Ph.D. DISSERTATION

Learning and Analysis of Neural Sentence Representations Using Syntax

구문론을 활용한 신경망 기반 문장 표현의 학습 및 분석

BY

TAEUK KIM

AUGUST 2021

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Learning and Analysis of Neural Sentence Representations Using Syntax

구문론을 활용한 신경망 기반 문장 표현의 학습 및 분석

지도교수 이 상 구

이 논문을 공학박사 학위논문으로 제출함

2021년 8월

서울대학교 대학원

컴퓨터공학부

김 태 욱

김태욱의 공학박사 학위논문을 인준함

2021년 8월

위 원 장	김형주
부위원장	이상구
위 원	황승원
위 원	김건희
위 원	서민준

Abstract

Syntax is a theory in linguistics that deals with the principles underlying the composition of sentences. As this theoretical framework provides formal instructions regarding the procedure of constructing a sentence with its constituents, it has been considered as a valuable reference in sentence representation learning, whose objective is to discover an approach of transforming a sentence into the vector that illustrates its meaning in a computationally tractable manner.

This dissertation provides two particular perspectives on harmonizing syntax with neural sentence representation models, especially focusing on constituency grammar. We first propose two methods for enriching the quality of sentence embeddings by exploiting the syntactic knowledge either represented as explicit parse trees or implicitly stored in neural models. Second, we regard syntactic formalism as a lens through which we reveal the inner workings of pre-trained language models which are state-of-the-art in sentence representation learning. With a series of demonstrations in practical scenarios, we show that syntax is useful even in the neural era where the models trained with huge corpora in an end-to-end manner are prevalent, functioning as either (i) a source of inductive biases that facilitate fast and effective learning of such models or (ii) an analytic tool that increases the interpretability of the black-box models.

Keywords: natural language processing, machine learning, deep learning, neural network, sentence representation, phrase-structure grammar, constituency parse extraction, unsupervised parsing, sentence classification, semantic textual similarity, recursive neural network, Transformer, pre-trained language models

Student Number: 2016-21197

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Language is structured by nature. A word, which is the smallest unit of a language that can stand alone (Bloomfield, 1926), consists of a set of characters or a group of morphemes. Likewise, words are organized into phrases, and constituents in a sentence are embedded inside of other constituents (Manning and Schutze, 1999; Carnie, 2012). It is thus essential for computational models of natural language to have a fundamental understanding of such hierarchy, either implicitly or explicitly.

Among the concepts existing in the hierarchical structure of language, a sentence is one of the most integral components, playing a key role as a basic unit of conveying a complete thought or meaning (Gardiner, 1922). As the larger parts of spoken or written language, such as a paragraph, a conversation, and a piece of writing, can be simply interpreted as a sequence of sentences in a sense, it is a crucial and elemental step in natural language processing (NLP) to figure out the semantics of sentences.

Sentences first need to be transformed into a computationally tractable

form, i.e., sentence representation, to be properly processed by machines. Sentence representation (or sentence embedding) is a real-valued vector that is designed to encode various aspects of the target sentence.

In the initial stages of NLP and information retrieval, a sentence (or potentially a bunch thereof) was usually represented as a sparse vector computed by the bag-of-words method (Schütze et al., 2008), where each of the vector's dimensions indicates the existence of the corresponding word or the number of occurrences of the word in the input. Despite its efficiency, the bag-of-words representation has a clear limitation in its performance due to its ignorance in word order and grammatical regularities.

As the neural era that is originated from the rediscovery of the approaches based on neural networks has begun in NLP, researchers start to pay their attention to developing a methodology that proposes how to learn *dense* representations of different language units rather than sparse ones. One of the representative cases is that of words, where several seminal studies (Mikolov et al., 2013a,b; Pennington et al., 2014) grounded on distributional hypothesis (Harris, 1954) have demonstrated their effectiveness in learning meaningful dense word embeddings. Witnessing the success of the word-level methods, the following research question naturally arises: Can we also effectively derive dense representations for sentences from such word vectors?

Sentence representation learning, the task of finding an optimal space in which each sentence is properly mapped to a point (i.e., a dense vector) that best illustrates the original meaning of the sentence, has been at the core of modern NLP, as (i) a substantial portion of research in NLP still lies at solving sentence-related problems, e.g., sentiment analysis (Socher et al., 2013b), natural language inference (Bowman et al., 2015; Williams et al., 2018b), and semantic textual similarity (Cer et al., 2017), and (ii) sentence representations

are also essential in dealing with higher-level tasks such as question answering (Rajpurkar et al., 2016) and machine translation (Bojar et al., 2016, 2018).

A variety of neural architectures have been proposed for effective sentence representation learning. For instance, a group of researchers have proposed to adopt window-based models such as convolutional neural network (Kim, 2014) to capture the local context of sentences. Meanwhile, recurrent neural models (Elman, 1990; Hochreiter and Schmidhuber, 1997; Cho et al., 2014)), which consume a token in a sentence one by one, have also shown their competitive performance in encoding an input sentence into a fixed-dimensional vector.

Although the above approaches have been extensively utilized for a while with decent performance, they often struggle with the shortcomings coming from their architecture design. First, they have a difficulty in recognizing a long-distance correlation between the words distant from each other, which is frequent in natural language. Second, they are not equipped with any inductive bias towards the hierarchical nature of language, which may result in their unsatisfactory performance or data inefficiency in training.

In this dissertation, we concentrate on exploring the potential of syntactic knowledge as an invaluable resource to relieve the limitations of the aforementioned neural models for sentence representations. Syntax in linguistics is the study of the principles and processes by which sentences are constructed in languages (Chomsky, 1957, 1965). As syntax provides a formal instruction as to the procedure of combining constituents into a full sentence, it has received much attention from researchers who attempt to transform this theoretical framework into a practical approach for sentence representation learning (Socher et al., 2013b; Tai et al., 2015), giving birth to recursive neural network whose computation graph dynamically mimics the syntactic structure of an input sentence. We examine how much effective such syntax-inspired neural models are in con-

structing sentence representations, and we also propose their improved variants that can better exploit the syntactic information provided by parse trees.

On the other hand, the emergence of self-attention models including Transformer (Vaswani et al., 2017) has recently reshaped the landscape of the sentence representation learning literature, demonstrating that their relative flexibility (compared to other old-fashioned neural architectures) in making connections between words is a key to generating high-quality sentence vectors. Some studies have also discussed the potential that Transformers may autonomously acquire an ability to detect some linguistic relationships between words, especially when they are pre-trained in an unsupervised manner (Goldberg, 2019; Hewitt and Manning, 2019). However, it is still under debate how much syntactic information such attention-based neural architectures can capture and how the captured knowledge can be more effectively utilized in diverse ways.

To shed some light on this active research topic, we introduce a method that estimates the extent to which neural models for sentence representations are aware of syntactic concepts, regarding syntax as a means of probing the inner workings of these models. Furthermore, we develop an approach that takes advantage of the lessons learnt from our investigation—e.g., syntactic knowledge is relatively distinct in the middle layers of pre-trained models—to further improve the quality of the sentence representations generated by pre-trained Transformers.

To summarize, in this thesis, we discuss two disparate usages of syntax for pursuing a better modeling of sentences using neural networks, mainly focusing on phrase-structure (a.k.a. constituency) grammar. First, we investigate the methods of directly improving existing strategies for sentence representation learning when syntactic knowledge is provided explicitly as a form of parse trees or implicitly from different parts of pre-trained models. Second, we introduce

an analytic tool for estimating the degree to which neural models for sentence representations recognize syntactic components residing in sentences, offering an insight on how much the working mechanism of these models accords with what is considered as the norm in the theory of syntax.

1.1 Dissertation Outline

This dissertation is largely divided into four parts, whose contents are specified as follows:

- **Background (Chapter 2)**: As a preliminary, we give a brief overview of the background knowledge that forms the foundation of this dissertation. Specifically, we explain some basic concepts in syntax and how they are utilized in this thesis. We also describe several neural network architectures for sentence representations. Finally, we provide a simple introduction to the topics in the NLP literature that have a close relationship with our work.

- **Part I (Chapters 3 & 4) — Sentence representation learning utilizing syntax**: In the first part of the main contents, we propose two practical methods of learning high-quality sentence representations by leveraging additional syntactic information. To be specific, in Chapter 3, we first suppose we have access to an explicit form of syntactic knowledge represented as constituency parse trees. We then suggest a variant of recursive neural networks, dubbed as **SATA Tree-LSTM**, that takes better advantage of syntactic classes and tags thereof (e.g., NP and VP) in addition to the phrase-structure of sentences. On the other hand, in Chapter 4, we aim to leverage the syntactic knowledge implicitly stored in pre-trained neural models in the process of constructing sentence em-

beddings. We develop a **self-guided contrastive learning** method for BERT (Devlin et al., 2019) sentence representations, where we exploit not only the last layer of BERT as the convention but also its lower and middle layers which are known as better at capturing syntactic information (Jawahar et al., 2019).

- **Part II (Chapters 5 & 6) — Syntactic analysis of neural models for sentence representations**: In the second part, we consider syntax as an analytic means of dissecting the inner workings of recent gigantic neural models for sentence representations. Specifically, we establish a novel approach called **C**onstituency **P**arse **E**xtraction from **P**re-trained **L**anguage **M**odels (**CPE-PLM**) and propose its two instantiations (i.e., **top-down** and **chart-based CPE-PLM**) in Chapter 5 and Chapter 6, respectively. The CPE-PLM framework enables us to predict the parse trees of input sentences by not training a separate parser but relying only on the internal patterns formed by the parameters of pre-trained language models (PLMs). The induced trees are utilized as a valuable resource for estimating the lower bound of such PLMs on understanding syntactic concepts. We also report the several intriguing findings discovered in the generated trees.

- **Conclusion (Chapter 7)**: We finally conclude our discussion and propose prospective avenues for future work.

1.2 Related Publications

Portions of this dissertation appeared in the following publications or preprint:

- **Chapter 3**: Taeuk Kim, Jihun Choi, Daniel Edmiston, Sanghwan Bae,

and Sang-goo Lee. 2019b. Dynamic compositionality in recursive neural networks with structure-aware tag representations. In *Proceedings of AAAI*.

- **Chapter 4**: Taeuk Kim, Kang Min Yoo, and Sang-goo Lee. 2021b. Self-guided contrastive learning for BERT sentence representations. In *Proceedings of ACL-IJCNLP*.

- **Chapter 5**: Taeuk Kim, Jihun Choi, Daniel Edmiston, and Sang-goo Lee. 2020. Are pre-trained language models aware of phrases? simple but strong baselines for grammar induction. In *Proceedings of ICLR*.

- **Chapter 6**: Taeuk Kim, Bowen Li, and Sang-goo Lee. 2021a. Multilingual chart-based constituency parse extraction from pre-trained language models. *arXiv preprint arXiv:2004.13805*.

Chapter 2

Background

In this chapter, we provide an overview of several concepts that constitute the core of this dissertation. We first touch on several basic notions in syntax which are frequently mentioned throughout the thesis, and then describe a few neural network architectures widely utilized for sentence representation learning. Finally, we introduce some literature in natural language processing with which our work has a close connection.

2.1 Introduction to Syntax

Syntax, which is originated from the Greek word *sỳntaxis*—meaning "setting out together or arrangement" (Jurafsky, 2000), is a subfield in linguistics that deals with how sentences are structured to make a meaning from a sequence of words (Carnie, 2012). The currently dominant form of syntactic theories, called generative grammar, has been mainly established by Noam Chomsky (Chomsky, 1957, 1965, 1975). The assumption lying at the heart of the theory

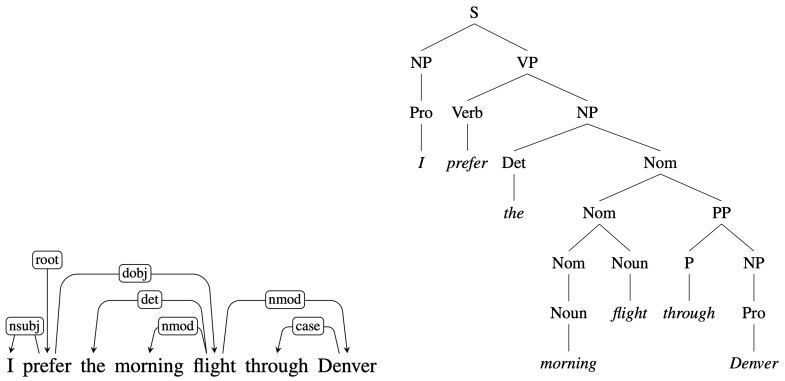

Figure 2.1 Comparison between the dependency (Left) and constituency parse tree (Right) for the sentence "I prefer the morning flight through Denver" (Jurafsky, 2000).

is that a language is defined by the set of possible sentences generated by a specific grammar.

In general, a grammar consists of a set of formal rules (operators) and their operands (symbols). Even though there exist different categories of syntactic formalism including dependency grammar (see Figure 2.1), we focus on constituency grammar in this thesis, which is also known as context-free or phrase-structure grammar. Formally, following Kim et al. (2019c), we define a constituency grammar as a five-tuple $G = (S, N, P, \Sigma, R)$ where S is the unique start symbol, N is a set of non-terminals (e.g., NP and VP), P is a set of pre-terminals, Σ is a set of terminal symbols (words), and R is a set of rules of the following form:

$$
\begin{aligned}
&S \to A, && A \in N \\
&A \to B\ C, && A \in N,\ B, C \in N \cup P \\
&T \to w, && T \in P,\ w \in \Sigma.
\end{aligned}
$$

That is, the generation of every sentence in the language governed by this grammar starts from the starting symbol S, and one of the three rules mentioned above is recursively applied until the sentence is completed with a sequence of terminal symbols $S = \{w_1, \ldots, w_n | \forall i, w_i \in \Sigma\}$. This entire generation (or rule expansion) procedure is called a *derivation* of the string of words, and it is generally represented by a *parse tree* as in Figure 2.1 (Jurafsky, 2000).

Unfortunately, we cannot immediately recognize the inherent structure of a given sentence at a first glance, mainly because its surface form is sequential in the real world. A common solution to remedy this problem is introducing a *parser*, which is a function that outputs the parse tree of a sentence obeying its target grammar. Parsers are typically trained with supervision from *treebanks* in which a collection of sentences and corresponding gold-standard parse trees annotated by domain experts (i.e., linguists) are stored. Note that each treebank may be established on disparate grammatical rule sets, and that following the majority in the literature, in this dissertation we rely on the grammatical system used in constructing the Penn Treebank dataset (PTB; Marcus et al. (1993)).

2.2 Neural Networks for Sentence Representations

To model a sentence in a vector space and convert it into a computationally tractable form, different types of neural architectures such as recurrent neural network (RNN; Elman (1990)) have been proposed. Among a variety of possible choices, we rely primarily on two specific encoders in our discussion—recursive neural network (RvNN[1]; Socher et al. (2011)) and Transformer (Vaswani et al., 2017). They are similar in that both are the functions that accept (sub-)word

[1]To avoid confusion, we name recursive neural network (a.k.a. tree-structured neural network) as *RvNN* to differentiate it from recurrent neural network (*RNN*), following the convention in the literature (Choi et al., 2018).

