 1.2 | 1.4 | 1.5 | 1.5 | 0.8 | 0.8 | 2.4 | 0.6 | 0.3 | -1 | 2.2 | 3.4 | 0.6 | 1.2 | 3 |
| es | 9.7 | 22.2 | 22.4 | 4.8 | 17.6 | 45.6 | 27.3 | 33.7 | 4.2 | 12.2 | 31.8 | 38 | 13.5 | 5.1 | 33.5 | 9.3 | 3.5 | 9.7 | 26 | 27.7 | 1.7 | -1 | 26.5 | 2.4 | 17.7 | 9.8 |
| sw | 9.2 | 15.3 | 18.9 | 4.9 | 13 | 33.5 | 24.1 | 22.4 | 4.8 | 12.8 | 24.9 | 29.2 | 13.5 | 5.9 | 29.3 | 8.4 | 6.6 | 10.1 | 20.8 | 19.7 | 1.4 | 28.4 | -1 | 6.3 | 16.1 | 8.1 |
| ta | 6 | 6.6 | 12 | 1.2 | 6.3 | 18.1 | 12.4 | 10.5 | 1.5 | 6.4 | 16.8 | 13.3 | 5.5 | 2.2 | 13.4 | 5.9 | 5.8 | 6.8 | 11 | 9.4 | 0.5 | 13.8 | 12.4 | -1 | 10.9 | 4.3 |
| ur | 8.7 | 12.8 | 19.4 | 3.1 | 11.6 | 26.6 | 20 | 18 | 3.3 | 9.6 | 26.7 | 22.5 | 9.4 | 3.6 | 22.2 | 9.4 | 12.4 | 10 | 18.6 | 15.9 | 1.4 | 23.2 | 19.6 | 8.4 | -1 | 7.5 |
| zu | 7.3 | 9.4 | 13.8 | 4.3 | 9.3 | 27.3 | 21.9 | 15.9 | 4.7 | 11.3 | 17.3 | 20.8 | 10.3 | 5.3 | 20.7 | 7 | 9.3 | 8.5 | 15.4 | 14.9 | 1.2 | 21.5 | 18.3 | 5.6 | 12.4 | -1 |

Table 6.12: spBLEU performance of last checkpoint of SMaLL-100 model on language pairs of TICO19.

## 6.10 Details of Fine-Tuning on Low-Resource Language Pairs

For further fine-tuning of SMaLL-100 and M2M-100 (418M) models on selected language pairs, we use bilingual data provided by Tiedemann (2020) (release 2021.08.07)[14] as its training data is less noisy. We evaluate fine-tuned models on `devtest` subset of FLORES-101 (Goyal et al., 2021b) benchmark with spBLEU metric (Goyal et al., 2021b). We use the same hyper-parameters, as defined in Appendix 6.8. We train each model on 2 TESLA V100-32GB GPUs.

[14] `https://github.com/Helsinki-NLP/Tatoeba-Challenge/blob/master/data/README-v2021-08-07.md`

# 7 Conclusions and Future Work

## 7.1 General Conclusions

In this thesis, we addressed two main problems of Transformer-based architectures including the inability to encode structured data (e.g. graphs) and the computational efficiency. Particularly, we focus on encoding syntactic and semantic graphs into the self-attention mechanism of Transformers. Additionally, we concentrate on the impacts of applying compression techniques to pre-trained Transformer-based multilingual NMT models. In the following, we demonstrate our main conclusions.

In Chapter 2, we proposed the Graph-to-Graph Transformer (G2GTr) architecture, which inputs and outputs arbitrary graphs through its attention mechanisms. Each graph relation is modelled as a label embedding to each attention function involving the relation's tokens, and each graph relation is predicted from its token's embeddings like an attention function. We demonstrate the effectiveness of this architecture on transition-based dependency parsing, where the input graph is the partial dependency structure specified by the parse history, and the output graph is predicted one dependency at a time by the parser actions. Incorporating BERT (Brown et al., 2020) pre-training results in substantially improving the state-of-the-art in traditional transition-based dependency parsing.

In Chapter 3, we propose a novel model for structured prediction, Recursive Non-autoregressive Graph-to-Graph Transformer (RNG Transformer), to iteratively refine arbitrary graphs. Given an initial graph, RNG Transformer learns to predict a corrected graph over the same set of nodes. Each iteration of refinement predicts the edges of the graph in a non-autoregressive fashion, but conditions these predictions on the entire graph from the previous iteration. This graph conditioning and prediction are made with the G2GTr architecture, which can capture complex patterns of inter-dependencies between graph edges. Evaluating on 13 languages of the Universal Dependencies Treebanks, the English and Chinese Penn Treebanks, and the German CoNLL 2009 shared task treebank, our RNG Transformer model (integrated with different initial parsers) significantly outperforms previous state-of-the-art models on all these treebanks.

In Chapter 4, we propose the Syntax-aware Graph-to-Graph Transformer architecture, which effectively incorporates syntactic information by inputting the syntactic dependency graph into the self-attention mechanism of Transformer. The mechanism for inputting graph relation embeddings differs from the original Graph-to-Graph Transformer (defined in Chapter 2 and Chapter 3) in that it models the complete interaction between the dependency relation, query vector and key vector. It also excludes the graph interaction with value vectors while maintaining good performance. We have evaluated our model on CoNLL 2005 and CoNLL 2009 SRL datasets and outperformed previous comparable models.

In Chapter 5, we demonstrated the impacts of applying compression methods to the massively Multilingual Machine Translation models by evaluating compressed models on FLORES-101 (Goyal et al., 2021b), gender bias benchmark (Stanovsky et al., 2019), and word sense disambiguation benchmark (Campolungo et al., 2022). We showed that while average BLEU drops negligibly, the performance of under-represented language pairs drops drastically. By analysing the attention patterns, we showed that sparsity improves the performance of some medium-resource language pairs by removing the noisy memorisation, resulting in less hallucinations in the target translations. By evaluating our compressed models on gender bias and word sense disambiguation benchmarks, we showed that the compression amplifies the intrinsic gender and semantic biases, even in high-resource language pairs.

In Chpater 6, we presented SMaLL-100 model, a shallow multilingual NMT model, focusing on low-resource languages. We evaluated our model on different NMT benchmarks. SMaLL-100 significantly outperforms multilingual models of comparable size on all of the tested benchmarks (FLORES-101, Tatoeba, TICO-19) and is much faster at inference. It also achieves competitive results with M2M-100 1.2B (Fan et al., 2020a), while being 4.3× faster at inference and 3.6× smaller. Compared to M2M-100 (12B), the biggest available MNMT model, SMaLL-100 loses nearly 1.7 spBLEU on average but it is significantly faster (7.8×) and smaller (36×), which makes it a good fit for resource-constrained settings. Additionally, we show that SMaLL-100 can achieve similar performance as M2M-100 (12B) with just a few steps of fine-tuning on specific language pairs.

## 7.2 Future Work

While the research demonstrated in this thesis shows significant improvement in the generality and efficiency of Transformer-based models, there is a vast scope for additional refinement. Below, we outline some potential research directions for future work.

- Regarding our Graph-to-Graph Transformer architecture, it can easily be applied to a wide variety of NLP tasks, which require encoding graph structures, including other semantic (e.g. AMR parsing) and syntactic (e.g. constituency graph) graphs, and knowledge graphs. Additionally, future studies can integrate our Graph-to-Graph Transformer model with efficient Transformer architectures (Katharopoulos et al., 2020; Wang et al., 2020a;

Ainslie et al., 2020).

- The RNG Transformer architecture is a very general and powerful method for structured prediction, which could easily be applied to other NLP tasks. It would especially benefit tasks that require capturing complex structured inter-dependencies between graph edges, without losing the computational benefits of a non-autoregressive model.
- Regarding the impacts of compression on Transformer-based models, we hope our findings could be a starting point to consider the fairness aspects when compressing multilingual models. Additionally, our SMaLL-100 model can be used as a compact and efficient NMT model for a great number of languages, including low and very-low language directions.

# Bibliography

Aharoni, R., Johnson, M., and Firat, O. (2019). Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3874–3884, Minneapolis, Minnesota. Association for Computational Linguistics.

Ahia, O., Kreutzer, J., and Hooker, S. (2021). The low-resource double bind: An empirical study of pruning for low-resource machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3316–3333, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ainslie, J., Ontanon, S., Alberti, C., Cvicek, V., Fisher, Z., Pham, P., Ravula, A., Sanghai, S., Wang, Q., and Yang, L. (2020). ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284, Online. Association for Computational Linguistics.

Allen, J. F. and Perrault, C. (1980). Analyzing intention in utterances. *Artificial Intelligence*, 15(3):143–178.

Anastasopoulos, A., Cattelan, A., Dou, Z.-Y., Federico, M., Federmann, C., Genzel, D., Guzmán, F., Hu, J., Hughes, M., Koehn, P., Lazar, R., Lewis, W., Neubig, G., Niu, M., Öktem, A., Paquin, E., Tang, G., and Tur, S. (2020). TICO-19: the translation initiative for COvid-19. In *Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020*, Online. Association for Computational Linguistics.

Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., and Collins, M. (2016). Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Association for Computational Linguistics.

Arivazhagan, N., Bapna, A., Firat, O., Lepikhin, D., Johnson, M., Krikun, M., Chen, M. X., Cao, Y., Foster, G., Cherry, C., Macherey, W., Chen, Z., and Wu, Y. (2019). Massively multilingual neural machine translation in the wild: Findings and challenges.

Attardi, G. and Ciaramita, M. (2007). Tree revision learning for dependency parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of* *the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 388–395, Rochester, New York. Association for Computational Linguistics.

Awasthi, A., Sarawagi, S., Goyal, R., Ghosh, S., and Piratla, V. (2019). Parallel iterative edit models for local sequence transduction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4251–4261.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization.

Bai, H., Zhang, W., Hou, L., Shang, L., Jin, J., Jiang, X., Liu, Q., Lyu, M., and King, I. (2021). BinaryBERT: Pushing the limit of BERT quantization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4334–4348, Online. Association for Computational Linguistics.

Ballesteros, M. and Bohnet, B. (2014). Automatic feature selection for agenda-based dependency parsing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 794–805, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Ballesteros, M., Dyer, C., Goldberg, Y., and Smith, N. A. (2017). Greedy Transition-Based Dependency Parsing with Stack LSTMs. *Computational Linguistics*, 43(2):311–347.

Ballesteros, M., Goldberg, Y., Dyer, C., and Smith, N. A. (2016). Training with exploration improves a greedy stack LSTM parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2005–2010, Austin, Texas. Association for Computational Linguistics.

Barry, J., Mohammadshahi, A., Wagner, J., Foster, J., and Henderson, J. (2021). The DCU-EPFL enhanced dependency parser at the IWPT 2021 shared task. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 204–212, Online. Association for Computational Linguistics.

Behnke, M. and Heafield, K. (2020). Losing heads in the lottery: Pruning transformer attention in neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2664–2674, Online. Association for Computational Linguistics.

Bentivogli, L., Savoldi, B., Negri, M., Di Gangi, M. A., Cattoni, R., and Turchi, M. (2020). Gender in danger? evaluating speech translation technology on the MuST-SHE corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6923–6933, Online. Association for Computational Linguistics.

Berard, A., Lee, D., Clinchant, S., Jung, K., and Nikoulina, V. (2021). Efficient inference for multilingual neural machine translation. In *Proceedings of the 2021 Conference on*

*Empirical Methods in Natural Language Processing*, pages 8563–8583, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Black, E., Jelinek, F., Lafferty, J., Magerman, D. M., Mercer, R., and Roukos, S. (1992). Towards history-based grammars: Using richer models for probabilistic parsing. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*.

Boito, M. Z., Besacier, L., Tomashenko, N., and Estève, Y. (2022). A study of gender impact in self-supervised models for speech-to-text systems.

Bondarenko, Y., Nagel, M., and Blankevoort, T. (2021). Understanding and overcoming the challenges of efficient transformer quantization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7947–7969, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Cai, J., He, S., Li, Z., and Zhao, H. (2018). A full end-to-end semantic role labeler, syntactic-agnostic over syntactic-aware? In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2753–2765, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Cai, R. and Lapata, M. (2019a). Semi-supervised semantic role labeling with cross-view training. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1018–1027, Hong Kong, China. Association for Computational Linguistics.

Cai, R. and Lapata, M. (2019b). Syntax-aware semantic role labeling without parsing. *Transactions of the Association for Computational Linguistics*, 7:343–356.

Campolungo, N., Martelli, F., Saina, F., and Navigli, R. (2022). DiBiMT: A novel benchmark for measuring Word Sense Disambiguation biases in Machine Translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4331–4352, Dublin, Ireland. Association for Computational Linguistics.

Carnap, R. (1947). *Meaning and Necessity: A Study in Semantics and Modal Logic*. Chicago, IL, USA: University of Chicago Press.

Carreras, X. (2007). Experiments with a higher-order projective dependency parser. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and*

*Computational Natural Language Learning (EMNLP-CoNLL)*, pages 957–961, Prague, Czech Republic. Association for Computational Linguistics.

Carreras, X. and Màrquez, L. (2005). Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan. Association for Computational Linguistics.

Chen, D. and Manning, C. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Chen, H., Huang, S., Chiang, D., and Chen, J. (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).*

Chen, T., Frankle, J., Chang, S., Liu, S., Zhang, Y., Wang, Z., and Carbin, M. (2020). The lottery ticket hypothesis for pre-trained bert networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15834–15846. Curran Associates, Inc.

Chen, W., Zhang, Y., and Zhang, M. (2014). Feature embedding for dependency parsing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 816–826, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Chen, X., Lyu, C., and Titov, I. (2019). Capturing argument interaction in semantic role labeling with capsule networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5415–5425, Hong Kong, China. Association for Computational Linguistics.

Cheng, H., Fang, H., He, X., Gao, J., and Deng, L. (2016). Bi-directional attention with agreement for dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2204–2214, Austin, Texas. Association for Computational Linguistics.

Chi, Z. (1999). Statistical properties of probabilistic context-free grammars. *Computational Linguistics*, 25(1):131–160.

Chomsky, N. (2002). *Syntactic structures*. Mouton de Gruyter.

Conia, S. and Navigli, R. (2020). Bridging the gap in multilingual semantic role labeling: a language-agnostic approach. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1396–1410, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Cross, J. and Huang, L. (2016). Incremental parsing with minimal features using bi-directional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 32–37, Berlin, Germany. Association for Computational Linguistics.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

de Lhoneux, M., Stymne, S., and Nivre, J. (2017). Old school vs. new school: Comparing transition-based parsers with and without neural network enhancement. In *Proceedings of the 15th International Workshop on Treebanks and Linguistic Theories (TLT15)*, pages 99–110.

de Vassimon Manela, D., Errington, D., Fisher, T., van Breugel, B., and Minervini, P. (2021). Stereotype and skew: Quantifying gender bias in pre-trained and fine-tuned language models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2232–2242, Online. Association for Computational Linguistics.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. (2022). GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

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

Dong, D., Wu, H., He, W., Yu, D., and Wang, H. (2015). Multi-task learning for multiple language translation. In *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)*, pages 1723–1732, Beijing, China. Association for Computational Linguistics.

Dong, X., Chen, S., and Pan, S. J. (2017). Learning to prune deep neural networks via layer-wise optimal brain surgeon. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4860–4874, Red Hook, NY, USA. Curran Associates Inc.

Dozat, T. and Manning, C. D. (2016). Deep biaffine attention for neural dependency parsing.

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., and Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. In *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)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Edmonds, J. (1967). Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.

Edunov, S., Ott, M., Auli, M., and Grangier, D. (2018). Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Eisner, J. M. (1996). Three new probabilistic models for dependency parsing: An exploration. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

El-Kishky, A., Chaudhary, V., Guzmán, F., and Koehn, P. (2020). CCAligned: A massive collection of cross-lingual web-document pairs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5960–5969, Online. Association for Computational Linguistics.

Espinosa Anke, L., Shvets, A., Mohammadshahi, A., Henderson, J., and Wanner, L. (2022). Multilingual extraction and categorization of lexical collocations with graph-aware transformers. In *Proceedings of the 11th Joint Conference on Lexical and Computational Semantics*, pages 89–100, Seattle, Washington. Association for Computational Linguistics.

Fan, A., Bhosale, S., Schwenk, H., Ma, Z., El-Kishky, A., Goyal, S., Baines, M., Celebi, O., Wenzek, G., Chaudhary, V., Goyal, N., Birch, T., Liptchinsky, V., Edunov, S., Grave, E., Auli, M., and Joulin, A. (2020a). Beyond english-centric multilingual machine translation.