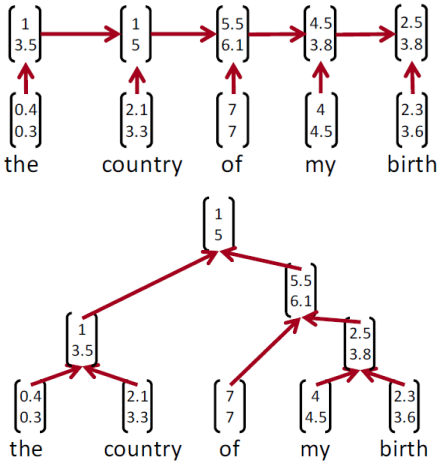

Figure 2.2 Comparison between RNN (Top) and RvNN (Bottom). RNN can be considered as a specific case of RvNN where the given tree structure is a chain.

embeddings as ingredients and compute the resulting sentence vectors, but different from each other in terms of the strategies they take to merge word and phrase level information. In the following subsections, we provide an introduction to the two architectures in more detail.

2.2.1 Recursive Neural Network

Recursive neural network (Pollack, 1990; Goller and Kuchler, 1996; Socher et al., 2011, 2012, 2013a; Tai et al., 2015) is a neural model whose computation graph is structured as a tree rather than just a sequence. It can be also interpreted as a generalization of recurrent networks; i.e., RNN is a special case of RvNN equipped with a chain-like tree structure as in Figure 2.2[2].

While RvNN in principle can accept any acyclic graph structure as input, in NLP, it is mostly combined with linguistically-inspired structures such as constituency and dependency parse trees which are annotated by domain experts

[2]The images are from `https://imgur.com/MmQRH38` and `http://imgur.com/bfVzTm5`.

or predicted by off-the-shelf parsers. The clear advantage we obtain by applying such parse trees to RvNN is that it naturally enables us to inject the inductive bias with regard to the composition order of words in a sentence—what is believed to be correct in linguistics—into our neural models. Moreover, RvNN is more robust to the vanishing and exploding gradient problem compared to the typical RNN model as its longest backpropagation path increases in the order of $\log O(n)$ (n: the length of an input sequence), which is asymptotically much smaller than that of RNN, i.e., $O(n)$ (Goodfellow et al., 2016).

Despite the intuitive linguistic motivation on which it was invented, RvNN sometimes suffers from difficulties in achieving its optimal performance. We conjecture this is partially because most RvNN instances do not properly exploit syntactic tags which are attached on each node of a constituency tree but stick only to structural information. To relieve this problem, we propose a revision of RvNN in Chapter 3 that shows better performance by making use of the proper combination of the information from both structure and phrasal categories.

2.2.2 Transformer

Transformer (Vaswani et al., 2017) has become the de-facto standard neural model for NLP and other areas where sequential data is abundant. Basically, it follows the sequence-to-sequence framework (Sutskever et al., 2014; Bahdanau et al., 2015), consisting of an encoder and decoder (Figure 2.3). However, it is distinct from conventional models in that both of its encoder and decoder rely solely on a stack of Transformer blocks that are made of attention-based modules and other supplementary components, dispensing with recurrence and convolutions. As a result, Transformer succeeds in becoming more suitable for parallel computation on modern hardware such as GPU while achieving competitive or even superior performance.

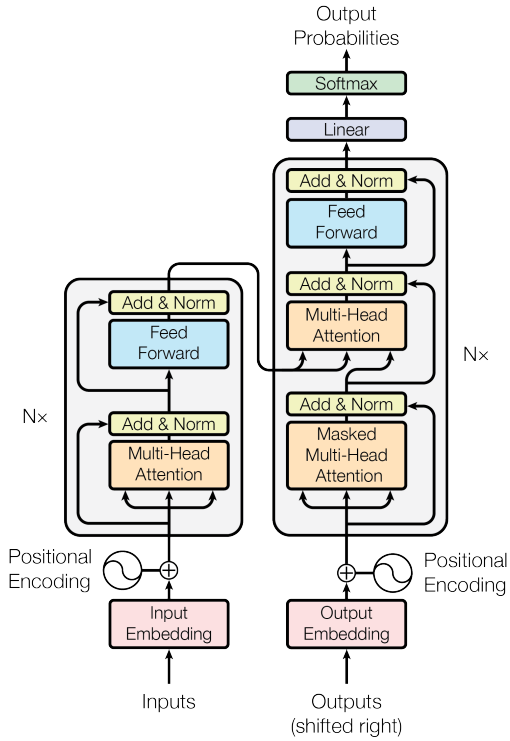

Figure 2.3 The Transformer architecture (Vaswani et al., 2017).

Out of various elements composing the Transformer block (the grey rectangles in Figure 2.3), the multi-head attention mechanism (orange) undoubtedly lies at the core of characterizing Transformer. To be specific, the multi-head attention mechanism is an extension of the scaled dot-product attention which is defined as follows:

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

where Q, K, and V are query, key, and value matrices respectively, and d_k is the dimension of the row vector of Q. In other words, given Q, K, and V, which are the linear transformations of the output of the previous Transformer block, the scaled dot-product attention computes a weighted sum of the value vectors

whose weights are derived from the dot products between respective query and key vectors. In the multi-head attention mechanism, the above computation is allowed to be conducted several times, but with different heads which are composed of separate parameters (i.e., W_i^Q, W_i^K, and W_i^V):

$$\begin{aligned}\text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_O \\ \text{where head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V).\end{aligned}$$

To summarize, the multi-head attention mechanism provides a place where input representations are reformulated and recombined in succession, following the multiple perspectives supported by different attention heads, and the final representations made by the mechanism are the outcome of a series of revision performed through several Transformer blocks such that each token in a sentence is best represented in the context of the input sequence.

It is worth noting that the softmax part (i.e., $\text{softmax}(\frac{QK^T}{\sqrt{d_k}})$) in the scaled dot-product attention offers a lens for interpretability, through which we can inspect the inner workings of Transformer. For instance, we can estimate the extent to which a token is regarded as important from the view of Transformer by investigating the degree to which this token receives attention from other tokens. In Chapter 5 and 6, we take advantage of this fact as a clue to reveal the degree to which pre-trained Transformer models grasp syntactic concepts.

2.2.3 Pre-trained Language Models

A language model (Brown et al., 1992; Kneser and Ney, 1993; Bengio et al., 2003), which is a model originally designed to compute a probability distribution over sequences of words in a language, is integral to natural language processing. Its impact on the field has been growing much faster and greater than ever before since it was known that the language model can be also understood

as a self-learner that masters how to capture useful features from text by itself (Howard and Ruder, 2018; Peters et al., 2018). In particular, its role as a general feature extractor for downstream tasks begins to be more prominent with the introduction of Transformer (Devlin et al. (2019); Radford et al. (2019); Yang et al. (2019); Liu et al. (2019b); *inter alia*).

A current, typical form of pre-trained language models, mostly triggered by the success of BERT (Devlin et al., 2019), is characterized by three factors. First, it adopts a Transformer encoder, decoder or both as its backbone architecture. For instance, BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019b) employ Transformer encoders while GPT3 (Brown et al., 2020) is based on a Transformer decoder. Second, the model is pre-trained on gigantic corpora such as web-crawled data (e.g., a Wikipedia dump) and the book corpus (Zhu et al., 2015). Third, its learning objective belongs to the family of language modeling—that is the reason why it is called a language model in the first place—including autoregressive language modeling or masked language modeling. In this thesis, we narrow the definition of the term *pre-trained language models (PLMs)* to refer to the models that adhere to the three aforementioned properties.

Although pre-trained language models are excellent feature providers for many NLP tasks, their usage for a specific purpose requires an adaptation to the task of interest, typically achieved by fine-tuning them with supervision from labeled data in addition to introducing extra parameters consisting of a task-specific layer. Moreover, as their outputs are basically contextualized token-level representations, it is still unclear how best to derive sentence representations from such models. In Chapter 4, we propose a novel method for constructing sentence representations by better exploiting PLMs, inspired by the fact that the different layers of the models are respectively specialized in catching disparate linguistic information.

2.3 Related Literature

We wrap up this chapter with the introduction of some seminal work in the fields of our concern, including sentence representation learning, probing approaches for neural NLP models, and grammar induction and unsupervised parsing. Further studies that have a closer relationship with our work will be additionally mentioned in the associated chapters.

2.3.1 Sentence Representation Learning

As a substantial portion of NLP tasks belong to sentence-level ones, e.g., sentiment analysis (Socher et al., 2013b), natural language inference (NLI; Bowman et al. (2015); Williams et al. (2018b)), and semantic textual similarity (Cer et al., 2017), sentence representation learning has been one of the fundamental research subjects in natural language processing. Basically, every neural model that is devised to solve such tasks should contain its own internal module that takes charge of representing input sentences in a vector space. This module typically concentrates on how to compose token (e.g., characters, subwords, and words) representations into *task-specific* sentence-level features.

On the other hand, much effort also has been devoted to develop methods for producing more *generally applicable* sentence representations, rather than training individual modules for respective tasks (Kiros et al., 2015; Hill et al., 2016; Conneau et al., 2017; Cer et al., 2018; Logeswaran and Lee, 2018; Reimers and Gurevych, 2019). For instance, Skip-Thought (Kiros et al., 2015) learns to compress the information of a sentence into a dense vector by training a sequence-to-sequence framework whose goal is to generate the previous and next sentences of the input sentence. InferSent (Conneau et al., 2017) demonstrates that a Bi-LSTM (Hochreiter and Schmidhuber, 1997) sentence encoder trained to resolve the NLI task can be generalized to others. SBERT (Reimers and

Gurevych, 2019) is also grounded on the strategy same as that of InferSent, however, it is a fine-tuned version of BERT (Devlin et al., 2019) instead of being trained from scratch.

In later chapters, we propose two additional improvements on top of the aforementioned work. We focus on a variant of task-specific recursive neural networks in Chapter 3. In Chapter 4, we attempt to uncover the potential of BERT as both task-specific and general sentence representation providers.

2.3.2 Probing Methods for Neural NLP Models

Since neural network based models, which are black boxes by nature, begin to dominate the field of natural language processing, there have arisen a series of approaches to probe their inner workings (Ettinger et al., 2016; Linzen et al., 2016; Adi et al., 2017; Conneau et al., 2018; Kim et al., 2019a). This trend has been accelerated with the rise of pre-trained Transformer language models (e.g., BERT, see Section 2.2.3 for details), which match or outperform human performance in several natural language understanding tasks (Wang et al., 2019a,b), blooming the subfield called BERTology (Rogers et al., 2020).

The most common type of probing techniques is a linear probe (Alain and Bengio, 2016; Hewitt and Manning, 2019), where a simple linear layer is appended on top of the target model which remains fixed, and the linear layer is trained with supervision such that it becomes a detector that perceives whether the target model is sensitive to the properties of our concern. Despite their simplicity and effectiveness, linear probes are often criticized because of their two explicit shortcomings: (i) they rely on supervision from labeled data which is hard to collect, and (ii) they cannot definitely distinguish whether their performance is entirely due to their target model or their own ability obtained from the supervision. In Chapter 5 and Chapter 6, we introduce a novel syntactic

probe to relieve this problem, which does not require any type of training and thus overcomes the limitations of the linear probes.

2.3.3 Grammar Induction and Unsupervised Parsing

One of the most controversial hypotheses advocated by Noam Chomsky and his school in linguistics is that many parts of language are originated from an innate ability of human beings, and that therefore, they cannot be acquired (Chomsky, 1957, 1965, 1975; Carnie, 2012). Contrary to this claim, many researchers in NLP and linguists have been interested in the area called grammar induction, where one's goal is to train a model that is aware of syntactic concepts or context-free grammars without parse tree annotations (Lari and Young, 1990; Klein and Manning, 2002a). Note that if it is possible to train such model, it can be also utilized as an unsupervised parser that derives the parse tree of a sentence following its acquired grammar rules.

In recent years, the attention on grammar induction and unsupervised parsing has been reignited by some pioneering work that proposes applying neural approaches to the problem. PRPN (Shen et al., 2018b) and ON-LSTM (Shen et al., 2019) have demonstrated that specific variations of neural language models, from which one can extract a parse tree-like structure, can be implemented through dedicated network design. Kim et al. (2019c) have verified that a neural parameterization of probabilistic context-free grammar models is viable. Loosely speaking, our method, dubbed as Constituency Parse Extraction from Pre-trained Language Models (CPE-PLM; in Chapter 5 and 6), is a sort of unsupervised parsing methods as it predicts parse trees without supervision from treebanks. We further discuss its relationship with other approaches for grammar induction and unsupervised parsing in the corresponding chapters.

Chapter 3

Sentence Representation Learning with Explicit Syntactic Structure

3.1 Introduction

One of the most fundamental topics in natural language processing is how best to derive high-level representations from constituent parts, and the development of a method for constructing a sentence representation from distributed word embeddings is an example domain of this larger issue. Although sequential neural models such as recurrent neural network (RNN; Elman (1990)) and its variants—Long Short-Term Memory (LSTM; Hochreiter and Schmidhuber (1997)) and Gated Recurrent Unit (GRU; Cho et al. (2014))—have become one of the de-facto standards for condensing sentence-level information into a fixed vector, there have been also other lines of approaches for sentence embeddings where convolutional neural network (CNN; Kim (2014)) or self-attention models (Vaswani et al., 2017; Shen et al., 2018a) are utilized.

From a linguistic point of view, the underlying tree structure—as expressed

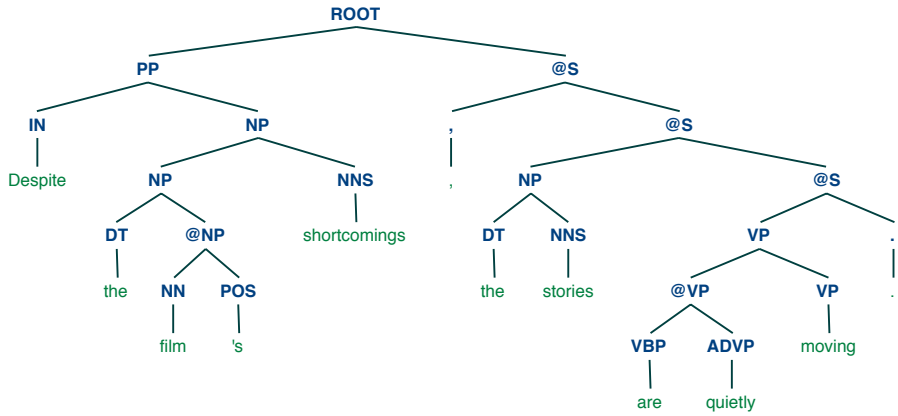

Figure 3.1 A (binarized) constituency pase tree example from Stanford Sentiment Treebank (Socher et al., 2013b).

by its constituency and dependency trees—of a sentence is an integral part of its meaning. Inspired by this fact, several recursive neural network (RvNN) models have been proposed in order to reflect such linguistic intuition in designing neural architectures, achieving impressive results on downstream tasks such as sentiment analysis (Socher et al., 2012, 2013b), machine translation (Yang et al., 2017), natural language inference (Bowman et al., 2016), and discourse relation classification (Wang et al., 2017). However, the issue is that the utilization of syntactic knowledge is only possible when we are already equipped with a decent parser or gold-standard trees, both of which are fairly complicated to prepare. To tackle this, a few studies have (Yogatama et al., 2017; Choi et al., 2018) suggested latent RvNNs that learn to construct task-specific tree structures by themselves, without explicit supervision from syntax. The rise of those latent tree models naturally leads to the following research question: Is it still optimal to adhere to parse trees when formulating RvNNs? Or can we entirely replace them by learning task-specific structures from scratch?

In this chapter, we advocate the effectiveness of the linguistic knowledge provided by parse trees in building sentence representations, with the demonstration that our novel RvNN variant (SATA Tree-LSTM) based on constituency trees and the tag[1] information thereof can attain competitive performance on several sentence-level tasks, e.g., sentiment analysis and natural language inference. The main novelty of our approach comes from the idea that we introduce a small, separate tag tree-LSTM in order to control the composition function of the original word tree-LSTM. Throughout extensive experiments, we show that the proper fusion of the signals obtained from both the structure and syntactic tags of parse trees facilitates a boost in the performance of linguistically-informed RvNNs.

3.2 Related Work

The models belonging to recursive neural network (RvNN) in NLP share the common principle that they compute higher-level representations based on an input tree structure (one obtained from some types of parse trees in most cases) in addition to a sequence of words. However, an RvNN instance is distinguished from one another depending on the composition function it leverages—e.g., feed-forward neural networks (Socher et al., 2011), matrix-vector multiplication (Socher et al., 2012), tensor computation (Socher et al., 2013b), and LSTM (Tai et al., 2015).

A drawback of such initial RvNNs is their inflexibility, i.e., their inability to handle *dynamic compositionality* for different syntactic configurations. For instance, they cannot properly differentiate adjective-noun composition from that of verb-noun or preposition-noun as they only attend to the structure of

[1]We do not distinguish part-of-speech (POS) tags (e.g., DT-determiner, JJ-adjective) from phrase-level tags (e.g., NP-noun phrase, VP-verb phrase), and refer to both simply as 'tags'.

parse trees, ignoring rich information from syntactic tags. This is obviously suboptimal, as models failing to make such a distinction ignore real-world syntactic considerations such as '-arity' of function words (i.e., types) and the adjunct/argument distinction.

To mitigate the aforementioned problem, a line of studies (Hashimoto et al., 2013; Dong et al., 2014; Qian et al., 2015; Wang et al., 2017; Liu et al., 2017b; Huang et al., 2017; Teng and Zhang, 2017) have proposed to directly leverage tag information which is produced as a by-product of parsing. In detail, Qian et al. (2015) suggested TG-RNN, which employs different composition functions according to POS tags, and TE-RNN/TE-RNTN, which leverage dense tag embeddings as supplementary inputs for existing tree-structured models. Despite the novelty of utilizing tag information, the explosion of the number of parameters (TG-RNN) or the limited performance of the original models (TE-RNN/TE-RNTN) have prevented these models from being widely adopted. Similarly, Wang et al. (2017) and Huang et al. (2017) proposed an improved version of tree-LSTM that also utilizes static tag vectors to control gate functions in the model. However, its realization is still too naïve to fully exploit both the structure and tags of a parse tree together. We introduce structure-aware tag representations in the next section to overcome this limitation.

Another direction of implementing dynamic compositionality in RvNNs is to take advantage of a meta-network (or hyper-network). Inspired by recent work on dynamic parameter prediction, DC-TreeLSTM (Liu et al., 2017b) dynamically creates parameters for compositional functions in a tree-LSTM. Specifically, DC-TreeLSTM has two separate tree-LSTM networks whose architectures are similar, but the smaller of the two is utilized to calculate the weights of the bigger one. A potential issue for this model is that the model's training is easy to be degenerated such that the role of the two tree-LSTMs becomes duplicated.

On the contrary, we design two disentangled tree-LSTMs in our model so that one focuses only on extracting useful features from syntactic inputs (tags) while the other composes semantic units (words) with the aid of the extracted features. Furthermore, our model reduces computation complexity by sticking to the original formulation of tree-LSTM instead of repetitively deriving weights for each example.

Finally, some recent work (Yogatama et al., 2017; Choi et al., 2018) has proposed latent tree-structured models that learn how to formulate tree structures only from a sequence of tokens, without supervision from parse trees. The latent RvNNs' merit is that they can make a task-specific word composition order rather than a sequential or syntactic one. However, the problem is that they are only able to predict a structure, ignoring additional context coming from the phrasal categories defined for constituents in the structure. In experiments, we compare our model not only with linguistically-informed RvNNs but also with the latent tree models, demonstrating that modeling RvNN with explicit linguistic knowledge can be still an attractive option.

3.3 Method

We introduce a novel RvNN architecture, called **SATA Tree-LSTM** (**S**tructure-**A**ware **T**ag **A**ugmented **Tree-LSTM**). This model is similar to typical tree-LSTMs, but provides dynamic compositionality by augmenting a separate *tag tree-LSTM* which produces a structure-aware tag representation for each node in a parse tree. In other words, our model has two independent tree-structured modules based on the same constituency tree, one of which (word tree-LSTM) is responsible for constructing sentence representations given a sequence of words as usual while the other of which (tag tree-LSTM) provides supplementary syntactic information to the former.

In Section 3.3.1, we first review the original tree-LSTM. Then, we introduce tag tree-LSTM and structure-aware tag representations in Section 3.3.2. In Section 3.3.3, we discuss an additional technique to boost the performance of tree-structured models, and finally, we specify the entire details of our approach in Section 3.3.4.

3.3.1 Tree-LSTM

LSTM (Hochreiter and Schmidhuber, 1997) is originally invented as an extension of a sequential RNN, with the objective of mitigating the RNN's vanishing and exploding gradient problems. On top of that, several studies including Tai et al. (2015) propose to apply the module to tree-structured inputs in addition to sequential ones, giving birth to Tree-LSTM.

Formally, the composition function of a tree-LSTM cell is defined as follows:

$$\begin{bmatrix} \mathbf{i} \\ \mathbf{f}_l \\ \mathbf{f}_r \\ \mathbf{o} \\ \mathbf{g} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(\mathbf{W} \begin{bmatrix} \mathbf{h}_l \\ \mathbf{h}_r \end{bmatrix} + \mathbf{b} \right)$$

$$\mathbf{c} = \mathbf{f}_l \odot \mathbf{c}_l + \mathbf{f}_r \odot \mathbf{c}_r + \mathbf{i} \odot \mathbf{g}$$

$$\mathbf{h} = \mathbf{o} \odot \tanh(\mathbf{c}),$$

where $\mathbf{h}, \mathbf{c} \in \mathbb{R}^d$ indicate the hidden and cell state of the LSTM cell, and $\mathbf{h}_l, \mathbf{h}_r, \mathbf{c}_l, \mathbf{c}_r \in \mathbb{R}^d$ are the hidden states and cell states of the left and right child. $\mathbf{g} \in \mathbb{R}^d$ is the new input for the current cell, and $\mathbf{i}, \mathbf{f}_l, \mathbf{f}_r, \mathbf{o} \in \mathbb{R}^d$ represent the input gate, two forget gates (left, right), and output gate respectively. $\mathbf{W} \in \mathbb{R}^{5d \times 2d}$ and $\mathbf{b} \in \mathbb{R}^{5d}$ are trainable parameters. σ corresponds to the sigmoid function, tanh to the hyperbolic tangent, and $\odot$ to element-wise multiplication.

Note that the above formulation presumes that there only exist two children for each node in a tree (i.e., the tree is inherently binary or binarized), following the convention in the literature. While RvNN can be modeled on any tree structure, we here consider constituency parse trees as input.

Despite the obvious upside that recursive models have in being so flexible, they are notorious for their incompatibility with parallel computation due to the diversity of structure found across sentences. To alleviate this problem, Bowman et al. (2016) proposed SPINN, which introduces the shift-reduce algorithm into tree-LSTM. As SPINN reduces the process of constructing a tree into two simple operations—shift and reduce, it can support more effective parallel computations while enjoying the advantages of tree structures. For efficiency, our model also starts from our own re-implementation of SPINN, whose functions are exactly the same as those of the original tree-LSTM.

3.3.2 Structure-aware Tag Representation

In most of the previous work using linguistic tag information (Qian et al., 2015; Wang et al., 2017; Huang et al., 2017), tags were usually represented as simple low-dimensional dense vectors, similar to word embeddings. This approach seems reasonable in the case of POS tags which are attached to corresponding words, but phrase-level constituent tags (e.g., NP, VP, ADJP) vary greatly in size and shape, making them less amenable to uniform treatment. For instance, even the same phrase tags within different syntactic contexts can vary greatly in size and internal structure, as the case of NP tags in Figure 3.1 shows. Here, the NP consisting of DT[the]-NN[stories] has a different internal structure than the NP consisting of NP[the film 's]-NNS[shortcomings].

One possible avenue of deriving *structure-aware* tag representations is to introduce a special tag tree-LSTM that accepts static tag embeddings at each

node of a tree and outputs the corresponding structure-aware tag representations. Note that the module concentrates on extracting useful syntactic features by only considering the tags and structures of parse trees, excluding lexical (word-level) information.

Formally, we denote a typical, static tag embedding as $\mathbf{e} \in \mathbb{R}^{d_{\mathrm{T}}}$. Then, the function of each cell in a tag tree-LSTM is defined in two ways. First, the computation at leaf nodes is specified as follows:

$$\begin{bmatrix} \hat{\mathbf{c}} \\ \hat{\mathbf{h}} \end{bmatrix} = \tanh\left(\mathbf{U}_{\mathrm{T}}\mathbf{e} + \mathbf{a}_{\mathrm{T}}\right),$$

while the computation at non-leaf nodes is determined by the following:

$$\begin{bmatrix} \hat{\mathbf{i}} \\ \hat{\mathbf{f}}_l \\ \hat{\mathbf{f}}_r \\ \hat{\mathbf{o}} \\ \hat{\mathbf{g}} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(\mathbf{W}_{\mathrm{T}} \begin{bmatrix} \hat{\mathbf{h}}_l \\ \hat{\mathbf{h}}_r \\ \mathbf{e} \end{bmatrix} + \mathbf{b}_{\mathrm{T}} \right)$$

$$\hat{\mathbf{c}} = \hat{\mathbf{f}}_l \odot \hat{\mathbf{c}}_l + \hat{\mathbf{f}}_r \odot \hat{\mathbf{c}}_r + \hat{\mathbf{i}} \odot \hat{\mathbf{g}}$$

$$\hat{\mathbf{h}} = \hat{\mathbf{o}} \odot \tanh\left(\hat{\mathbf{c}}\right),$$

where $\hat{\mathbf{h}}, \hat{\mathbf{c}} \in \mathbb{R}^{d_{\mathrm{T}}}$ represent the hidden and cell state of each node in the tag tree-LSTM. We regard the hidden state ($\hat{\mathbf{h}}$) as the *structure-aware tag representation* for the node. $\mathbf{U}_{\mathrm{T}} \in \mathbb{R}^{2d_{\mathrm{T}} \times d_{\mathrm{T}}}$, $\mathbf{a}_{\mathrm{T}} \in \mathbb{R}^{2d_{\mathrm{T}}}$, $\mathbf{W}_{\mathrm{T}} \in \mathbb{R}^{5d_{\mathrm{T}} \times 3d_{\mathrm{T}}}$, and $\mathbf{b}_{\mathrm{T}} \in \mathbb{R}^{5d_{\mathrm{T}}}$ are trainable parameters. The other notations follow those defined in Section 3.3.1. In case of leaf nodes, the states ($\hat{\mathbf{h}}$ and $\hat{\mathbf{c}}$) are computed by a simple non-linear transformation. Meanwhile, the composition function in a non-leaf node absorbs a static tag embedding ($\mathbf{e}$) as an additional input as well as the hidden states of two children nodes. The benefit of contextualizing tag representations

Original Tags	Groups
NN, NP, NNP, NNS, NNPS, NX, WHNP	N
PRP, PRP$, WP, WP$, PRP, PRP$	PN
VP, VB, VBD, VBG, VBN, VBP, VBZ, MD	V
ADJP, JJ, JJR, JJS	ADJ
ADVP, WHADVP, RB, RBR, RBS, WRB	ADV
DT, EX, PDT, WDT	DET
IN	ADP
CC	CONJ
CD	NUM
!, #, $, ?, comma(,), colon(:), period(.), quotation marks(“, ”), -LRB-, -RRB-, LST, PRN	PUNC
PRT, PP, TO, POS, RP	PRT
Other word-level tags	X

Table 3.1 The universal POS tagset (Petrov et al., 2012) for word-level tags.

Original Tags	Groups
NP, @NP, NX, @NX, WHNP, @WHNP	NP
VP, @VP	VP
ADJP, @ADJP, WHADJP, @WHADJP	ADJP
ADVP, @ADVP, WHADVP, @WHADVP	ADVP
S, @S, SBAR, @SBAR, SQ, @SQ, SINV, @SINV	S
ROOT	ROOT
CONJP, @CONJP	CONJP
QP, @QP	NUMP
LST, @LST	PUNCP
PRT, @PRT, PP, @PP, WHPP, @WHPP	PRTP
Other phrase-level tags	XP

Table 3.2 Pre-defined categories for phrase-level tags.

according to the internal structure is that the derived embedding is a function of the corresponding makeup of the node, rather than a monolithic, categorical tag.

With regard to the granularity of syntactic tags, we conjecture that the taxonomy of the tags currently in use in many NLP systems is too complex to be effectively utilized in deep neural models, considering the specificity of many tag sets and the limited amount of data with which to train. Thus, we

cluster POS (word-level) tags into 12 groups following the universal POS tagset (Petrov et al., 2012), and phrase-level tags into 11 groups according to the criteria analogous to those of words, resulting in 23 tag categories in total. We refer readers to Table 3.1 and 3.2 for details.

3.3.3 Leaf-LSTM

Another inherent shortcoming of classical RvNNs is that the intermediate representation computed at each node of a tree is unaware of its external context until all the information is gathered together at the root node. In other words, each composition process in RvNN is prone to be stuck in local optima rather than globally considered. To mitigate this problem, we propose using a leaf-LSTM, following some previous work (Eriguchi et al., 2016; Yang et al., 2017; Choi et al., 2018), which is a typical LSTM that accepts a sequence of words in order. Instead of directly leveraging static word embeddings, we can make use of each hidden state and cell state of the leaf-LSTM as input for leaf nodes in a tree-LSTM, anticipating the proper contextualization of the input sequence.