Fan, A., Grave, E., and Joulin, A. (2020b). Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*.

Fei, H., Li, F., Li, B., and Ji, D. (2021). Encoder-decoder based unified semantic role labeling with label-aware syntax. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12794–12802.

Fernández-González, D. and Gómez-Rodríguez, C. (2019). Left-to-right dependency parsing with pointer networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 710–716, Minneapolis, Minnesota. Association for Computational Linguistics.

Fernández-González, D. (2023). Transition-based semantic role labeling with pointer networks. *Knowledge-Based Systems*, 260:110127.

Firat, O., Cho, K., and Bengio, Y. (2016). Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California. Association for Computational Linguistics.

Frankle, J. and Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Gale, T., Elsen, E., and Hooker, S. (2019). The state of sparsity in deep neural networks.

Garcia, X., Siddhant, A., Firat, O., and Parikh, A. (2021). Harnessing multilinguality in unsupervised machine translation for rare languages. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1126–1137, Online. Association for Computational Linguistics.

Gesmundo, A., Henderson, J., Merlo, P., and Titov, I. (2009). A latent variable model of synchronous syntactic-semantic parsing for multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 37–42, Boulder, Colorado. Association for Computational Linguistics.

Gildea, D. and Palmer, M. (2002). The necessity of parsing for predicate argument recognition. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 239–246, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Gou, J., Yu, B., Maybank, S. J., and Tao, D. (2021). Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819.

Goyal, N., Du, J., Ott, M., Anantharaman, G., and Conneau, A. (2021a). Larger-scale transformers for multilingual masked language modeling.

Goyal, N., Gao, C., Chaudhary, V., Chen, P.-J., Wenzek, G., Ju, D., Krishnan, S., Ranzato, M., Guzman, F., and Fan, A. (2021b). The flores-101 evaluation benchmark for low-resource and multilingual machine translation.

Haddow, B., Bawden, R., Barone, A. V. M., Helcl, J., and Birch, A. (2021). Survey of low-resource machine translation.

Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M. A., Màrquez, L., Meyers, A., Nivre, J., Padó, S., Štěpánek, J., Straňák, P., Surdeanu, M., Xue, N., and Zhang, Y. (2009). The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado. Association for Computational Linguistics.

Hall, D., Durrett, G., and Klein, D. (2014). Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–237, Baltimore, Maryland. Association for Computational Linguistics.

Hall, K. and Novák, V. (2005). Corrective modeling for non-projective dependency parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 42–52, Vancouver, British Columbia. Association for Computational Linguistics.

Han, S., Pool, J., Tran, J., and Dally, W. (2015). Learning both weights and connections for efficient neural network. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

He, L., Lee, K., Levy, O., and Zettlemoyer, L. (2018a). Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369, Melbourne, Australia. Association for Computational Linguistics.

He, L., Lee, K., Lewis, M., and Zettlemoyer, L. (2017). Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada. Association for Computational Linguistics.

He, S., Li, Z., and Zhao, H. (2019). Syntax-aware multilingual semantic role labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5350–5359, Hong Kong, China. Association for Computational Linguistics.

He, S., Li, Z., Zhao, H., and Bai, H. (2018b). Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2061–2071, Melbourne, Australia. Association for Computational Linguistics.

Henderson, J. (1990). Structure unification grammar: A unifying framework for investigating natural language. .

Henderson, J. (2003). Inducing history representations for broad coverage statistical parsing. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 103–110.

Henderson, J. (2004). Discriminative training of a neural network statistical parser. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 95–102, Barcelona, Spain.

Henderson, J., Merlo, P., Titov, I., and Musillo, G. (2013). Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics*, 39(4):949–998.

Hennig, F. and Köhn, A. (2017). Dependency tree transformation with tree transducers. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 58–66.

Hermann, K. M. and Blunsom, P. (2013). The role of syntax in vector space models of compositional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computa-*

*tional Linguistics (Volume 1: Long Papers)*, pages 894–904, Sofia, Bulgaria. Association for Computational Linguistics.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network.

Hooker, S., Courville, A., Clark, G., Dauphin, Y., and Frome, A. (2019). What do compressed deep neural networks forget?

Hooker, S., Moorosi, N., Clark, G., Bengio, S., and Denton, E. (2020). Characterising bias in compressed models.

Hu, M., Peng, Y., Wei, F., Huang, Z., Li, D., Yang, N., and Zhou, M. (2018). Attention-guided answer distillation for machine reading comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2077–2086, Brussels, Belgium. Association for Computational Linguistics.

i Chu, Y. and Liu, T. (1965). On the shortest arborescence of a directed graph. In *BIT Numerical Mathematics*.

Ji, T., Jain, S., Ferdman, M., Milder, P., Schwartz, H. A., and Balasubramanian, N. (2021). On the distribution, sparsity, and inference-time quantization of attention values in transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4147–4157, Online. Association for Computational Linguistics.

Ji, T., Wu, Y., and Lan, M. (2019). Graph-based dependency parsing with graph neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2475–2485, Florence, Italy. Association for Computational Linguistics.

Jia, Z., Yan, Z., Wu, H., and Tu, K. (2022). Span-based semantic role labeling with argument pruning and second-order inference. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):10822–10830.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., and Liu, Q. (2020). TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Jindal, I., Aharonov, R., Brahma, S., Zhu, H., and Li, Y. (2020). Improved semantic role labeling using parameterized neighborhood memory adaptation. *ArXiv*, abs/2011.14459.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., Hughes, M., and Dean, J. (2017). Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., and Mikolov, T. (2016a). Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016b). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759.*

Karger, D. R., Klein, P. N., and Tarjan, R. E. (1995). A randomized linear-time algorithm to find minimum spanning trees. *Journal of the ACM (JACM)*, 42(2):321–328.

Kasai, J., Friedman, D., Frank, R., Radev, D., and Rambow, O. (2019). Syntax-aware neural semantic role labeling with supertags. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 701–709, Minneapolis, Minnesota. Association for Computational Linguistics.

Kasai, J., Pappas, N., Peng, H., Cross, J., and Smith, N. (2021). Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In *International Conference on Learning Representations*.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. (2020). Transformers are RNNs: Fast autoregressive transformers with linear attention. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR.

Katz, J. J. and Fodor, J. A. (1963). The structure of a semantic theory. *Language*, 39:170–210.

Kazemi, A., Toral, A., Way, A., Monadjemi, A., and Nematbakhsh, M. (2017). Syntax- and semantic-based reordering in hierarchical phrase-based statistical machine translation. *Expert Systems with Applications*, 84:186–199.

Kim, S., Gholami, A., Yao, Z., Mahoney, M. W., and Keutzer, K. (2021a). I-bert: Integer-only bert quantization. *International Conference on Machine Learning (Accepted).*

Kim, Y. and Rush, A. M. (2016). Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Kim, Z. M., Besacier, L., Nikoulina, V., and Schwab, D. (2021b). Do multilingual neural machine translation models contain language pair specific attention heads? In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2832–2841, Online. Association for Computational Linguistics.

Kiperwasser, E. and Goldberg, Y. (2016). Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Kitaev, N., Cao, S., and Klein, D. (2019). Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy. Association for Computational Linguistics.

Kitaev, N. and Klein, D. (2018). Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Knight, K. and Graehl, J. (2005). An overview of probabilistic tree transducers for natural language processing. In Gelbukh, A., editor, *Computational Linguistics and Intelligent Text Processing. CICLing 2005. Lecture Notes in Computer Science, vol 3406*, pages 1–24. Springer, Berlin, Heidelberg.

Ko, W.-J., El-Kishky, A., Renduchintala, A., Chaudhary, V., Goyal, N., Guzmán, F., Fung, P., Koehn, P., and Diab, M. (2021). Adapting high-resource NMT models to translate low-resource related languages without parallel data. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 802–812, Online. Association for Computational Linguistics.

Kocmi, T., Limisiewicz, T., and Stanovsky, G. (2020). Gender coreference and bias evaluation at WMT 2020. In *Proceedings of the Fifth Conference on Machine Translation*, pages 357–364, Online. Association for Computational Linguistics.

Kondratyuk, D. and Straka, M. (2019). 75 languages, 1 model: Parsing universal dependencies universally. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).*

Koo, T. and Collins, M. (2010). Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11.

Kulmizev, A., de Lhoneux, M., Gontrum, J., Fano, E., and Nivre, J. (2019). Deep contextualized word embeddings in transition-based and graph-based dependency parsing - a tale of two parsers revisited. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2755–2768, Hong Kong, China. Association for Computational Linguistics.

Kuncoro, A., Ballesteros, M., Kong, L., Dyer, C., and Smith, N. A. (2016). Distilling an ensemble of greedy dependency parsers into one MST parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1744–1753, Austin, Texas. Association for Computational Linguistics.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2020). Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Lauscher, A., Lueken, T., and Glavaš, G. (2021). Sustainable modular debiasing of language models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4782–4797, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Lee, C., Kim, Y.-B., Lee, D., and Lim, H. (2018a). Character-level feature extraction with densely connected networks. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3228–3239, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Lee, J., Mansimov, E., and Cho, K. (2018b). Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium. Association for Computational Linguistics.

Li, B., Wang, Z., Liu, H., Du, Q., Xiao, T., Zhang, C., and Zhu, J. (2021a). Learning light-weight translation models from deep transformer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13217–13225.

Li, Z., Cai, J., He, S., and Zhao, H. (2018). Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Li, Z., He, S., Zhao, H., Zhang, Y., Zhang, Z., Zhou, X., and Zhou, X. (2019). Dependency or span, end-to-end uniform semantic role labeling.

Li, Z., Zhao, H., He, S., and Cai, J. (2021b). Syntax role for neural semantic role labeling. *Computational Linguistics*, 47(3):529–574.

Li, Z., Zhao, H., and Parnow, K. (2020). Global greedy dependency parsing. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8319–8326. AAAI Press.

Liang, T., Glossner, J., Wang, L., Shi, S., and Zhang, X. (2021). Pruning and quantization for deep neural network acceleration: A survey.

Lichtarge, J., Alberti, C., Kumar, S., Shazeer, N., and Parmar, N. (2018). Weakly supervised grammatical error correction using iterative decoding. *arXiv preprint arXiv:1811.01710*.

Louizos, C., Welling, M., and Kingma, D. P. (2018). Learning sparse neural networks through l0 regularization. In *International Conference on Learning Representations*.

Lyu, C., Cohen, S. B., and Titov, I. (2019). Semantic role labeling with iterative structure refinement. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1071–1082, Hong Kong, China. Association for Computational Linguistics.

Ma, X. and Hovy, E. (2017). Neural probabilistic model for non-projective MST parsing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 59–69, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Ma, X., Hu, Z., Liu, J., Peng, N., Neubig, G., and Hovy, E. (2018). Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414, Melbourne, Australia. Association for Computational Linguistics.

Ma, X. and Zhao, H. (2012). Fourth-order dependency parsing. In *Proceedings of COLING 2012: Posters*, pages 785–796, Mumbai, India. The COLING 2012 Organizing Committee.

Manning, C. and Schutze, H. (1999). *Foundations of statistical natural language processing*. MIT press.

Marcheggiani, D., Frolov, A., and Titov, I. (2017). A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 411–420, Vancouver, Canada. Association for Computational Linguistics.

Marcheggiani, D. and Titov, I. (2017). Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen, Denmark. Association for Computational Linguistics.

Marcheggiani, D. and Titov, I. (2020). Graph convolutions over constituent trees for syntax-aware semantic role labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3915–3928, Online. Association for Computational Linguistics.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

McDonald, R., Crammer, K., and Pereira, F. (2005a). Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98, Ann Arbor, Michigan. Association for Computational Linguistics.

McDonald, R. and Nivre, J. (2011). Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(1):197–230.

McDonald, R. and Pereira, F. (2006). Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–88, Trento, Italy. Association for Computational Linguistics.

McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. (2005b). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Menghani, G. (2021). Efficient deep learning: A survey on making deep learning models smaller, faster, and better.

Michael H. Zhu, S. G. (2018). To prune, or not to prune: Exploring the efficacy of pruning for model compression.

Michel, P., Levy, O., and Neubig, G. (2019). Are sixteen heads really better than one? In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. J. (1990). Introduction to WordNet: An On-line Lexical Database*. *International Journal of Lexicography*, 3(4):235–244.

Mohammadshahi, A. and Henderson, J. (2020). Graph-to-graph transformer for transition-based dependency parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3278–3289, Online. Association for Computational Linguistics.

Mohammadshahi, A. and Henderson, J. (2021a). Recursive Non-Autoregressive Graph-to-Graph Transformer for Dependency Parsing with Iterative Refinement. *Transactions of the Association for Computational Linguistics*, 9:120–138.

Mohammadshahi, A. and Henderson, J. (2021b). Syntax-aware graph-to-graph transformer for semantic role labelling.

Mohammadshahi, A., Lebret, R., and Aberer, K. (2019). Aligning multilingual word embeddings for cross-modal retrieval task. In *Proceedings of the Beyond Vision and LANguage: inTEgrating Real-world kNowledge (LANTERN)*, pages 11–17, Hong Kong, China. Association for Computational Linguistics.

Mohammadshahi, A., Nikoulina, V., Berard, A., Brun, C., Henderson, J., and Besacier, L. (2022a). SMaLL-100: Introducing shallow multilingual machine translation model for low-resource languages. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8348–8359, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mohammadshahi, A., Nikoulina, V., Berard, A., Brun, C., Henderson, J., and Besacier, L. (2022b). What do compressed multilingual machine translation models forget? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4308–4329, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mohammadshahi, A., Scialom, T., Yazdani, M., Yanki, P., Fan, A., Henderson, J., and Saeidi, M. (2022c). Rquge: Reference-free metric for evaluating question generation by answering the question.

Munir, K., Zhao, H., and Li, Z. (2021). Adaptive convolution for semantic role labeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:782–791.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA. Omnipress.

Neubig, G. and Hu, J. (2018). Rapid adaptation of neural machine translation to new languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 875–880, Brussels, Belgium. Association for Computational Linguistics.

Nilsson, J. and Nivre, J. (2008). MaltEval: an evaluation and visualization tool for dependency parsing. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Nivre, J. (2003). An efficient algorithm for projective dependency parsing. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 149–160, Nancy, France.

Nivre, J. (2004). Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain. Association for Computational Linguistics.

Nivre, J. (2009). Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359, Suntec, Singapore. Association for Computational Linguistics.