Formally, we denote a sequence of words in an input sentence as $w_{1:n}$ (n: the length of the sentence) and corresponding word embeddings as $\mathbf{x}_{1:n}$. Then, a leaf-LSTM's operation at time t can be formulated as,

$$\begin{bmatrix} \tilde{\mathbf{i}} \\ \tilde{\mathbf{f}} \\ \tilde{\mathbf{o}} \\ \tilde{\mathbf{g}} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(\mathbf{W}_{\mathrm{L}} \begin{bmatrix} \tilde{\mathbf{h}}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b}_{\mathrm{L}} \right)$$

$$\tilde{\mathbf{c}}_t = \tilde{\mathbf{f}} \odot \tilde{\mathbf{c}}_{t-1} + \tilde{\mathbf{i}} \odot \tilde{\mathbf{g}}$$

$$\tilde{\mathbf{h}}_t = \tilde{\mathbf{o}} \odot \tanh\left(\tilde{\mathbf{c}}_t\right)$$

where $\mathbf{x}_t \in \mathbb{R}^{d_w}$ indicates the input word vector at time t, and $\tilde{\mathbf{h}}_t$, $\tilde{\mathbf{c}}_t \in \mathbb{R}^{d_h}$

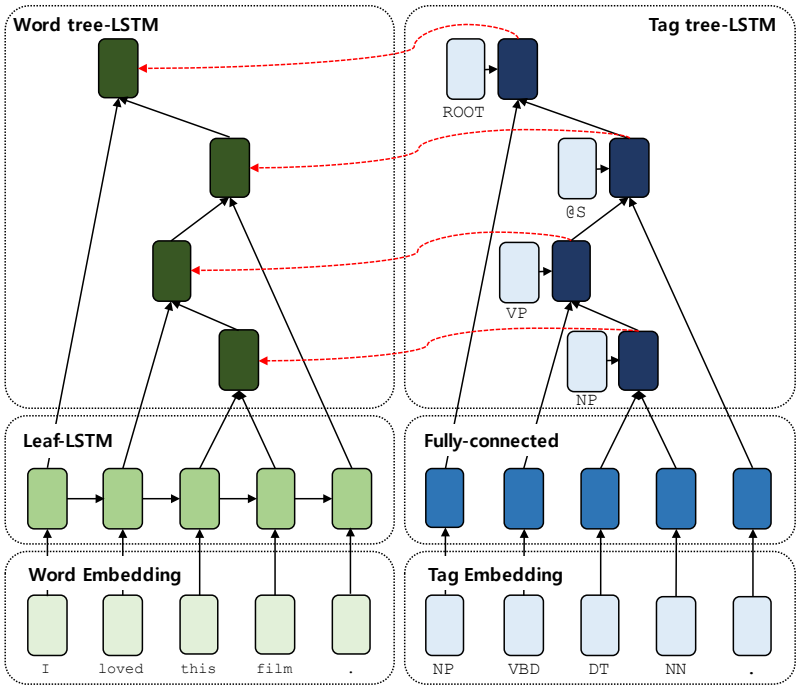

Figure 3.2 Illustration of SATA Tree-LSTM. It has two separate tree-LSTM modules, the right of which (tag tree-LSTM) computes structure-aware tag representations to control the composition function of the remaining tree-LSTM (word tree-LSTM). **Fully-connected**: an affine transformation followed by a non-linear function.

represent the hidden and cell state of the leaf-LSTM at time t (similarly, $\tilde{\mathbf{h}}_{t-1}$ is the hidden state at time t-1). $\mathbf{W}_{\mathrm{L}}$ and $\mathbf{b}_{\mathrm{L}}$ are learnable parameters.

In experiments, we demonstrate that introducing a leaf-LSTM fares better at processing the input words of a tree-LSTM, compared to using a feed-forward neural network. We also explore the possibility of its bidirectional setting in ablation study.

3.3.4 SATA Tree-LSTM

Finally, we formulate **SATA Tree-LSTM** (**S**tructure-**A**ware **T**ag **A**ugmented **Tree-LSTM**; Figure 3.2) which combines tag tree-LSTM (Section 3.3.2), leaf-

LSTM (Section 3.3.3), and word tree-LSTM (Section 3.3.1) together.

In formal, we derive the final sentence representation for an input sentence in two steps. First, we compute structure-aware tag representations ($\hat{\mathbf{h}}$) for each node of the input tree using the tag tree-LSTM (the right side of Figure 3.2) as follows:

$$\begin{bmatrix} \hat{\mathbf{c}} \\ \hat{\mathbf{h}} \end{bmatrix} = \begin{cases} \text{Tag Tree-LSTM}(\mathbf{e}) & \text{if a leaf node} \\ \text{Tag Tree-LSTM}(\hat{\mathbf{h}}_l, \hat{\mathbf{h}}_r, \mathbf{e}) & \text{otherwise} \end{cases}$$

where Tag-Tree-LSTM indicates the module we described in Section 3.3.2.

Second, we merge semantic units recursively on the word tree-LSTM in a bottom-up fashion. For leaf nodes, we leverage the Leaf-LSTM (the bottom-left of Figure 3.2; explained in Section 3.3.3) to compute $\tilde{\mathbf{c}}_t$ and $\tilde{\mathbf{h}}_t$ in sequential order with the static word embedding $\mathbf{x}_t$.

$$\begin{bmatrix} \tilde{\mathbf{c}}_t \\ \tilde{\mathbf{h}}_t \end{bmatrix} = \text{Leaf-LSTM}(\tilde{\mathbf{h}}_{t-1}, \mathbf{x}_t).$$

Then, $\tilde{\mathbf{c}}_t$ and $\tilde{\mathbf{h}}_t$ are inserted into the word tree-LSTM such that the non-leaf node whose left/right child is w_t accepts $\tilde{\mathbf{c}}_t$ and $\tilde{\mathbf{h}}_t$ as input:

$$\begin{bmatrix} \check{\mathbf{c}}_{\{l,r\}} \\ \check{\mathbf{h}}_{\{l,r\}} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{c}}_t \\ \tilde{\mathbf{h}}_t \end{bmatrix}.$$

In the case of other non-leaf nodes (the upper-left of Figure 3.2), we recursively calculate phrase representations for respective nodes as follows:

$$\check{\mathbf{g}} = \tanh\left(\mathbf{U}_{\text{w}} \begin{bmatrix} \check{\mathbf{h}}_l \\ \check{\mathbf{h}}_r \end{bmatrix} + \mathbf{a}_{\text{w}}\right)$$

$$\begin{bmatrix} \check{\mathbf{i}} \\ \check{\mathbf{f}}_l \\ \check{\mathbf{f}}_r \\ \check{\mathbf{o}} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(\mathbf{W}_{\text{w}} \begin{bmatrix} \check{\mathbf{h}}_l \\ \check{\mathbf{h}}_r \\ \hat{\mathbf{h}} \end{bmatrix} + \mathbf{b}_{\text{w}}\right)$$

Dataset	$\mathbf{d}_{\mathrm{T}}$	$\mathbf{d}_h$	$\mathbf{d}_s$	Optimizer	L	B	E	W	D	Word Emb. Fine-tuning
SST-2	50	100	500	Adadelta	1	64	20	1e-5	0.5	Y
SST-5	100	300	1200	Adadelta	1	64	10	5e-6	0.5	Y
MR	25	100	500	Adadelta	1	64	30	1e-5	0.5	Y
SUBJ	25	150	1000	Adadelta	1	64	40	2e-5	0.5	Y
TREC	50	300	800	Adadelta	1	64	40	5e-5	0.5	Y
SNLI	100	300	1200	Adam	1e-3	64	10	1e-5	0.1	N

Table 3.3 Hyperparameters for respective datasets. $\mathbf{d}_{\mathrm{T}}$: Dimension of tag embeddings. $\mathbf{d}_h$: Dimension of a word tree-LSTM cell. $\mathbf{d}_s$: Dimension of linear classifiers. **L**: Learning rate. **B**: Batch size. **E**: Maximum epochs. **W**: Weight decay rate. **D**: Dropout (drop) probability. **Word Emb. Fine-tuning**: Whether word embeddings are fine-tuned during training or not (Y: fine-tuned, N: fixed).

$$\check{\mathbf{c}} = \check{\mathbf{f}}_l \odot \check{\mathbf{c}}_l + \check{\mathbf{f}}_r \odot \check{\mathbf{c}}_r + \check{\mathbf{i}} \odot \check{\mathbf{g}}$$

$$\check{\mathbf{h}} = \check{\mathbf{o}} \odot \tanh\left(\check{\mathbf{c}}\right)$$

where $\check{\mathbf{h}}$, $\check{\mathbf{c}} \in \mathbb{R}^{d_h}$ represent the hidden and cell state of each node in the word tree-LSTM. $\mathbf{U}_{\mathrm{w}} \in \mathbb{R}^{d_h \times 2d_h}$, $\mathbf{W}_{\mathrm{w}} \in \mathbb{R}^{4d_h \times (2d_h + d_{\mathrm{T}})}$, $\mathbf{a}_{\mathrm{w}} \in \mathbb{R}^{d_h}$, $\mathbf{b}_{\mathrm{w}} \in \mathbb{R}^{4d_h}$ are learned parameters. Note that the structure-aware tag representation ($\hat{\mathbf{h}}$) is restrictively utilized to control the gate functions of the word tree-LSTM in the form of additional inputs, and are not involved in the semantic composition ($\check{\mathbf{g}}$) directly. Lastly, the hidden state of the root node ($\check{\mathbf{h}}_{\mathrm{root}}$) in the word tree-LSTM becomes the final sentence representation of the input sentence.

3.4 Experiments

3.4.1 General Configurations

For static word embeddings, we initialize them with 300 dimensional 840B GloVe (Pennington et al., 2014) and fine-tune them depending on the task. On the other hand, tag representations are randomly initialized using the uniform distribution [-0.005, 0.005] and tuned during training. Our model is trained using the Adam (Kingma and Ba, 2014) or Adadelta (Zeiler, 2012) optimizer.

Dataset	$\mathbf{d_c}$	l	# Train	# Dev	# Test
SST-2	2	19	93,517	872	1,821
SST-5	5	18	300,192	1,101	2,210
MR	2	20	10,662	-	CV
SUBJ	2	23	10,000	-	CV
TREC	6	10	5,452	-	500
SNLI	3	11	550,152	10,000	10,000

Table 3.4 Statistics for sentence classification datasets. $\mathbf{d}_c$: Number of classes. $\boldsymbol{l}$: Average sentence length. # **Train**, # **Dev**, # **Test**: Number of sentences in the training, validation, and test set respectively. **CV**: 10-fold cross validation.

For regularization, the weight decay technique is added (except for SNLI) to the cross-entropy loss function following Loshchilov and Hutter (2017). Dropout (Srivastava et al., 2014) is also applied for word embeddings and task-specific classifiers, and batch normalization (Ioffe and Szegedy, 2015) is adopted for classifiers. All trainable parameters except the embeddings are initialized following He et al. (2015) and biases are set to 0. The total L2 norm of the gradients of the parameters is clipped not to be over 5 during training. Our best instances are selected by their validation accuracy in cases where the validation set is provided. Otherwise, we perform a grid search on a reasonable range of hyperparameters. We run 10-fold cross-validation when the test set does not exist. The selected hyperparameters are listed in Table 3.3.

3.4.2 Sentence Classification Tasks

One of the most basic approaches to evaluating a sentence encoder is to measure its performance on downstream tasks such as classification. We thus conduct experiments on five classification datasets (see Table 3.4 for the statistics of the datasets):

- **MR** (Pang and Lee, 2005): A group of movie reviews with binary (positive / negative) classes.

- **SST-2** (Socher et al., 2013b): Stanford Sentiment Treebank; Similar to MR, but each review is provided in the form of a binary parse tree whose nodes are annotated with numeric sentiment values. For SST-2, we only consider binary (positive / negative) classes.

- **SST-5**: Identical to SST-2, but the reviews are grouped into fine-grained (very negative, negative, neutral, positive, and very positive) classes. Note that we use the subtrees of parse trees in addition to the whole parse trees when training models for SST-2 and SST-5, following the standard in the literature.

- **SUBJ** (Pang and Lee, 2004): Sentences grouped as being either subjective or objective (binary classes).

- **TREC** (Li and Roth, 2002): A dataset that groups questions into six different question types (classes).

As a preprocessing step, we parse sentences in the datasets by leveraging the Stanford PCFG parser (Klein and Manning, 2003).

To determine a sentence's class given its sentence representation ($\check{\mathbf{h}}_{\text{root}}$), we utilize one fully-connected layer with a ReLU activation, followed by a softmax classifier. The prediction for the probability of each class y given the sentence $w_{1:n}$ is defined as:

$$\mathbf{s} = \text{ReLU}(\mathbf{W}_{\text{s}}\check{\mathbf{h}}_{\text{root}} + \mathbf{b}_{\text{s}})$$

$$p(y|w_{1:n}) = \text{softmax}(\mathbf{W}_{\text{c}}\mathbf{s} + \mathbf{b}_{\text{c}})$$

where $\mathbf{s} \in \mathbb{R}^{d_{\text{s}}}$ is the task-specific sentence representation for the softmax classifier, and $\mathbf{W}_{\text{s}} \in \mathbb{R}^{d_{\text{s}} \times d_h}$, $\mathbf{W}_{\text{c}} \in \mathbb{R}^{d_{\text{c}} \times d_s}$, $\mathbf{b}_{\text{s}} \in \mathbb{R}^{d_s}$, $\mathbf{b}_{\text{c}} \in \mathbb{R}^{d_c}$ are trainable parameters. We employ the cross entropy as the objective function.

Models / Tasks	**SST-2**	**SST-5**	**MR**	**SUBJ**	**TREC**
RvNNs					
RNTN (Socher et al., 2013b)	85.4	45.7	-	-	-
AdaMC-RNTN (Dong et al., 2014)	88.5	46.7	-	-	-
TE-RNTN (Qian et al., 2015)	87.7	49.8	-	-	-
TBCNN (Mou et al., 2015)	87.9	51.4	-	-	96.0
Tree-LSTM (Tai et al., 2015)	88.0	51.0	-	-	-
AdaHT-LSTM-CM (Liu et al., 2017a)	87.8	50.2	81.9	94.1	-
DC-TreeLSTM (Liu et al., 2017b)	87.8	-	81.7	93.7	93.8
TE-LSTM (Huang et al., 2017)	89.6	52.6	82.2	-	-
BiConTree (Teng and Zhang, 2017)	90.3	53.5	-	-	94.8
Gumbel Tree-LSTM* (Choi et al., 2018)	90.7	53.7	-	-	-
TreeNet (Cheng et al., 2018)	-	-	83.6	<u>95.9</u>	96.1
SATA Tree-LSTM (Ours)	**91.3**	**54.4**	**83.8**	**95.4**	**<u>96.2</u>**
Other neural models					
CNN (Kim, 2014)	88.1	48.0	81.5	93.4	93.6
AdaSent (Zhao et al., 2015)	-	-	83.1	95.5	92.4
LSTM-CNN (Zhou et al., 2016)	89.5	52.4	82.3	94.0	96.1
byte-mLSTM (Radford et al., 2017)	<u>91.8</u>	52.9	<u>86.9</u>	94.6	-
BCN + Char + CoVe (McCann et al., 2017)	90.3	53.7	-	-	95.8
BCN + Char + ELMo (Peters et al., 2018)	-	<u>54.7</u>	-	-	-

Table 3.5 Comparison of various sentence representation models on five classification tasks. SATA Tree-LSTM shows competitive or comparable performance compared to previous RvNNs as well as other sophisticated neural models. The best figure for each dataset is <u>underlined</u>. $\star$: Latent tree-structured models.

Experimental results on the five datasets are shown in Table 3.5, where we report the test accuracy (percentage) of our model and other different models on each dataset. Compared against a series of parse tree-based RvNNs as well as other neural models, SATA Tree-LSTM shows superior or comparable performance on all the tasks we considered. Specifically, within the tree-structured model class, our model achieves new state-of-the-art results on 4 out of 5 sentence classification tasks—SST-2, SST-5, MR, and TREC. The model shows its strength, in particular, when datasets provide phrase-level supervision (i.e., SST-2 and SST-5). Moreover, we demonstrate that SATA Tree-LSTM's performance in SST2, SST-5 and TREC is competitive to that of large pre-trained models such as CoVe (McCann et al., 2017) and ELMo (Peters et al., 2018), con-

firming its effectiveness. Note also that SATA Tree-LSTM outperforms Gumbel Tree-LSTM which is a latent RvNN, indicating that modeling RvNN with explicit linguistic knowledge can be a reasonable choice.

3.4.3 Natural Language Inference

This time, we aim to evaluate the efficacy of SATA Tree-LSTM on sentence-pair datasets. To be specific, we conduct experiments on the Stanford Natural Language Inference (SNLI; Bowman et al. (2015)) dataset, each example of which consists of a premise and hypothesis. The objective is to predict the correct relationship between the two sentences among three options—contradiction, neutral, or entailment.