Nivre, J., Abrams, M., Agić, Ž., Ahrenberg, L., Antonsen, L., Aplonova, K., Aranzabe, M. J., Arutie, G., Asahara, M., Ateyah, L., Attia, M., Atutxa, A., Augustinus, L., Badmaeva, E., Ballesteros, M., Banerjee, E., Bank, S., Barbu Mititelu, V., Basmov, V., Bauer, J., Bellato, S., Bengoetxea, K., Berzak, Y., Bhat, I. A., Bhat, R. A., Biagetti, E., Bick, E., Blokland, R., Bobicev, V., Börstell, C., Bosco, C., Bouma, G., Bowman, S., Boyd, A., Burchardt, A., Candito, M., Caron, B., Caron, G., Cebiroğlu Eryiğit, G., Cecchini, F. M., Celano, G. G. A., Čéplö, S., Cetin, S., Chalub, F., Choi, J., Cho, Y., Chun, J., Cinková, S., Collomb, A., Çöltekin, Ç., Connor, M., Courtin, M., Davidson, E., de Marneffe, M.-C., de Paiva, V., Diaz de Ilarraza, A., Dickerson, C., Dirix, P., Dobrovoljc, K., Dozat, T., Droganova, K., Dwivedi, P., Eli, M., Elkahky, A., Ephrem, B., Erjavec, T., Etienne, A., Farkas, R., Fernandez Alcalde, H., Foster, J., Freitas, C., Gajdošová, K., Galbraith, D., Garcia, M., Gärdenfors, M., Garza, S., Gerdes, K., Ginter, F., Goenaga, I., Gojenola, K., Gökırmak, M., Goldberg, Y., Gómez Guinovart, X., Gonzáles Saavedra, B., Grioni, M., Grūzītis, N., Guillaume, B., Guillot-Barbance, C., Habash, N., Hajič, J., Hajič jr., J., Hà Mỹ, L., Han, N.-R., Harris, K., Haug, D., Hladká, B., Hlaváčová, J., Hociung, F., Hohle, P., Hwang, J., Ion, R., Irimia, E., Ishola, Ọ., Jelínek, T., Johannsen, A., Jørgensen, F., Kaşıkara, H., Kahane, S., Kanayama, H., Kanerva, J., Katz, B., Kayadelen, T., Kenney, J., Kettnerová, V., Kirchner, J., Kopacewicz, K., Kotsyba, N., Krek, S., Kwak, S., Laippala, V., Lambertino, L., Lam, L., Lando, T., Larasati, S. D., Lavrentiev, A., Lee, J., Lê Hồng, P., Lenci, A., Lertpradit, S., Leung, H., Li, C. Y., Li, J., Li, K., Lim, K., Ljubešić, N., Loginova, O., Lyashevskaya, O., Lynn, T., Macketanz, V., Makazhanov, A., Mandl, M., Manning, C., Manurung, R., Mărănduc, C., Mareček, D., Marheinecke, K., Martínez Alonso, H., Martins, A., Mašek, J., Matsumoto, Y., McDonald, R., Mendonça, G., Miekka, N., Misirpashayeva, M., Missilä, A., Mititelu, C., Miyao, Y., Montemagni, S., More, A., Moreno Romero, L., Mori, K. S., Mori, S., Mortensen, B., Moskalevskyi, B., Muischnek, K., Murawaki, Y., Müürisep, K., Nainwani, P., Navarro Horñiacek, J. I., Nedoluzhko, A., Nešpore-Bērzkalne, G., Nguyễn Thị, L., Nguyễn Thị Minh, H., Nikolaev, V., Nitisaroj, R., Nurmi, H., Ojala, S., Olúòkun, A., Omura, M., Osenova, P., Östling, R., Øvrelid, L., Partanen, N., Pascual, E., Passarotti, M., Patejuk, A., Paulino-Passos, G., Peng, S., Perez, C.-A., Perrier, G., Petrov, S., Piitulainen, J., Pitler, E., Plank, B., Poibeau, T., Popel, M., Pretkalniņa, L., Prévost, S., Prokopidis, P., Przepiórkowski, A., Puolakainen, T., Pyysalo, S., Rääbis, A., Rademaker, A., Ramasamy, L., Rama, T., Ramisch, C., Ravishankar, V., Real, L., Reddy, S., Rehm, G., Rießler, M., Rinaldi, L., Rituma, L., Rocha, L., Romanenko, M., Rosa, R., Rovati, D., Roșca, V., Rudina, O., Rueter, J., Sadde, S., Sagot, B., Saleh, S., Samardžić, T., Samson, S., Sanguinetti, M., Saulīte, B., Sawanakunanon, Y., Schneider, N., Schuster, S., Seddah, D., Seeker, W., Seraji, M., Shen, M., Shimada, A., Shohibussirri, M., Sichinava, D., Silveira, N., Simi, M., Simionescu, R., Simkó, K., Šimková, M., Simov, K., Smith, A., Soares-Bastos, I., Spadine, C., Stella, A., Straka, M., Strnadová, J., Suhr, A., Sulubacak, U., Szántó, Z., Taji, D., Takahashi, Y., Tanaka, T., Tellier, I., Trosterud, T., Trukhina, A., Tsarfaty, R., Tyers, F., Uematsu, S., Urešová, Z., Uria, L., Uszkoreit, H., Vajjala, S., van Niekerk, D., van Noord, G., Varga, V., Villemonte de la Clergerie, E., Vincze, V., Wallin, L., Wang, J. X., Washington, J. N., Williams, S., Wirén, M., Woldemariam, T., Wong, T.-s., Yan, C., Yavrumyan, M. M., Yu, Z., Žabokrtský, Z., Zeldes, A., Zeman, D., Zhang, M., and Zhu, H. (2018). Universal dependencies 2.3. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Nivre, J. and McDonald, R. (2008). Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958, Columbus, Ohio. Association for Computational Linguistics.

Nivre, J. and Scholz, M. (2004). Deterministic dependency parsing of English text. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 64–70, Geneva, Switzerland. COLING.

NLLB, Costa-jussà, M. R., Cross, J., Çelebi, O., Elbayad, M., Heafield, K., Heffernan, K., Kalbassi, E., Lam, J., Licht, D., Maillard, J., Sun, A., Wang, S., Wenzek, G., Youngblood, A., Akula, B., Barrault, L., Gonzalez, G. M., Hansanti, P., Hoffman, J., Jarrett, S., Sadagopan, K. R., Rowe, D., Spruit, S., Tran, C., Andrews, P., Ayan, N. F., Bhosale, S., Edunov, S., Fan, A., Gao, C., Goswami, V., Guzmán, F., Koehn, P., Mourachko, A., Ropers, C., Saleem, S., Schwenk, H., and Wang, J. (2022). No language left behind: Scaling human-centered machine translation.

Novak, R., Auli, M., and Grangier, D. (2016). Iterative refinement for machine translation.

Ogueji, K., Ahia, O., Onilude, G., Gehrmann, S., Hooker, S., and Kreutzer, J. (2022). Intriguing properties of compression on multilingual models.

OpenAI (2023). Gpt-4 technical report.

Ouchi, H., Shindo, H., and Matsumoto, Y. (2018). A span selection model for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1630–1642, Brussels, Belgium. Association for Computational Linguistics.

Pang, D., Lin, L. H., and Smith, N. A. (2019). Improving natural language inference with a pretrained parser.

Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D., Texier, M., and Dean, J. (2021). Carbon emissions and large neural network training.

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

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

Platanios, E. A., Sachan, M., Neubig, G., and Mitchell, T. (2018). Contextual parameter generation for universal neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 425–435, Brussels, Belgium. Association for Computational Linguistics.

Popović, M. (2015). chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Post, M. (2018). A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Pradhan, S., Moschitti, A., Xue, N., Uryupina, O., and Zhang, Y. (2012). CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference* *on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea. Association for Computational Linguistics.

Prato, G., Charlaix, E., and Rezagholizadeh, M. (2020). Fully quantized transformer for machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1–14, Online. Association for Computational Linguistics.

Punyakanok, V., Roth, D., and Yih, W.-t. (2008). The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Qu, Z., Liu, L., Tu, F., Chen, Z., Ding, Y., and Xie, Y. (2022). Dota: Detect and omit weak attentions for scalable transformer acceleration. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '22, page 14–26, New York, NY, USA. Association for Computing Machinery.

Quinn, J. and Ballesteros, M. (2018). Pieces of eight: 8-bit neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 114–120, New Orleans - Louisiana. Association for Computational Linguistics.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer.

Raunak, V., Menezes, A., and Junczys-Dowmunt, M. (2021). The curious case of hallucinations in neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1172–1183, Online. Association for Computational Linguistics.

Renduchintala, A., Diaz, D., Heafield, K., Li, X., and Diab, M. (2021). Gender bias amplification during speed-quality optimization in neural machine translation. In *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)*, pages 99–109, Online. Association for Computational Linguistics.

Roth, M. and Lapata, M. (2016). Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1192–1202, Berlin, Germany. Association for Computational Linguistics.

Sajjad, H., Dalvi, F., Durrani, N., and Nakov, P. (2023). On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Sanh, V., Wolf, T., and Rush, A. M. (2020). Movement pruning: Adaptive sparsity by fine-tuning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Satta, G. and Brill, E. (1996). Efficient transformation-based parsing. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 255–262, Santa Cruz, California, USA. Association for Computational Linguistics.

Savoldi, B., Gaido, M., Bentivogli, L., Negri, M., and Turchi, M. (2022). Under the morphosyntactic lens: A multifaceted evaluation of gender bias in speech translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1807–1824, Dublin, Ireland. Association for Computational Linguistics.

Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., Tow, J., Rush, A. M., Biderman, S., Webson, A., Ammanamanchi, P. S., Wang, T., Sagot, B., Muennighoff, N., del Moral, A. V., Ruwase, O., Bawden, R., Bekman, S., McMillan-Major, A., Beltagy, I., Nguyen, H., Saulnier, L., Tan, S., Suarez, P. O., Sanh, V., Laurençon, H., Jernite, Y., Launay, J., Mitchell, M., Raffel, C., Gokaslan, A., Simhi, A., Soroa, A., Aji, A. F., Alfassy, A., Rogers, A., Nitzav, A. K., Xu, C., Mou, C., Emezue, C., Klamm, C., Leong, C., van Strien, D., Adelani, D. I., Radev, D., Ponferrada, E. G., Levkovizh, E., Kim, E., Natan, E. B., Toni, F. D., Dupont, G., Kruszewski, G., Pistilli, G., Elsahar, H., Benyamina, H., Tran, H., Yu, I., Abdulmumin, I., Johnson, I., Gonzalez-Dios, I., de la Rosa, J., Chim, J., Dodge, J., Zhu, J., Chang, J., Frohberg, J., Tobing, J., Bhattacharjee, J., Almubarak, K., Chen, K., Lo, K., Werra, L. V., Weber, L., Phan, L., allal, L. B., Tanguy, L., Dey, M., Muñoz, M. R., Masoud, M., Grandury, M., Šaško, M., Huang, M., Coavoux, M., Singh, M., Jiang, M. T.-J., Vu, M. C., Jauhar, M. A., Ghaleb, M., Subramani, N., Kassner, N., Khamis, N., Nguyen, O., Espejel, O., de Gibert, O., Villegas, P., Henderson, P., Colombo, P., Amuok, P., Lhoest, Q., Harliman, R., Bommasani, R., López, R. L., Ribeiro, R., Osei, S., Pyysalo, S., Nagel, S., Bose, S., Muhammad, S. H., Sharma, S., Longpre, S., Nikpoor, S., Silberberg, S., Pai, S., Zink, S., Torrent, T. T., Schick, T., Thrush, T., Danchev, V., Nikoulina, V., Laippala, V., Lepercq, V., Prabhu, V., Alyafeai, Z., Talat, Z., Raja, A., Heinzerling, B., Si, C., Taşar, D. E., Salesky, E., Mielke, S. J., Lee, W. Y., Sharma, A., Santilli, A., Chaffin, A., Stiegler, A., Datta, D., Szczechla, E., Chhablani, G., Wang, H., Pandey, H., Strobelt, H., Fries, J. A., Rozen, J., Gao, L., Sutawika, L., Bari, M. S., Al-shaibani, M. S., Manica, M., Nayak, N., Teehan, R., Albanie, S., Shen, S., Ben-David, S., Bach, S. H., Kim, T., Bers, T., Fevry, T., Neeraj, T., Thakker, U., Raunak, V., Tang, X., Yong, Z.-X., Sun, Z., Brody, S., Uri, Y., Tojarieh, H., Roberts, A., Chung, H. W., Tae, J., Phang, J., Press, O., Li, C., Narayanan, D., Bourfoune, H., Casper, J., Rasley, J., Ryabinin, M., Mishra, M., Zhang, M., Shoeybi, M., Peyrounette, M., Patry, N., Tazi, N., Sanseviero, O., von Platen, P., Cornette, P., Lavallée, P. F., Lacroix, R., Rajbhandari, S., Gandhi, S., Smith, S., Requena, S., Patil, S., Dettmers, T., Baruwa, A., Singh, A., Cheveleva, A., Ligozat, A.-L., Subramonian, A., Névéol, A., Lovering, C., Garrette, D., Tunuguntla, D., Reiter, E., Taktasheva, E., Voloshina, E., Bogdanov, E., Winata, G. I., Schoelkopf, H., Kalo, J.-C., Novikova, J., Forde, J. Z., Clive, J., Kasai, J., Kawamura, K., Hazan, L., Carpuat, M., Clinciu, M., Kim, N., Cheng, N., Serikov, O., Antverg, O., van der Wal, O., Zhang, R., Zhang, R., Gehrmann, S., Mirkin, S., Pais, S., Shavrina, T., Scialom, T., Yun, T., Limisiewicz, T., Rieser, V., Protasov, V., Mikhailov, V., Pruksachatkun, Y., Belinkov, Y., Bamberger, Z., Kasner, Z., Rueda, A., Pestana, A., Feizpour, A., Khan, A., Faranak, A., Santos, A., Hevia, A., Unldreaj, A., Aghagol, A., Abdollahi, A., Tammour, A., HajiHosseini, A., Behroozi, B., Ajibade, B., Saxena, B., Ferrandis, C. M., Contractor, D., Lansky, D., David, D., Kiela, D., Nguyen, D. A., Tan, E., Baylor, E., Ozoani, E., Mirza, F., Ononiwu, F., Rezanejad, H., Jones, H., Bhattacharya, I., Solaiman, I., Sedenko, I., Nejadgholi, I., Passmore, J., Seltzer, J., Sanz, J. B., Dutra, L., Samagaio, M., Elbadri, M., Mieskes, M., Gerchick, M., Akinlolu, M., McKenna, M., Qiu, M., Ghauri, M., Burynok, M., Abrar, N., Rajani, N., Elkott, N., Fahmy, N., Samuel, O., An, R., Kromann, R., Hao, R., Alizadeh, S., Shubber, S., Wang, S., Roy, S., Viguier, S., Le, T., Oyebade, T., Le, T., Yang, Y., Nguyen, Z., Kashyap, A. R., Palasciano, A., Callahan, A., Shukla, A., Miranda-Escalada, A., Singh, A., Beilharz, B., Wang, B., Brito, C., Zhou, C., Jain, C., Xu, C., Fourrier, C., Periñán, D. L., Molano, D., Yu, D., Manjavacas, E., Barth, F., Fuhrimann, F., Altay, G., Bayrak, G., Burns, G., Vrabec, H. U., Bello, I., Dash, I., Kang, J., Giorgi, J., Golde, J., Posada, J. D., Sivaraman, K. R., Bulchandani, L., Liu, L., Shinzato, L., de Bykhovetz, M. H., Takeuchi, M., Pàmies, M., Castillo, M. A., Nezhurina, M., Sänger, M., Samwald, M., Cullan, M., Weinberg, M., Wolf, M. D., Mihaljcic, M., Liu, M., Freidank, M., Kang, M., Seelam, N., Dahlberg, N., Broad, N. M., Muellner, N., Fung, P., Haller, P., Chandrasekhar, R., Eisenberg, R., Martin, R., Canalli, R., Su, R., Su, R., Cahyawijaya, S., Garda, S., Deshmukh, S. S., Mishra, S., Kiblawi, S., Ott, S., Sang-aroonsiri, S., Kumar, S., Schweter, S., Bharati, S., Laud, T., Gigant, T., Kainuma, T., Kusa, W., Labrak, Y., Bajaj, Y. S., Venkatraman, Y., Xu, Y., Xu, Y., Xu, Y., Tan, Z., Xie, Z., Ye, Z., Bras, M., Belkada, Y., and Wolf, T. (2023). Bloom: A 176b-parameter open-access multilingual language model.

Schwenk, H., Wenzek, G., Edunov, S., Grave, E., Joulin, A., and Fan, A. (2021). CCMatrix: Mining billions of high-quality parallel sentences on the web. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6490–6500, Online. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016). Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Shen, D. and Lapata, M. (2007). Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 12–21, Prague, Czech Republic. Association for Computational Linguistics.

Shen, S., Dong, Z., Ye, J., Ma, L., Yao, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. (2020). Q-bert: Hessian based ultra low precision quantization of bert. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8815–8821.

Shi, P. and Lin, J. (2019). Simple bert models for relation extraction and semantic role labeling.

Smith, A., de Lhoneux, M., Stymne, S., and Nivre, J. (2018). An investigation of the interactions between pre-trained word embeddings, character models and POS tags in dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2711–2720, Brussels, Belgium. Association for Computational Linguistics.

Socher, R., Huang, E. H., Pennin, J., Manning, C. D., and Ng, A. Y. (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in neural information processing systems*, pages 801–809.

Socher, R., Karpathy, A., Le, Q. V., Manning, C. D., and Ng, A. Y. (2014). Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Stanovsky, G., Smith, N. A., and Zettlemoyer, L. (2019). Evaluating gender bias in machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1679–1684, Florence, Italy. Association for Computational Linguistics.

Stanton, S. D., Izmailov, P., Kirichenko, P., Alemi, A. A., and Wilson, A. G. (2021). Does knowledge distillation really work? In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*.

Straka, M. (2018). UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium. Association for Computational Linguistics.

Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Strubell, E., Verga, P., Andor, D., Weiss, D., and McCallum, A. (2018). Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y., and Zhou, D. (2020). MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, Online. Association for Computational Linguistics.

Surdeanu, M., Johansson, R., Meyers, A., Màrquez, L., and Nivre, J. (2008). The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England. Coling 2008 Organizing Committee.

Swayamdipta, S., Thomson, S., Lee, K., Zettlemoyer, L., Dyer, C., and Smith, N. A. (2018). Syntactic scaffolds for semantic structures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3772–3782, Brussels, Belgium. Association for Computational Linguistics.

Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks.

Tan, Z., Wang, M., Xie, J., Chen, Y., and Shi, X. (2017). Deep semantic role labeling with self-attention.