We utilize the Siamese architecture to encode both the premise ($p_{1:m}$) and hypothesis ($h_{1:n}$). Specifically, the premise $p_{1:m}$ is encoded as $\check{\mathbf{h}}^p_{\text{root}} \in \mathbb{R}^{d_h}$ while the hypothesis $h_{1:n}$ is encoded as $\check{\mathbf{h}}^h_{\text{root}} \in \mathbb{R}^{d_h}$ with the same encoder. We then introduce some heuristics (Mou et al., 2016) that facilitates the interaction between $\check{\mathbf{h}}^p_{\text{root}}$ and $\check{\mathbf{h}}^h_{\text{root}}$, followed by one fully-connected layer with a ReLU activation and a softmax classifier. Namely,

$$\mathbf{z} = \left[\check{\mathbf{h}}^p_{\text{root}}; \check{\mathbf{h}}^h_{\text{root}}; |\check{\mathbf{h}}^p_{\text{root}} - \check{\mathbf{h}}^h_{\text{root}}|; \check{\mathbf{h}}^p_{\text{root}} \odot \check{\mathbf{h}}^h_{\text{root}}\right]$$

$$\mathbf{s} = \text{ReLU}(\mathbf{W}_{\text{s}}\mathbf{z} + \mathbf{b}_{\text{s}})$$

$$p(y|p_{1:m}, h_{1:n}) = \text{softmax}(\mathbf{W}_{\text{c}}\mathbf{s} + \mathbf{b}_{\text{c}}),$$

where $\mathbf{z} \in \mathbb{R}^{4d_h}$, $\mathbf{s} \in \mathbb{R}^{d_s}$ are intermediate features for the classifier and $\mathbf{W}_{\text{s}} \in \mathbb{R}^{d_{\text{s}} \times 4d_h}$, $\mathbf{W}_{\text{c}} \in \mathbb{R}^{d_{\text{c}} \times d_s}$, $\mathbf{b}_{\text{s}} \in \mathbb{R}^{d_s}$, $\mathbf{b}_{\text{c}} \in \mathbb{R}^{d_c}$ are trainable parameters.

Our experimental results on the SNLI dataset are shown in Table 3.5. We here report the test accuracy and number of trainable parameters for each model. SATA-LSTM again demonstrates its decent performance, compared against syntactic and latent RvNNs as well as non-tree models. Note that the

Models	Accuracy(%)	# Params
RvNNs		
100D Latent Syntax Tree-LSTM* (Yogatama et al., 2017)	80.5	500K
300D Tree-based CNN (Mou et al., 2016)	82.1	3.5M
300D SPINN-PI (Bowman et al., 2016)	83.2	3.7M
300D Gumbel Tree-LSTM* (Choi et al., 2018)	85.6	2.9M
300D SATA Tree-LSTM (Ours)	**85.9**	**3.3M**
Other neural models		
300D NSE (Munkhdalai and Yu, 2017)	84.6	3.0M
300D Reinforced Self-Attention Network (Shen et al., 2018a)	86.3	3.1M
600D Residual stacked encoders (Nie and Bansal, 2017)	86.0	29M
600D BiLSTM with generalized pooling (Chen et al., 2018)	<u>86.6</u>	65M

Table 3.6 Performance of diverse sentence embedding models on SNLI. SATA Tree-LSTM attains competitive performance with a moderate number of parameters. The best performance is <u>underlined</u>. $\star$: Latent tree models.

number of parameters in our model is also comparable to those of other sophisticated models, showing its efficiency. Even though our model has proven its mettle, the effect of tag information seems relatively weak in the case of SNLI, which contains a relatively large amount of training data compared to the datasets used in Section 3.4.2. One plausible explanation about this phenomenon is that neural models may autonomously learn some syntactic concepts from a gigantic text corpus when the size of the corpus is sufficiently large, reducing the necessity of external linguistic knowledge. We leave thorough exploration on this issue as future work.

3.5 Analysis

3.5.1 Ablation Study

We design an ablation study on SATA Tree-LSTM's core modules to estimate their contributions to the final performance. We replace a target module with possible alternatives while other parts remain fixed. The specific targets considered are the (i) leaf-LSTM and (ii) tag tree-LSTM (structure-aware tag embed-

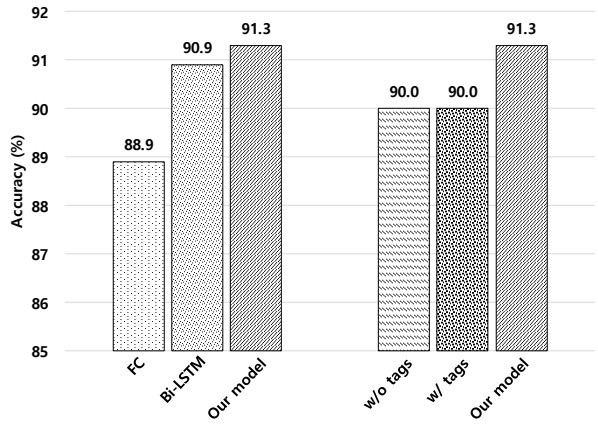

Figure 3.3 Ablation study on SATA Tree-LSTM's core modules. The results show that each part in SATA-Tree LSTM plays a crucial role in achieving the optimal performance. **FC**: a fully connected-layer with a tanh function. **w/o tags**: tag information is not considered at all. **w/ tags**: static tag embeddings are employed instead of structure-aware ones.

dings). In the first case, the leaf-LSTM is replaced with a fully-connected layer (with a tanh activation) or a Bi-LSTM. In the second case, we switch structure-aware tag embeddings to static tag embeddings or do not employ both of them at all.

In Figure 3.3, we present the performance of the modifications on SST2. Overall, we confirm that the original SATA Tree-LSTM outperforms all the other options. In detail, we first justify our decision in formulating the leaf-LSTM module in the left graph of Figure 3.3. It is worth noting that the unidirectional leaf-LSTM turns out to be more effective than its bidirectional counterpart when they have the exactly same number of parameters—i.e., the hidden dimension size of the bidirectional LSTM is the half of that of the unidirectional. We conjecture this is because the gain we earn by introducing the backward portion of the bidirectional LSTM is subsumed by the benefit of explicitly considering sentences' syntactic structures. This interpretation is

reasonable in that a backward LSTM can be regarded as a special case of Tree-LSTM that receives the right-branching tree as input. In conclusion, we adopt a typical LSTM as the leaf module because of its simplicity and competitive performance.

Next, the right part in Figure 3.3 demonstrates that structure-aware embeddings have a genuine influence on improving performance. Interestingly, employing static tag embeddings makes no difference in terms of the test accuracy, although we observe a small increase in the validation accuracy (which is not reported in the figure). This outcome strongly supports our claim that the knowledge of syntactic tags should be processed within the tree structure which they come from.

3.5.2 Representation Visualization

So far, we have mostly concentrated on verifying SATA Tree-LSTM's superiority in a quantitative manner. In this section, we turn our attention to qualitative analysis and attempt to interpret its inner workings by visualizing some phrase representations computed by the model. To be concrete, we select a sentence, insert its parse tree into SATA Tree-LSTM for computing the representation for each node in the tree, and finally draw a scatter plot where each point corresponds to the representation. As the target, we reuse the sentence encountered in Figure 3.1—[Despite the films 's shortcomings , the stories are quietly moving .]—and utilize Principal Component Analysis (PCA) to project the representations into a two-dimensional vector space. In addition, we draw supplementary (red) lines to indicate the relationship between each two nodes in the parse tree.

From Figure 3.4, we observe that the visualized representations are hierarchically structured in the space, mimicking their positions in the original parse tree. We also see the tendency that the low-level representations, located in

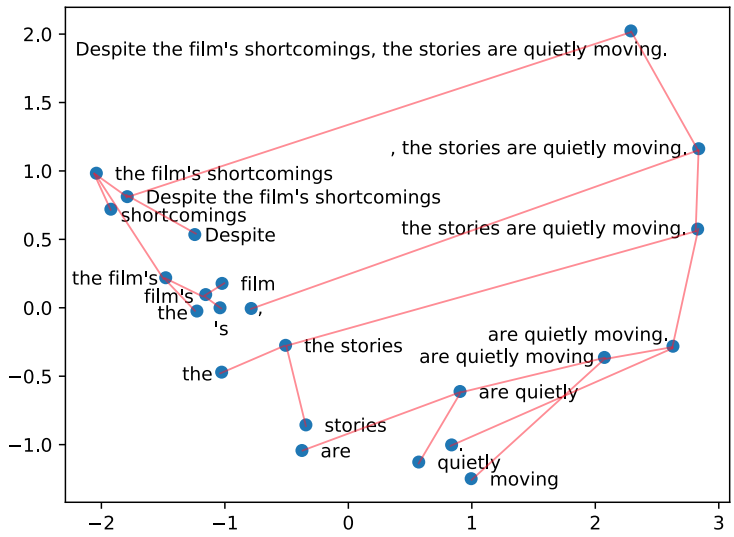

Figure 3.4 Scatter plot where each point corresponds to the embedding of each phrase appeared in the parse in Figure 3.1. We additionally draw red lines to visualize the tree structure. We observe that the embeddings are hierarchically structured in the space, mimicking their positions in the original parse tree.

the bottom-left of the figure, are recursively combined such that the resulting high-level embeddings are moved towards the relatively upper and right area. The last finding is that the final sentence representation is closer to that of the phrase [, the stories are quietly moving .] than that of the phrase [Despite the film's shortcomings], successfully catching the sentence's main theme. To conclude, we confirm via this qualitative analysis that SATA Tree-LSTM's intermediate representations in fact reflect the structure of sentences in their semantic space, as desired in our model's formulation.

3.6 Limitations and Future Work

First, we assumed in Section 3.3.2 that our redefined, clustered classes of syntactic tags would be useful. Even though the outcome of the proposed model

relied on the taxonomy turns out to be satisfying, it has not been validated yet whether the model in fact benefited from the clustering. Therefore, it is desired to verify its independent effectiveness in future work.

Second, the overall evaluation conducted in Section 3.4 is biased towards a sort of classification tasks, although a sentence encoder in principle can be utilized for diverse NLP tasks such as machine translation and question answering. A possible following study would be thus testing the impact of our approach in more broader categories of downstream tasks.

3.7 Summary

In this chapter, we have proposed a novel RvNN architecture, SATA Tree-LSTM, as a guidance on how to fully exploit the rich information presented by phrase-structure trees in constructing effective sentence representations. We have demonstrated with extensive experiments that by introducing an independent tag tree-LSTM that governs syntactic information coming from both the structure and tags of parse trees, we can better control the semantic composition of constituents in RvNNs. This result implies that despite the recent prosperity of data-driven approaches in NLP, linguistic knowledge can still take charge of an important role as an assistant who offers a useful inductive bias to neural models. As our method is, in theory, also applicable for other types of syntactic structures besides constituency trees, a viable next step as future work would be to explore its applicability on dependency parse trees.

Chapter 4

Sentence Representation Learning with Implicit Syntactic Knowledge

4.1 Introduction

In the last chapter, we have introduced the methodology of better exploiting syntax information for constructing sentence representations when such knowledge is provided as an explicit form of constituency parse trees. By contrast, we assume in this chapter that we only have access to pre-trained language models (PLMs), which implicitly store a variety of linguistic knowledge, including syntactic one, in their parameters. Inspired from the recent discovery by several studies (Jawahar et al., 2019) that the lower and middle layers of PLMs are relatively more sensitive to syntax while the upper layers are good at capturing semantics, we propose a method for computing comprehensive sentence representations by dynamically considering all the intermediate layers of PLMs.

Recently, pre-trained Transformer (Vaswani et al., 2017) language models such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019b) have been

integral to achieving recent improvements in natural language understanding. However, it is not straightforward to directly utilize these models for sentence-level tasks, as they are basically pre-trained to focus on predicting (sub)word tokens given context. The most typical way of converting the models into sentence encoders is to fine-tune them with supervision from a downstream task. In the process, as initially proposed by Devlin et al. (2019), a pre-defined token's (a.k.a. `[CLS]`) embedding from the last layer of the encoder is deemed as the representation of an input sequence. This simple but effective method is possible because, during supervised fine-tuning, the `[CLS]` embedding functions as the only communication gate between the pre-trained encoder and a task-specific layer, encouraging the `[CLS]` vector to capture the holistic information.

On the other hand, in cases where labeled datasets are unavailable, it is unclear what the best strategy is for deriving sentence embeddings from BERT.[1] In practice, previous studies (Reimers and Gurevych, 2019; Li et al., 2020; Hu et al., 2020) reported that naïvely (i.e., without any processing) leveraging the `[CLS]` embedding as a sentence representation, as is the case of supervised fine-tuning, results in disappointing outcomes. Currently, the most common rule of thumb for building BERT sentence embeddings without supervision is to apply mean pooling on the last layer(s) of BERT. Yet, this approach can be still sub-optimal. In a preliminary experiment, we constructed sentence embeddings by employing various combinations of different BERT layers and pooling methods, and tested them on the Semantic Textual Similarity (STS) benchmark dataset (Cer et al., 2017).[2] We discovered that BERT(-base)'s performance, measured in Spearman correlation ($\times$ 100), can range from as low as 16.71 (`[CLS]`, the 10^{th}

[1] In this chapter, the term *BERT* has two meanings: Narrowly, the BERT model itself, and more broadly, pre-trained Transformer encoders that share the same spirit with BERT.

[2] In the experiment, we employ the settings identical with ones used in Chapter 4.4. Refer to Chapter 4.4 for more details.

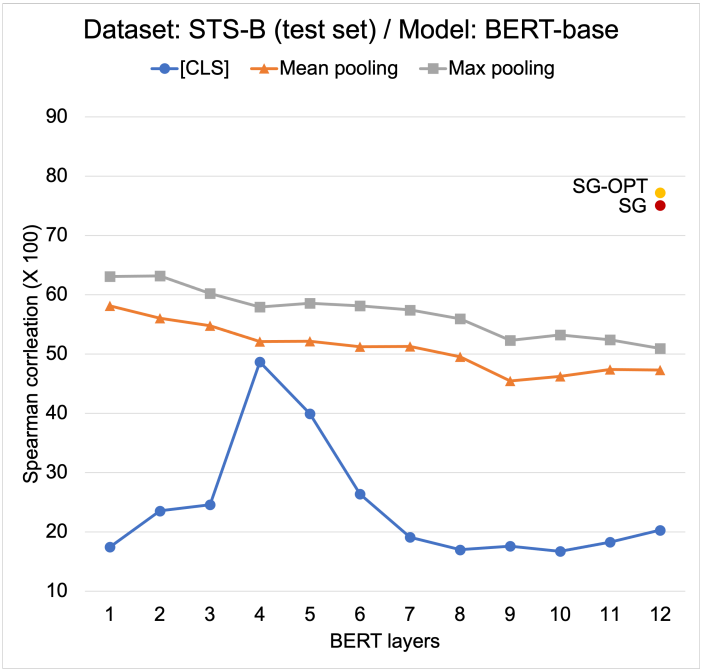

Figure 4.1 BERT(-base)'s layer-wise performance with different pooling methods on the STS-B test set. We observe that the performance can be dramatically varied according to the selected layer and pooling strategy. Our self-guided training (SG / SG-OPT) assures much improved results compared to those of the baselines.

layer) to 63.19 (max pooling, the 2$^{\text{nd}}$ layer) depending on the selected layer and pooling method (see Figure 4.1). This result suggests that the current practice of building BERT sentence vectors is not solid enough, and that there is room to bring out more of BERT's expressiveness.