Tang, Y., Tran, C., Li, X., Chen, P.-J., Goyal, N., Chaudhary, V., Gu, J., and Fan, A. (2020). Multilingual translation with extensible multilingual pretraining and finetuning.

Tang, Y., Tran, C., Li, X., Chen, P.-J., Goyal, N., Chaudhary, V., Gu, J., and Fan, A. (2021). Multilingual translation from denoising pre-training. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3450–3466, Online. Association for Computational Linguistics.

Tao, C., Hou, L., Zhang, W., Shang, L., Jiang, X., Liu, Q., Luo, P., and Wong, N. (2022). Compression of generative pre-trained language models via quantization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4821–4836, Dublin, Ireland. Association for Computational Linguistics.

Tchernowitz, I., Yedidsion, L., and Reichart, R. (2016). Effective greedy inference for graph-based non-projective dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 711–720, Austin, Texas. Association for Computational Linguistics.

Tiedemann, J. (2012). Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).

Tiedemann, J. (2020). The tatoeba translation challenge – realistic data sets for low resource and multilingual MT. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1174–1182, Online. Association for Computational Linguistics.

Titov, I. and Henderson, J. (2007a). Constituent parsing with incremental sigmoid belief networks. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 632–639, Prague, Czech Republic. Association for Computational Linguistics.

Titov, I. and Henderson, J. (2007b). Fast and robust multilingual dependency parsing with a generative latent variable model. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 947–951, Prague, Czech Republic. Association for Computational Linguistics.

Titov, I. and Henderson, J. (2007c). A latent variable model for generative dependency parsing. In *Proceedings of the Tenth International Conference on Parsing Technologies*, pages 144–155, Prague, Czech Republic. Association for Computational Linguistics.

Torres Martins, A. F., Das, D., Smith, N. A., and Xing, E. P. (2008). Stacking dependency parsers. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 157–166, Honolulu, Hawaii. Association for Computational Linguistics.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–259.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023). Llama: Open and efficient foundation language models.

Treviso, M., Lee, J.-U., Ji, T., van Aken, B., Cao, Q., Ciosici, M. R., Hassid, M., Heafield, K., Hooker, S., Raffel, C., Martins, P. H., Martins, A. F. T., Forde, J. Z., Milder, P., Simpson, E., Slonim, N., Dodge, J., Strubell, E., Balasubramanian, N., Derczynski, L., Gurevych, I., and Schwartz, R. (2023). Efficient methods for natural language processing: A survey.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Veenstra, J. and Daelemans, W. (2000). A memory-based alternative for connectionist shift-reduce parsing. *CiteSeer*.

Vig, J. and Belinkov, Y. (2019). Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. Association for Computational Linguistics.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. (2019). Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Wang, F., Yan, J., Meng, F., and Zhou, J. (2021). Selective knowledge distillation for neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6456–6466, Online. Association for Computational Linguistics.

Wang, H., Ma, S., Dong, L., Huang, S., Zhang, D., and Wei, F. (2022). Deepnet: Scaling transformers to 1,000 layers.

Wang, R., Zhao, H., Ploux, S., Lu, B.-L., and Utiyama, M. (2016). A bilingual graph-based semantic model for statistical machine translation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 2950–2956. AAAI Press.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. (2020a). Linformer: Self-attention with linear complexity.

Wang, W. and Chang, B. (2016). Graph-based dependency parsing with bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2306–2315, Berlin, Germany. Association for Computational Linguistics.

Wang, Z., Wohlwend, J., and Lei, T. (2020b). Structured pruning of large language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6151–6162, Online. Association for Computational Linguistics.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. (2022a). Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Wei, X., Gong, R., Li, Y., Liu, X., and Yu, F. (2022b). Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization.

Weiss, D., Alberti, C., Collins, M., and Petrov, S. (2015). Structured training for neural network transition-based parsing. In *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)*, pages 323–333, Beijing, China. Association for Computational Linguistics.

Wilks, Y. (1975). A preferential, pattern-seeking, semantics for natural language inference. *Artificial Intelligence*, 6(1):53–74.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. (2019). Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Woods, W. A. (1970). Transition network grammars for natural language analysis. *Commun. ACM*, 13(10):591–606.

Wu, H., Judd, P., Zhang, X., Isaev, M., and Micikevicius, P. (2020). Integer quantization for deep learning inference: Principles and empirical evaluation.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xia, Q., Li, Z., Zhang, M., Zhang, M., Fu, G., Wang, R., and Si, L. (2019). Syntax-aware neural semantic role labeling. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7305–7313.

Xia, Q., Wang, R., Li, Z., Zhang, Y., and Zhang, M. (2020). Semantic role labeling with heterogeneous syntactic knowledge. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2979–2990, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Xu, C., Yao, J., Lin, Z., Ou, W., Cao, Y., Wang, Z., and Zha, H. (2018). Alternating multi-bit quantization for recurrent neural networks.

Xu, C., Zhou, W., Ge, T., Xu, K., McAuley, J., and Wei, F. (2021). Beyond preserved accuracy: Evaluating loyalty and robustness of BERT compression. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10653–10659, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xue, N., Chiou, F.-D., and Palmer, M. (2002). Building a large-scale annotated Chinese corpus. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Yamada, H. and Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 195–206, Nancy, France.

Yang, L., Zhang, M., Liu, Y., Yu, N., Sun, M., and Fu, G. (2017). Joint pos tagging and dependency parsing with transition-based neural networks.

Yang, Z., Cui, Y., and Chen, Z. (2022). Textpruner: A model pruning toolkit for pre-trained language models.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Yao, Z., Aminabadi, R. Y., Zhang, M., Wu, X., Li, C., and He, Y. (2022). Zeroquant: Efficient and affordable post-training quantization for large-scale transformers.

Yazdani, M. and Henderson, J. (2015). Incremental recurrent neural network dependency parser with search-based discriminative training. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 142–152, Beijing, China. Association for Computational Linguistics.

Yih, W.-t., Richardson, M., Meek, C., Chang, M.-W., and Suh, J. (2016). The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany. Association for Computational Linguistics.

Zadeh, A. H., Mahmoud, M., Abdelhadi, A., and Moshovos, A. (2022). Mokey: Enabling narrow fixed-point inference for out-of-the-box floating-point transformer models. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22, page 888–901, New York, NY, USA. Association for Computing Machinery.

Zeman, D., Hajič, J., Popel, M., Potthast, M., Straka, M., Ginter, F., Nivre, J., and Petrov, S. (2018). CoNLL 2018 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium. Association for Computational Linguistics.

Zhang, B., Williams, P., Titov, I., and Sennrich, R. (2020a). Improving massively multilingual neural machine translation and zero-shot translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1628–1639, Online. Association for Computational Linguistics.

Zhang, H. and McDonald, R. (2012). Generalized higher-order dependency parsing with cube pruning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 320–331, Jeju Island, Korea. Association for Computational Linguistics.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T., and Zettlemoyer, L. (2022). Opt: Open pre-trained transformer language models.

Zhang, T., Huang, H., Feng, C., and Cao, L. (2021a). Enlivening redundant heads in multi-head self-attention for machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3238–3248, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhang, W., Hou, L., Yin, Y., Shang, L., Chen, X., Jiang, X., and Liu, Q. (2020b). TernaryBERT: Distillation-aware ultra-low bit BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 509–521, Online. Association for Computational Linguistics.

Zhang, Y. and Nivre, J. (2011). Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics:*

*Human Language Technologies*, pages 188–193, Portland, Oregon, USA. Association for Computational Linguistics.

Zhang, Y., Qi, P., and Manning, C. D. (2018). Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215, Brussels, Belgium. Association for Computational Linguistics.

Zhang, Z., Liu, S., Li, M., Zhou, M., and Chen, E. (2017). Stack-based multi-layer attention for transition-based dependency parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1677–1682, Copenhagen, Denmark. Association for Computational Linguistics.

Zhang, Z., Strubell, E., and Hovy, E. H. (2021b). Comparing span extraction methods for semantic role labeling. In *SPNLP*.

Zhang, Z., Wu, Y., Zhao, H., Li, Z., Zhang, S., Zhou, X., and Zhou, X. (2020c). Semantics-aware bert for language understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9628–9635.

Zheng, X. (2017). Incremental graph-based neural dependency parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1655–1665, Copenhagen, Denmark. Association for Computational Linguistics.