In this chapter, we propose a contrastive learning method that makes use of a newly proposed self-guidance mechanism to tackle the aforementioned problem. The core idea is to recycle intermediate BERT hidden representations as

positive samples to which the final sentence embedding should be close. As our method does not require data augmentation, which is essential in most recent contrastive learning frameworks, it is much simpler and easier to use than existing methods (Fang and Xie, 2020; Xie et al., 2020). Moreover, we customize the NT-Xent loss (Chen et al., 2020), a contrastive learning objective widely used in computer vision, for better sentence representation learning with BERT. We demonstrate that our approach outperforms competitive baselines designed for building BERT sentence vectors (Li et al., 2020; Wang and Kuo, 2020) in various environments. With comprehensive analyses, we also show that our method is more computationally efficient than the baselines at inference in addition to being more robust to domain shifts.

4.2 Related Work

Contrastive Representation Learning. Contrastive learning has been long considered as effective in constructing meaningful representations. For instance, Mikolov et al. (2013b) propose to learn word embeddings by framing words nearby a target word as positive samples while others as negative. Logeswaran and Lee (2018) generalize the approach of Mikolov et al. (2013b) for sentence representation learning. More recently, several studies (Fang and Xie, 2020; Giorgi et al., 2020; Wu et al., 2020) suggest to utilize contrastive learning for training Transformer models, similar to our approach. However, they generally require data augmentation techniques, e.g., back-translation (Sennrich et al., 2016), or prior knowledge on training data such as order information, while our method does not. Furthermore, we focus on revising BERT for computing better sentence embeddings rather than training a language model from scratch.

On the other hand, contrastive learning has been also receiving much attention from the computer vision community (Chen et al. (2020); Chen and He

(2020); He et al. (2020), *inter alia*). We improve the framework of Chen et al. (2020) by optimizing its learning objective for pre-trained Transformer-based sentence representation learning. For extensive surveys on contrastive learning, refer to Le-Khac et al. (2020) and Jaiswal et al. (2020).

Fine-tuning BERT with Supervision. It is not always trivial to fine-tune pre-trained Transformer models of gigantic size with success, especially when the number of target domain data is limited (Mosbach et al., 2020). To mitigate this training instability problem, several approaches (Aghajanyan et al., 2020; Jiang et al., 2020; Zhu et al., 2020) have been recently proposed. In particular, Gunel et al. (2021) propose to exploit contrastive learning as an auxiliary training objective during fine-tuning BERT with supervision from target tasks. In contrast, we deal with the problem of adjusting BERT when such supervision is not available.

Sentence Embeddings from BERT. Since BERT and its variants are originally designed to be fine-tuned on each downstream task to attain their optimal performance, it remains ambiguous how best to extract general sentence representations from them, which are broadly applicable across diverse sentence-related tasks. Following Conneau et al. (2017), Reimers and Gurevych (2019) (SBERT) propose to compute sentence embeddings by conducting mean pooling on the last layer of BERT and then fine-tuning the pooled vectors on the natural language inference (NLI) datasets (Bowman et al., 2015; Williams et al., 2018b). Meanwhile, some other studies concentrate on more effectively leveraging the knowledge embedded in BERT to construct sentence embeddings without supervision. Specifically, Wang and Kuo (2020) propose a pooling method based on linear algebraic algorithms to draw sentence vectors from BERT's interme-

diate layers. Li et al. (2020) suggest to learn a mapping from the average of the embeddings obtained from the last two layers of BERT to a spherical Gaussian distribution using a flow model, and to leverage the redistributed embeddings in place of the original BERT representations. We follow the setting of Li et al. (2020) in that we only utilize plain text during training, however, unlike all the others that rely on a certain pooling method even after training, we directly refine BERT so that the typical `[CLS]` vector can function as a sentence embedding. Note also that there exists concurrent work (Carlsson et al., 2021; Gao et al., 2021; Wang et al., 2021) whose motivation is analogous to ours, attempting to improve BERT sentence embeddings in an unsupervised fashion.

4.3 Method

As BERT mostly requires some type of adaptation to be properly applied to a task of interest, it might not be desirable to derive sentence embeddings directly from BERT without fine-tuning. While Reimers and Gurevych (2019) attempt to alleviate this problem with typical supervised fine-tuning, we restrict ourselves to revising BERT in an unsupervised manner, meaning that our method only demands a bunch of raw sentences for training.

Among possible unsupervised learning strategies, we concentrate on contrastive learning which can inherently motivate BERT to be aware of similarities between different sentence embeddings. Considering that sentence vectors are widely used in computing the similarity of two sentences, the inductive bias introduced by contrastive learning can be helpful for BERT to work well on such tasks. The problem is that sentence-level contrastive learning usually requires data augmentation (Fang and Xie, 2020) or prior knowledge on training data, e.g., order information (Logeswaran and Lee, 2018), to make plausible positive/negative samples. We attempt to circumvent these constraints by utilizing

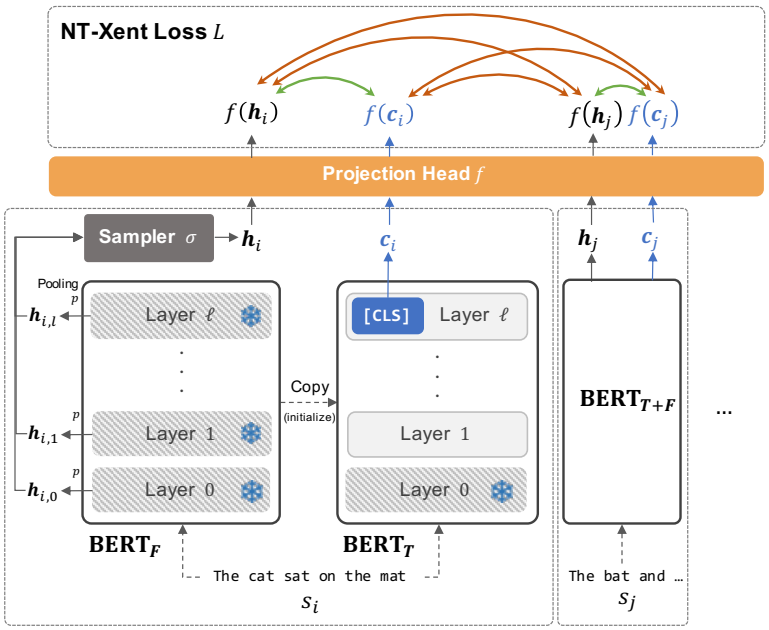

Figure 4.2 Self-guided contrastive learning framework. We clone BERT into two copies at the beginning of training. BERT_T (except Layer 0) is then fine-tuned to optimize the sentence vector $\mathbf{c}_i$ while BERT_F is fixed.

the hidden representations of BERT, which are readily accessible, as samples in the embedding space.

4.3.1 Contrastive Learning with Self-Guidance

We aim at developing a contrastive learning method that is free from external procedure such as data augmentation. A possible solution is to leverage (virtual) adversarial training (Miyato et al., 2018) in the embedding space. However, there is no assurance that the semantics of a sentence embedding would remain unchanged when it is added with a random noise. As an alternative, we propose to utilize the hidden representations from BERT's intermediate layers, which are conceptually guaranteed to represent corresponding sentences, as pivots that

BERT sentence vectors should be close to or be away from. We call our method as **self-guided contrastive learning** since we exploit the internal training signals made by BERT itself to fine-tune it.

We describe our training framework in Figure 4.2. First, we clone BERT into two copies, BERT_F (*fixed*) and BERT_T (*tuned*) respectively. BERT_F is fixed during training to provide a training signal while BERT_T is fine-tuned to construct better sentence embeddings. The reason why we differentiate BERT_F from BERT_T is that we want to prevent the training signal computed by BERT_F from being degenerated as the training procedure continues, which often happens when $\text{BERT}_F = \text{BERT}_T$. This design decision also reflects our philosophy that our goal is to dynamically conflate the knowledge stored in BERT's different layers to produce sentence embeddings, rather than introducing new information via extra training. Note that in our setting, the `[CLS]` vector from the last layer of BERT_T, i.e., $\mathbf{c}_i$, is regarded as the final sentence embedding we aim to optimize/utilize during/after fine-tuning.

Second, given b sentences in a mini-batch, say $s_1, s_2, \cdots, s_b$, we feed each sentence s_i into BERT_F and compute token-level hidden representations $H_{i,k} \in \mathbb{R}^{len(s_i) \times d}$:

$$[H_{i,0}; H_{i,1}; \cdots; H_{i,k}; \cdots; H_{i,l}] = \text{BERT}_F(s_i),$$

where $0 \leq k \leq l$ (0: the non-contextualized layer), l is the number of hidden layers in BERT, $len(s_i)$ is the length of the tokenized sentence, and d is the size of BERT's hidden representations. Then, we apply a pooling function p to $H_{i,k}$ for deriving diverse sentence-level views $\mathbf{h}_{i,k} \in \mathbb{R}^d$ from all layers, i.e., $\mathbf{h}_{i,k} = p(H_{i,k})$. Finally, we choose the final view to be utilized by applying a sampling function σ:

$$\mathbf{h}_i = \sigma(\{\mathbf{h}_{i,k} | 0 \leq k \leq l\}).$$

As we have no specific constraints in defining p and σ, we employ max pooling as p and a uniform sampler as σ for simplicity, unless otherwise stated. This simple choice for the sampler implies that each $\mathbf{h}_{i,k}$ has the same importance, which is persuasive considering it is known that different BERT layers are specialized at capturing disparate linguistic concepts (Jawahar et al., 2019).[3]

Third, we compute our sentence embedding $\mathbf{c}_i$ for s_i as follows:

$$\mathbf{c}_i = \text{BERT}_T(s_i)_{[\text{CLS}]},$$

where $\text{BERT}(\cdot)_{[\text{CLS}]}$ corresponds to the `[CLS]` vector obtained from the last layer of BERT. Next, we collect the set of the computed vectors into $X = \{\mathbf{x}|\mathbf{x} \in \{\mathbf{c}_i\} \cup \{\mathbf{h}_i\}\}$, and for all $\mathbf{x}_m \in X$, we compute the NT-Xent loss (Chen et al., 2020):

$$L_m^{base} = -\log\left(\phi(\mathbf{x}_m, \mu(\mathbf{x}_m))/Z\right),$$
$$\text{where } \phi(\mathbf{u}, \mathbf{v}) = \exp(g(f(\mathbf{u}), f(\mathbf{v}))/\tau)$$
$$\text{and } Z = \textstyle\sum_{n=1, n\neq m}^{2b} \phi(\mathbf{x}_m, \mathbf{x}_n).$$

Note that τ is a temperature hyperparameter, f is a projection head consisting of MLP layers,[4] $g(\mathbf{u}, \mathbf{v}) = \mathbf{u} \cdot \mathbf{v}/\|\mathbf{u}\|\|\mathbf{v}\|$ is the cosine similarity function, and $\mu(\cdot)$ is the matching function defined as follows,

$$\mu(\mathbf{x}) = \begin{cases} \mathbf{h}_i & \text{if } \mathbf{x} \text{ is equal to } \mathbf{c}_i. \\ \\ \mathbf{c}_i & \text{if } \mathbf{x} \text{ is equal to } \mathbf{h}_i. \end{cases}$$

Lastly, we sum all L_m^{base} divided by $2b$, and add a regularizer $L^{reg} = \|\text{BERT}_F - \text{BERT}_T\|_2^2$ to prevent BERT_T from being too distant from BERT_F.[5] As a result,

[3] We can also potentially make use of another sampler functions to inject our bias or prior knowledge on target tasks.

[4] We employ a two-layered MLP whose hidden size is 4096. Each linear layer in the MLP is followed by a GELU function.

[5] To be specific, L^{reg} is the square of the L2 norm of the difference between BERT_F and BERT_T. As shown in Figure 4.2, we also freeze the 0th layer of BERT_T for stable learning.

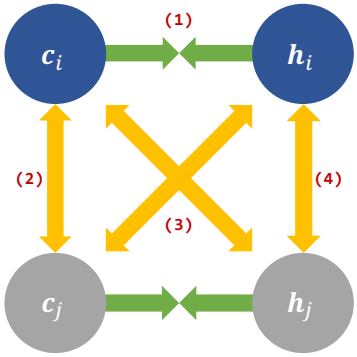

Figure 4.3 Four factors of the original NT-Xent loss. Green and yellow arrows represent the force of attraction and repulsion, respectively.

the final loss L^{base} is:

$$L^{base} = \frac{1}{2b} \sum_{m=1}^{2b} L_m^{base} + \lambda \cdot L^{reg},$$

where the coefficient λ is a hyperparameter.

To summarize, our method refines BERT so that the sentence embedding $\mathbf{c}_i$ has a higher similarity with $\mathbf{h}_i$, which is another representation for the sentence s_i, in the subspace projected by f while being relatively dissimilar with $\mathbf{c}_{j,j\neq i}$ and $\mathbf{h}_{j,j\neq i}$. After training is completed, we remove all the components except BERT_T and simply use $\mathbf{c}_i$ as the final sentence representation.

4.3.2 Learning Objective Optimization

In Section 4.3.1, we relied on a simple variation of the general NT-Xent loss, which is composed of four factors. Given sentence s_i and s_j without loss of generality, the factors are as follows (Figure 4.3):

(1) $\mathbf{c}_i \rightarrow\leftarrow \mathbf{h}_i$ (or $\mathbf{c}_j \rightarrow\leftarrow \mathbf{h}_j$): The main component that mirrors our core motivation that a BERT sentence vector ($\mathbf{c}_i$) should be consistent with intermediate views ($\mathbf{h}_i$) from BERT.

(2) $\mathbf{c}_i \longleftrightarrow \mathbf{c}_j$: A factor that forces sentence embeddings ($\mathbf{c}_i$, $\mathbf{c}_j$) to be distant from each other.

(3) $\mathbf{c}_i \longleftrightarrow \mathbf{h}_j$ (or $\mathbf{c}_j \longleftrightarrow \mathbf{h}_i$): An element that makes $\mathbf{c}_i$ being inconsistent with views for other sentences ($\mathbf{h}_j$).

(4) $\mathbf{h}_i \longleftrightarrow \mathbf{h}_j$: A factor that causes a discrepancy between views of different sentences ($\mathbf{h}_i$, $\mathbf{h}_j$).

Even though all the four factors play a certain role, some components may be useless or even cause a negative influence on our goal. For instance, Chen and He (2020) have recently reported that in image representation learning, only (1) is vital while others are nonessential. Likewise, we customize the training loss with three major modifications so that it can be more well-suited for our purpose.

First, as our aim is to improve $\mathbf{c}_i$ with the aid of $\mathbf{h}_i$, we re-define our loss focusing more on $\mathbf{c}_i$ rather than considering $\mathbf{c}_i$ and $\mathbf{h}_i$ as equivalent entities:

$$L_i^{opt1} = -\log\left(\phi(\mathbf{c}_i, \mathbf{h}_i)/\hat{Z}\right),$$
$$\text{where } \hat{Z} = \textstyle\sum_{j=1,j\neq i}^{b} \phi(\mathbf{c}_i, \mathbf{c}_j) + \sum_{j=1}^{b} \phi(\mathbf{c}_i, \mathbf{h}_j).$$

In other words, $\mathbf{h}_i$ only functions as points that $\mathbf{c}_i$ is encouraged to be close to or away from, and is not deemed as targets to be optimized. This revision naturally results in removing (4). Furthermore, we discover that (2) is also insignificant for improving performance, and thus derive L_i^{opt2}:

$$L_i^{opt2} = -\log(\phi(\mathbf{c}_i, \mathbf{h}_i)/\textstyle\sum_{j=1}^{b} \phi(\mathbf{c}_i, \mathbf{h}_j)).$$

Lastly, we diversify signals from (1) and (3) by allowing multiple views $\{\mathbf{h}_{i,k}\}$ to guide $\mathbf{c}_i$:

$$L_{i,k}^{opt3} = -\log \frac{\phi(\mathbf{c}_i, \mathbf{h}_{i,k})}{\phi(\mathbf{c}_i, \mathbf{h}_{i,k}) + \sum_{m=1,m\neq i}^{b} \sum_{n=0}^{l} \phi(\mathbf{c}_i, \mathbf{h}_{m,n})}.$$

Hyperparameters	Values
Random seed	1, 2, 3, 4, 1234, 2345, 3456, 7890
Evaluation step	50
Epoch	1
Batch size (b)	16
Optimizer	AdamW (β_1, β_2=(0.9, 0.9))
Learning rate	0.00005
Early stopping endurance	10
τ	0.01
λ	0.1

Table 4.1 Hyperparameters for experiments.

We expect with this refinement that the learning objective can provide more precise and fruitful training signals by considering additional (and freely available) samples being provided with. The final form of our optimized loss is:

$$L^{opt} = \frac{1}{b(l+1)} \sum_{i=1}^{b} \sum_{k=0}^{l} L_{i,k}^{opt3} + \lambda \cdot L^{reg}.$$

In Section 4.5.1, we show the decisions made in this section contribute to improvements in performance.

4.4 Experiments

4.4.1 General Configurations

In terms of pre-trained encoders, we leverage BERT (Devlin et al., 2019) for English datasets and MBERT, which is a multilingual variant of BERT, for multilingual datasets. We also employ RoBERTa (Liu et al., 2019b) and SBERT (Reimers and Gurevych, 2019) in some cases to evaluate the generalizability of tested methods. We use the suffixes '-base' and '-large' to distinguish small and large models. Every trainable model's performance is reported as the average of 8 separate runs to reduce randomness. Hyperparameters are optimized on the STS-B validation set using BERT-base and utilized across different models. See Table 4.1 for details. Our implementation is based on the HuggingFace's

`Transformers` (Wolf et al., 2019) and `SBERT` (Reimers and Gurevych, 2019) libraries.

4.4.2 Semantic Textual Similarity Tasks

We first evaluate our method and baselines on Semantic Textual Similarity (STS) tasks. Given two sentences, we derive their similarity score by computing the cosine similarity of their embeddings.

Datasets and Metrics. Following the literature, we evaluate models on 7 datasets in total, that is, STS-B (Cer et al., 2017), SICK-R (Marelli et al., 2014), and STS12-16 (Agirre et al., 2012, 2013, 2014, 2015, 2016). These datasets contain pairs of two sentences, whose similarity scores are labeled from 0 to 5. The relevance between gold annotations and the scores predicted by sentence vectors is measured in Spearman correlation ($\times$ 100).

Baselines and Model Specification. We first prepare two non-BERT approaches as baselines, i.e., Glove (Pennington et al., 2014) mean embeddings and Universal Sentence Encoder (USE; Cer et al. (2018)). In addition, various methods for BERT sentence embeddings that do not require supervision are also introduced as baselines:

- **CLS** token embedding: It regards the `[CLS]` vector from the last layer of BERT as a sentence representation.
- **Mean** pooling: This method conducts mean pooling on the last layer of BERT and use the output as a sentence embedding.
- **WK** pooling: This follows the method of Wang and Kuo (2020), which exploits QR decomposition and extra techniques to derive meaningful sentence vectors from BERT.

- **Flow**: This is *BERT-flow* proposed by Li et al. (2020), which is a flow-based model that maps the vectors made by taking mean pooling on the last two layers of BERT to the Gaussian space. We utilize this model in a restrictive manner, as we find it difficult to exactly reproduce the model's result with its official code.

- **Contrastive (BT)**: Following Fang and Xie (2020), we revise BERT with contrastive learning. This baseline is identical with our **Contrastive (SG)** model, except that it utilizes back-translation to generate positive samples. To be specific, English sentences in the training set are traslated into German sentences using the WMT'19 English-German translator provided by Ng et al. (2019), and then the translated German sentences are back-translated into English with the aid of the WMT'19 German-English model also offered by Ng et al. (2019). We utilize beam search during decoding with the beam size 100, which is relatively large, since we want the generated sentences to be more diverse while grammatically correct at the same time. Note that the contrastive (BT) model is trained with the NT-Xent loss (Chen et al., 2020), unlike CERT (Fang and Xie, 2020) which leverages the MoCo training objective (He et al., 2020).

We make use of plain sentences from STS-B to fine-tune BERT using our approach, identical with Flow. To be specific, for training, Li et al. (2020) utilize the concatenation of the STS-B training, validation, and test set (*without* gold annotations), and we follow the same setting for a fair comparison. We name the BERT instances trained with our self-guided method as **Contrastive (SG)** and **Contrastive (SG-OPT)**, which utilize L^{base} and L^{opt} in Section 4.3 respectively.

Models	Pooling	STS-B	SICK-R	STS12	STS13	STS14	STS15	STS16	Avg.
Non-BERT Baselines									
GloVe†	Mean	58.02	53.76	55.14	70.66	59.73	68.25	63.66	61.32
USE†	-	74.92	76.69	64.49	67.80	64.61	76.83	73.18	71.22
BERT-base									
+ No tuning	CLS	20.30	42.42	21.54	32.11	21.28	37.89	44.24	31.40
+ No tuning	Mean	47.29	58.22	30.87	59.89	47.73	60.29	63.73	52.57
+ No tuning	WK	16.07	41.54	16.01	21.80	15.96	33.59	34.07	25.58
+ Flow	Mean-2	$71.35_{\pm 0.27}$	$64.95_{\pm 0.16}$	$64.32_{\pm 0.17}$	$69.72_{\pm 0.25}$	$63.67_{\pm 0.06}$	$77.77_{\pm 0.15}$	$69.59_{\pm 0.28}$	$68.77_{\pm 0.07}$
+ Contrastive (BT)	CLS	$63.27_{\pm 1.48}$	$66.91_{\pm 1.29}$	$54.26_{\pm 1.84}$	$64.03_{\pm 2.35}$	$54.28_{\pm 1.87}$	$68.19_{\pm 0.95}$	$67.50_{\pm 0.96}$	$62.63_{\pm 1.28}$
+ **Contrastive (SG)**	CLS	$75.08_{\pm 0.73}$	$\mathbf{68.19}_{\pm 0.36}$	$63.60_{\pm 0.98}$	$76.48_{\pm 0.69}$	$67.57_{\pm 0.57}$	$79.42_{\pm 0.49}$	$74.85_{\pm 0.54}$	$72.17_{\pm 0.44}$
+ **Contrastive (SG-OPT)**	CLS	$\mathbf{77.23}_{\pm 0.43}$	$68.16_{\pm 0.50}$	$\mathbf{66.84}_{\pm 0.73}$	$\mathbf{80.13}_{\pm 0.51}$	$\mathbf{71.23}_{\pm 0.40}$	$\mathbf{81.56}_{\pm 0.28}$	$\mathbf{77.17}_{\pm 0.22}$	$\mathbf{74.62}_{\pm 0.25}$
BERT-large									
+ No tuning	CLS	26.75	43.44	27.44	30.76	22.59	29.98	42.74	31.96
+ No tuning	Mean	47.00	53.85	27.67	55.79	44.49	51.67	61.88	48.91
+ No tuning	WK	35.75	38.39	12.65	26.41	23.74	29.34	34.42	28.67
+ Flow	Mean-2	$72.72_{\pm 0.36}$	$63.77_{\pm 0.18}$	$62.82_{\pm 0.17}$	$71.24_{\pm 0.22}$	$65.39_{\pm 0.15}$	$78.98_{\pm 0.21}$	$73.23_{\pm 0.24}$	$70.07_{\pm 0.81}$
+ Contrastive (BT)	CLS	$63.84_{\pm 1.05}$	$66.53_{\pm 2.62}$	$52.04_{\pm 1.75}$	$62.59_{\pm 1.84}$	$54.25_{\pm 1.45}$	$71.07_{\pm 1.11}$	$66.71_{\pm 1.08}$	$62.43_{\pm 1.07}$
+ **Contrastive (SG)**	CLS	$75.22_{\pm 0.57}$	$69.63_{\pm 0.95}$	$64.37_{\pm 0.72}$	$77.59_{\pm 1.01}$	$68.27_{\pm 0.40}$	$80.08_{\pm 0.28}$	$74.53_{\pm 0.43}$	$72.81_{\pm 0.31}$
+ **Contrastive (SG-OPT)**	CLS	$\mathbf{76.16}_{\pm 0.42}$	$\mathbf{70.20}_{\pm 0.65}$	$\mathbf{67.02}_{\pm 0.72}$	$\mathbf{79.42}_{\pm 0.80}$	$\mathbf{70.38}_{\pm 0.65}$	$\mathbf{81.72}_{\pm 0.32}$	$\mathbf{76.35}_{\pm 0.22}$	$\mathbf{74.46}_{\pm 0.35}$
RoBERTa-base									
+ No tuning	CLS	45.41	61.89	16.67	45.57	30.36	55.08	56.98	44.57
+ No tuning	Mean	54.53	62.03	32.11	56.33	45.22	61.34	61.98	53.36
+ No tuning	WK	35.75	54.69	20.31	36.51	32.41	48.12	46.32	39.16
+ Contrastive (BT)	CLS	$\mathbf{79.93}_{\pm 1.08}$	$\mathbf{71.97}_{\pm 1.00}$	$62.34_{\pm 2.41}$	$78.60_{\pm 1.74}$	$68.65_{\pm 1.48}$	$79.31_{\pm 0.65}$	$77.49_{\pm 1.29}$	$\mathbf{74.04}_{\pm 1.16}$
+ **Contrastive (SG)**	CLS	$78.38_{\pm 0.43}$	$69.74_{\pm 1.00}$	$\mathbf{62.85}_{\pm 0.88}$	$78.37_{\pm 1.55}$	$68.28_{\pm 0.89}$	$\mathbf{80.42}_{\pm 0.65}$	$\mathbf{77.69}_{\pm 0.76}$	$73.67_{\pm 0.62}$
+ **Contrastive (SG-OPT)**	CLS	$77.60_{\pm 0.30}$	$68.42_{\pm 0.71}$	$62.57_{\pm 1.12}$	$\mathbf{78.96}_{\pm 0.67}$	$\mathbf{69.24}_{\pm 0.44}$	$79.99_{\pm 0.44}$	$77.17_{\pm 0.24}$	$73.42_{\pm 0.31}$

RoBERTa-large									
+ No tuning	CLS	12.52	40.63	19.25	22.97	14.93	33.41	38.01	25.96
+ No tuning	Mean	47.07	58.38	33.63	57.22	45.67	63.00	61.18	52.31
+ No tuning	WK	30.29	28.25	23.17	30.92	23.36	40.07	43.32	31.34
+ Contrastive (BT)	CLS	77.05$_{\pm 1.22}$	67.83$_{\pm 1.34}$	57.60$_{\pm 3.57}$	72.14$_{\pm 1.16}$	62.25$_{\pm 2.10}$	71.49$_{\pm 3.24}$	71.75$_{\pm 1.73}$	68.59$_{\pm 1.53}$
+ **Contrastive (SG)**	CLS	76.15$_{\pm 0.54}$	66.07$_{\pm 0.82}$	**64.77**$_{\pm 2.52}$	71.96$_{\pm 1.53}$	64.54$_{\pm 1.04}$	78.06$_{\pm 0.52}$	75.14$_{\pm 0.94}$	70.95$_{\pm 1.13}$
+ **Contrastive (SG-OPT)**	CLS	**78.14**$_{\pm 0.72}$	**67.97**$_{\pm 1.09}$	64.29$_{\pm 1.54}$	**76.36**$_{\pm 1.47}$	**68.48**$_{\pm 1.58}$	**80.10**$_{\pm 1.05}$	**76.60**$_{\pm 0.98}$	**73.13**$_{\pm 1.20}$
SBERT-base									
+ No tuning	CLS	73.66	69.71	70.15	71.17	68.89	75.53	70.16	71.32
+ No tuning	Mean	76.98	72.91	70.97	76.53	73.19	79.09	74.30	74.85
+ No tuning	WK	78.38	74.31	69.75	76.92	72.32	81.17	76.25	75.59
+ Flow‡	Mean-2	81.03	74.97	68.95	78.48	**77.62**	81.95	78.94	77.42
+ Contrastive (BT)	CLS	74.67$_{\pm 0.30}$	70.31$_{\pm 0.45}$	71.19$_{\pm 0.37}$	72.41$_{\pm 0.60}$	69.90$_{\pm 0.43}$	77.16$_{\pm 0.48}$	71.63$_{\pm 0.55}$	72.47$_{\pm 0.37}$
+ **Contrastive (SG)**	CLS	81.05$_{\pm 0.34}$	75.78$_{\pm 0.55}$	73.76$_{\pm 0.76}$	80.08$_{\pm 0.45}$	75.58$_{\pm 0.57}$	83.52$_{\pm 0.43}$	79.10$_{\pm 0.51}$	78.41$_{\pm 0.33}$
+ **Contrastive (SG-OPT)**	CLS	**81.46**$_{\pm 0.27}$	**76.64** $_{\pm 0.42}$	**75.16**$_{\pm 0.56}$	**81.27**$_{\pm 0.37}$	76.31$_{\pm 0.38}$	**84.71**$_{\pm 0.26}$	**80.33**$_{\pm 0.19}$	**79.41**$_{\pm 0.17}$
SBERT-large									
+ No tuning	CLS	76.01	70.99	69.05	71.34	69.50	76.66	70.08	71.95
+ No tuning	Mean	79.19	73.75	72.27	78.46	74.90	80.99	76.25	76.54
+ No tuning	WK	61.87	67.06	49.95	53.02	46.55	62.47	60.32	57.32
+ Flow‡	Mean-2	81.18	74.52	70.19	80.27	**78.85**	82.97	**80.57**	78.36
+ Contrastive (BT)	CLS	76.71$_{\pm 1.22}$	71.56$_{\pm 1.34}$	69.95$_{\pm 3.57}$	72.66$_{\pm 1.16}$	70.38$_{\pm 2.10}$	77.80$_{\pm 3.24}$	71.41$_{\pm 1.73}$	72.92$_{\pm 1.53}$
+ **Contrastive (SG)**	CLS	**82.35**$_{\pm 0.15}$	**76.44**$_{\pm 0.41}$	**74.84**$_{\pm 0.57}$	**82.89**$_{\pm 0.41}$	77.27$_{\pm 0.35}$	84.44$_{\pm 0.23}$	79.54$_{\pm 0.49}$	79.68$_{\pm 0.37}$
+ **Contrastive (SG-OPT)**	CLS	82.05$_{\pm 0.39}$	**76.44**$_{\pm 0.29}$	74.58$_{\pm 0.59}$	**83.79**$_{\pm 0.14}$	76.98$_{\pm 0.19}$	**84.57**$_{\pm 0.27}$	79.87$_{\pm 0.42}$	**79.76**$_{\pm 0.33}$

Table 4.2 Experimental results on STS tasks. Results for trained models are averaged over 8 runs (±: the standard deviation). The best figure in each (model-wise) part is in **bold** and the best in each column is underlined. Our method with self-guidance (SG, SG-OPT) generally outperforms competitive baselines. We borrow scores from previous work if we could not reproduce them. †: from Reimers and Gurevych (2019). ‡: from Li et al. (2020).