Zhou, J., Li, Z., and Zhao, H. (2020a). Parsing all: Syntax and semantics, dependencies and spans. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4438–4449, Online. Association for Computational Linguistics.

Zhou, J. and Xu, W. (2015). End-to-end learning of semantic role labeling using recurrent neural networks. In *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)*, pages 1127–1137, Beijing, China. Association for Computational Linguistics.

Zhou, J., Zhang, Z., Zhao, H., and Zhang, S. (2020b). LIMIT-BERT : Linguistics informed multi-task BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4450–4461, Online. Association for Computational Linguistics.

Zhou, J. and Zhao, H. (2019). Head-Driven Phrase Structure Grammar parsing on Penn Treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy. Association for Computational Linguistics.

Zhou, S., Xia, Q., Li, Z., Zhang, Y., Hong, Y., and Zhang, M. (2022). Fast and accurate end-to-end span-based semantic role labeling as word-based graph parsing. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4160–4171, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

# Alireza **Mohammadshahi**

*Machine Learning Expert in Switzerland*

alireza.mohammadshahi@idiap.ch   www.idiap.ch/ amohammadshahi/   alirezamshi-zz   Alireza-mohammadshahi   +41 76 244 6997

## Key Competencies

**Experts in** machine learning, deep learning, NLP, data science, statistical analysis, computer vision
**Doctoral Researcher at** **EPFL**/IDIAP, working on deep learning for NLP and graph structures
**Part-time Postdoctoral at** **UZH**, working on multilingual pre-trained (generative) models
**Internships at** **META AI**, NAVERLABS, and EPFL on several ML and deep learning projects

## Education

**École Polytechnique Fédérale de Lausanne (EPFL)** *Lausanne, Switzerland*
DIRECT PH.D. IN COMMUNICATION AND COMPUTER SCIENCE (EDIC) *Sep 2018 - PRESENT*
- Supervisors: Prof. Karl Aberer, Dr. James Henderson

**Sharif University of Technology** *Tehran,Iran*
B.SC. IN ELECTRICAL ENGINEERING;COMMUNICATION,MINOR IN COMPUTER SCIENCE *Sep 2013 - Aug 2018*
- Supervisors: Prof. Babak Khalaj, Prof. Ali Fotowat Ahmadi

## Working Experience

**IDIAP Research Institute** *Martigny, Switzerland*
PH.D. RESEARCH ASSISTANT *February 2019 - PRESENT*
- Supervisor: Dr. James Henderson
- Topic: Deep Learning and Machine Learning on Representation Learning, Structure Prediction

**Computational Linguistics Group at UZH** *Zurich, Switzerland*
POST-DOCTORATE RESEARCH ASSISTANT (PART-TIME) *March 2023-PRESENT*
- Supervisor: Prof. Rico Sennrich
- Topic: Multilinguality in Generative AI Models (Large Language Models and Neural Machine Translation)

**META AI** *London, UK*
RESEARCH ENGINEER INTERN *July 2022 - November 2022*
- Supervisors: Marzieh Saeidi, **Thomas Scialom (FAIR)**, **Angela Fan (FAIR Leadership)**
- Topic: Generative AI Models (QG/QA), Information Retreival

**NAVERLABS Europe** *Grenoble, France*
RESEARCH ENGINEER INTERN *February 2022 - Jun 2022*
- Supervisors: Vassilina Nikoulina (Leader of NLP team), Alexandre Bérard, **Laurent Besacier (Head of AI)**
- Topic: Compression of pre-trained AI Models, **Co-creator of SMaLL-100 Translator**

**EPFL** *Lausanne, Switzerland*
RESEARCH SCIENTIST INTERN *September 2018 - February 2019*
- Supervisor: Prof. Karl Aberer and Remi Lebret
- Topic: Image-Caption Retrieval

## Programming Languages & Applications

**Programming Languages:** SQL, MATLAB, C/C++, Python
**Applications:** Github, Docker, Azure, AWS, bash/CLI
**ML-related:** Pytoch, Tensorflow, JAX, scikit-learn(regression, MLP, SVM), pandas, numpy, xgboost, Huggingface, Wandb, parallel computing (e.g. Deepspeed, Fairscale, Accelerator), Tableau, CUDA

## Honors & Awards

| | |
|---|---|
| **Offer** Post-doctorate SNSF scholarship by University of Zurich | *2023* |
| **Offer** EDIC PhD program fellowship by EPFL | *2018* |
| **Offer** EE PhD program fellowship by UCLA, USC, UofT | *2018* |
| **Rank 8th** out of +400,000 undergraduate applicants in universities entrance exam for B.Sc. degree | *2013* |
| **Member** National elite foundation of Iran | *2013 - PRESENT* |
| **Best Paper Award** Swiss Machine Learning Day | *2019* |
| **Offer** Google Travel Scholarship for 3rd Google NLP Summit | *2019* |
| **Finalist** of 10th ICC with proposing MyLinguist prototype (mylinguist.ch) | *2021* |

## Selected Projects

**Graph-to-Graph Transformers for Modeling Structured Data** *James Henderson*

MAIN PH.D. PROJECT AT IDIAP
- Proposing a general architecture to model structured data, then applied it to different ML tasks, and achieved the-state-of-the-art results in several benchmarks. It resulted in 5 publications in ACL-related venues.
- Collaborators: IDIAP, EPFL, Cardiff University, Geneva Graduate Institute
- Skills: Pytorch, Python, Tensorflow, AWS, Azure, deepspeed, sk-learn, docker

**Improving Meta AI Search Engine by using Generative Models** *M.Saeidi, T.Scialom, A.Fan*

INTERNSHIP AT META AI
- Proposing an evaluation metric for the question generation task (named RQUGE). We then successfully integrated it into the search engine of customer support at Meta. It led to one publication, submitted to ACL2023.
- Skills: Pytorch, Python, Tensorflow, AWS, deepspeed, sk-learn, docker, Amazon Turk, Parallel Computing

**Compression of Pre-trained Generative Models** *V.Nikoulina, L.Besacier*

INTERNSHIP AT NAVERLABS
- Compressing massively multilingual machine translation models. We first show the effect of compression on fairness and semantic biases, then we provide a compact multilingual NMT model. It led to 2 publications, presented at EMNLP2022 and multiple seminars.
- Creation of SMaLL-100 model: the smallest massively multilingual translator.
- Skills: Pytorch, Python, Tensorflow, AWS, Wandb, deepspeed, sk-learn

## Selected Publications (Google scholar link)

**RQUGE: Introducing Reference-Free Question Generation Metric by Better Encoding of Relevant Context** *2023*

SUBMITTED TO ACL 2023
- **A. Mohammadshahi**, T. Scialom, A. Fan, M. Yazdani, P. Yanki, M. Saeidi

**Syntax-Aware Graph-to-Graph Transformer for Semantic Role Labelling** *2023*

SUBMITTED TO ACL 2023
- **A. Mohammadshahi**, J. Henderson

**SMaLL-100: Introducing Shallow Multilingual Machine Translation Model for Low-Resource Languages** *2022*

THE CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING (EMNLP)
- **A. Mohammadshahi**, V. Nikoulina, A. Berard, C. Brun, J. Henderson and L. Besacier

**What Do Compressed Multilingual Machine Translation Models Forget?** *2022*

THE CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING (EMNLP)-FINDINGS
- **A. Mohammadshahi**, V. Nikoulina, A. Berard, C. Brun, J. Henderson and L. Besacier

**Recursive non-autoregressive graph-to-graph transformer for dependency parsing with iterative refinement** *2021*

TRANSACTIONS OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS
- **A. Mohammadshahi**, J. Henderson

**Aligning Multilingual Word Embeddings for Cross-Modal Retrieval Task** *2019*

FACT EXTRACTION AND VERIFICATION WORKSHOP AT EMNLP
- **A. Mohammadshahi**, R. Lebret, K. Aberer

## Invited Talks & Conferences

| | | |
|---|---|---|
| **Program Committee** | ACL, EMNLP, EACL, ARR, NAACL, TextGraph, FEVER | *2019 - PRESENT* |
| **Invited Talks** | University of Zurich, EPFL, IST & Unbabel Seminars, University of Tehran | *2022 - PRESENT* |

## Language

**English (Fluent), French (Basic), German (Basic), Persian (Native)**

## References

**Dr. James Henderson (advisor)**

HEAD OF NATURAL LANGUAGE UNDERSTANDING GROUP AT IDIAP
- Email: james.henderson@idiap.ch