Models	Spanish
Baseline (Agirre et al., 2014)	
UMCC-DLSI-run2 (Rank #1)	80.69
MBERT	
+ CLS	12.60
+ Mean pooling	81.14
+ WK pooling	79.78
+ Contrastive (BT)	78.04
+ **Contrastive (SG)**	82.09
+ **Contrastive (SG-OPT)**	82.74

Table 4.3 Performance on the SemEval-2014 Task 10 Spanish task.

Results. We report the performance of different approaches on STS tasks in Table 4.2. We confirm the fact that our methods (SG and SG-OPT) mostly outperform other baselines in a variety of experimental settings. As reported in earlier studies, the naïve `[CLS]` embedding and mean pooling are turned out to be inferior to sophisticated methods. To our surprise, WK pooling's performance is even lower than that of mean pooling in most cases, and the only exception is when WK pooling is applied to SBERT-base. Flow shows its strength outperforming the simple strategies. Nevertheless, its performance is shown to be worse than that of our methods (although some exceptions exist in the case of SBERT-large). Note that contrastive learning becomes much more competitive when it is combined with our self-guidance algorithm rather than back-translation, except for the case of RoBERTa-base. It is also worth mentioning that the optimized version of our method (SG-OPT) generally shows better performance than the basic one (SG), proving the efficacy of learning objective optimization (Section 4.3.2). To conclude, we demonstrate that our self-guided contrastive learning is effective in improving the quality of BERT sentence embeddings when tested on STS tasks.

Models	Arabic (Track 1)	Spanish (Track 3)	English (Track 5)
Baselines			
Cosine baseline (Cer et al., 2017)	60.45	71.17	72.78
ENCU (Rank #1, Tian et al. (2017))	74.40	85.59	85.18
MBERT			
+ CLS	30.57	29.38	24.97
+ Mean pooling	51.09	54.56	54.86
+ WK pooling	50.38	55.87	54.87
+ Contrastive (BT)	54.24	68.16	73.89
+ Contrastive (SG)	57.09	78.93	78.24
+ Contrastive (SG-OPT)	58.52	80.19	78.03

Table 4.4 Results on SemEval-2017 Task 1: Track 1 (Arabic), Track 3 (Spanish), and Track 5 (English).

4.4.3 Multilingual STS Tasks

We expand our experiments to multilingual settings by utilizing MBERT and cross-lingual zero-shot transfer. Specifically, we refine MBERT using only English data and test it on datasets written in other languages. As in Section 4.4.2, we use the English STS-B for training. We consider two datasets for evaluation: (1) SemEval-2014 Task 10 (Spanish; Agirre et al. (2014)) and (2) SemEval-2017 Task 1 (Arabic, Spanish, and English; Cer et al. (2017)). Performance is measured in Pearson correlation ($\times$ 100) for a fair comparison with previous work.

From Table 4.3, we see that MBERT with mean pooling already outperforms the best system (at the time of the competition was held) on SemEval-2014 and that our method further boosts the model's performance. In contrast, in the case of SemEval-2017 (Table 4.4), MBERT with mean pooling even fails to beat the strong Cosine baseline.[6] However, MBERT becomes capable of outperforming (in English/Spanish) or being comparable with (Arabic) the baseline by adopting our algorithm. We observe that while cross-lingual transfer using MBERT

[6]The Cosine baseline computes its score as the cosine similarity of binary sentence vectors with each dimension representing whether an individual word appears in a sentence.

Models	MR	CR	SUBJ	MPQA	SST2	TREC	MRPC	Avg.
BERT-base								
+ Mean	81.46	86.71	95.37	87.90	85.83	90.30	73.36	85.85
+ WK	80.64	85.53	95.27	88.63	85.03	**94.03**	71.71	85.83
+ **SG-OPT**	**82.47**	**87.42**	**95.40**	**88.92**	**86.20**	91.60	**74.21**	**86.60**
BERT-large								
+ Mean	84.38	89.01	95.60	86.69	89.20	90.90	72.79	86.94
+ WK	82.68	87.92	95.32	**87.25**	87.81	91.18	70.13	86.04
+ **SG-OPT**	**86.03**	**90.18**	**95.82**	87.08	**90.73**	**94.65**	**73.31**	**88.26**
SBERT-base								
+ Mean	82.80	89.03	94.07	89.79	88.08	86.93	75.11	86.54
+ WK	82.96	89.33	**95.13**	**90.56**	88.10	**91.98**	**76.66**	**87.82**
+ **SG-OPT**	**83.34**	**89.45**	94.68	89.78	**88.57**	87.30	75.26	86.91

Table 4.5 Experimental results on SentEval.

looks promising for the languages analogous to English (e.g., Spanish), its effectiveness may shrink on distant languages (e.g., Arabic). Compared against the best system which is trained on task-specific data, MBERT shows reasonable performance considering that it is never exposed to any labeled STS datasets. In summary, we demonstrate that MBERT fine-tuned with our method has a potential to be used as a simple but effective tool for multilingual (especially European) STS tasks.

4.4.4 SentEval Benchmark

We also evaluate BERT sentence vectors using the SentEval (Conneau and Kiela, 2018) toolkit. Given sentence embeddings, SentEval trains linear classifiers on top of them and estimates the quality of the vectors via their performance (accuracy) on downstream tasks. Among available tasks, we employ 7: MR, CR, SUBJ, MPQA, SST2, TREC, MRPC.[7]

We specify some minor modifications applied on our contrastive method (SG-OPT). First, we use the portion of the concatenation of SNLI (Bowman

[7]Refer to Conneau and Kiela (2018) for each task's spec.

et al., 2015) and MNLI (Williams et al., 2018b) datasets as the training data instead of STS-B. Second, we do not leverage the first several layers of PLMs when making positive samples, similar to Wang and Kuo (2020), and utilize mean pooling instead of max pooling.

In Table 4.5, we compare SG-OPT with two baselines. We find that our method is helpful over usual mean pooling in improving the performance of BERT-like models on SentEval. SG-OPT also outperforms WK pooling on BERT-base/large while being comparable on SBERT-base. From the results, we conjecture that self-guided contrastive learning and SBERT training suggest a similar inductive bias in a sense, as the benefit we earn by revising SBERT with our method is relatively lower than the gain we obtain when fine-tuning BERT. Meanwhile, it seems that WK pooling provides an orthogonal contribution that is effective in the focused case, i.e., SBERT-base.

4.5 Analysis

We here further investigate the working mechanism of our method with supplementary experiments. All the experiments conducted in this section follow the configurations stipulated in Section 4.4.1 and 4.4.2.

4.5.1 Ablation Study

We conduct an ablation study to justify the decisions made in optimizing our algorithm. To this end, we evaluate each possible variant on the test sets of STS tasks. From Table 4.6, we confirm that all our modifications to the NT-Xent loss contribute to improvements in performance. Moreover, we show that correct choices for hyperparameters are important for achieving the optimal performance, and that the projection head (f) plays a significant role as in Chen et al. (2020).

Models	STS Tasks (Avg.)
BERT-base	
+ **SG-OPT** (L^{opt3})	74.62
+ L^{opt2}	73.14 (-1.48)
+ L^{opt1}	72.61 (-2.01)
+ **SG** (L^{base})	72.17 (-2.45)
BERT-base + **SG-OPT** ($\tau = 0.01$, $\lambda = 0.1$)	74.62
+ $\tau = 0.1$	70.39 (-4.23)
+ $\tau = 0.001$	74.16 (-0.46)
+ $\lambda = 0.0$	73.76 (-0.86)
+ $\lambda = 1.0$	73.18 (-1.44)
− Projection head (f)	72.78 (-1.84)

Table 4.6 Ablation study.

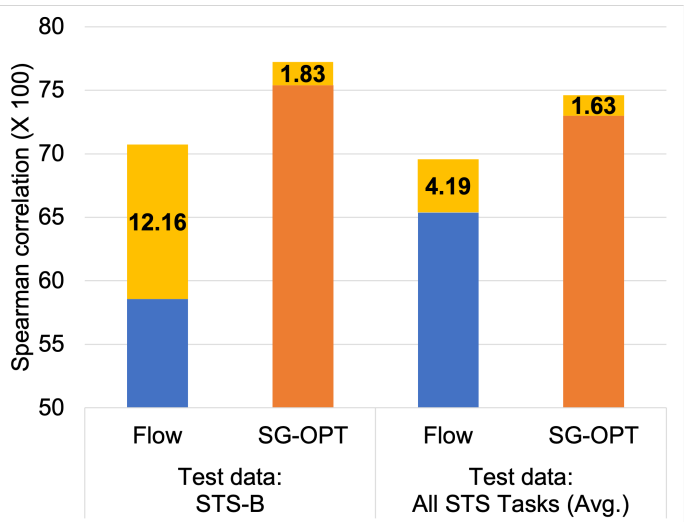

Figure 4.4 Domain robustness study. The yellow bars indicate the performance gaps each method has according to the fact that it is trained with whether in-domain (STS-B) or out-of-domain (NLI) data. Our method (SG-OPT) clearly shows its relative robustness compared to Flow.

4.5.2 Robustness to Domain Shifts

Although our method in principle can accept any sentences in training, its performance might be varied with the training data it employs (especially de-

Models	Elapsed Time	
	Training (sec.)	**Inference (sec.)**
BERT-base		
+ Mean pooling	-	13.94
+ WK pooling	-	197.03 (≈ 3.3 min.)
+ Flow	155.37 (≈ 2.6 min.)	28.49
+ **Contrastive (SG-OPT)**	455.02 (≈ 7.5 min.)	10.51

Table 4.7 Computational efficiency tested on STS-B.

pending on whether the training and test data share the same domain). To explore this issue, we apply SG-OPT on BERT-base by leveraging the mix of NLI datasets (Bowman et al., 2015; Williams et al., 2018b) instead of STS-B, and observe the difference. From Figure 4.4, we confirm the fact that no matter which test set is utilized (STS-B or all the seven STS tasks), our method clearly outperforms Flow in every case, showing its relative robustness to domain shifts. SG-OPT only loses 1.83 (on the STS-B test set) and 1.63 (on average when applied to all the STS tasks) points respectively when trained with NLI rather than STS-B, while Flow suffers from the considerable losses of 12.18 and 4.19 for each case. Note, however, that follow-up experiments in more diverse conditions might be desired as future work, as the NLI dataset inherently shares some similarities with STS tasks.

4.5.3 Computational Efficiency

In this part, we compare the computational efficiency of our method to that of other baselines. For each algorithm, we measure the time elapsed during training (if required) and inference when tested on STS-B. All methods are run on the same machine (an Intel Xeon CPU E5-2620 v4 @ 2.10GHz and a Titan Xp GPU) using batch size 16. The experimental results specified in Table 4.7 show that although our method demands a moderate amount of time (< 8 min.)

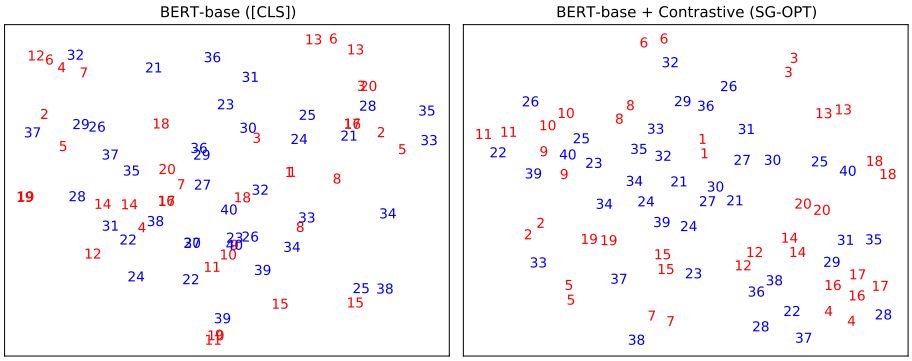

Figure 4.5 Sentence representation visualization. (Left) Embeddings from the original BERT. (Right) Embeddings from the BERT instance fine-tuned with SG-OPT. Red numbers correspond to positive sentence pairs and blue to negative pairs.

for training, it is the most efficient at inference, since our method is free from any post-processing such as pooling once training is completed.

4.5.4 Representation Visualization

We visualize a few variants of BERT sentence representations to grasp an intuition on why our method is effective in improving performance. Specifically, we sample 20 positive pairs (red, whose similarity scores are 5) and 20 negative pairs (blue, whose scores are 0) from the STS-B validation set. Then we compute their vectors and draw them on the 2D space with the aid of t-SNE. In Figure 4.5, we confirm that our SG-OPT encourages BERT sentence embeddings to be more well-aligned with their positive pairs while still being relatively far from their negative pairs.

4.6 Limitations and Future Work

In this section, we discuss a few weaknesses of our method in its current form and look into some possible avenues for future work.

Models / Tasks	STS-B	SICK-R	STS12	STS13
BERT-base				
+ Contrastive (BT)	$63.27_{\pm 1.48}$	$66.91_{\pm 1.29}$	$54.26_{\pm 1.84}$	$64.03_{\pm 2.35}$
+ Contrastive (SG-OPT)	$77.23_{\pm 0.43}$	$68.16_{\pm 0.50}$	$66.84_{\pm 0.73}$	$\mathbf{80.13}_{\pm 0.51}$
+ **Contrastive (BT + SG-OPT)**	$\mathbf{77.99}_{\pm 0.23}$	$\mathbf{68.75}_{\pm 0.79}$	$\mathbf{68.49}_{\pm 0.38}$	$80.00_{\pm 0.78}$
Models / Tasks	**STS14**	**STS15**	**STS16**	**Avg.**
BERT-base				
+ Contrastive (BT)	$54.28_{\pm 1.87}$	$68.19_{\pm 0.95}$	$67.50_{\pm 0.96}$	$62.63_{\pm 1.28}$
+ Contrastive (SG-OPT)	$71.23_{\pm 0.40}$	$81.56_{\pm 0.28}$	$77.17_{\pm 0.22}$	$74.62_{\pm 0.25}$
+ **Contrastive (BT + SG-OPT)**	$\mathbf{71.34}_{\pm 0.40}$	$\mathbf{81.71}_{\pm 0.29}$	$\mathbf{77.43}_{\pm 0.46}$	$\mathbf{75.10}_{\pm 0.15}$

Table 4.8 Ensemble of the techniques for contrastive learning: back-translation (BT) and self-guidance (SG-OPT).

First, while defining the proposed method in Section 4.3, we have made decisions on some parts without much consideration about their optimality, prioritizing simplicity instead. For instance, although we proposed utilizing all the intermediate layers of BERT and max pooling in a normal setting (indeed, it worked pretty well for most cases), a specific subset of the layers or another pooling method might bring better performance in a particular environment, as we observed in Section 4.4.4 that we could achieve higher numbers by employing mean pooling and excluding lower layers in the case of SentEval. Therefore, in future work, it is encouraged to develop a systematic way of making more optimized design choices in specifying our method by considering the characteristics of target tasks.

Second, we expect that the effectiveness of contrastive learning in revising BERT can be improved further by properly combining different techniques developed for it. As an initial attempt towards this direction, we conduct an extra experiment where we test the ensemble of back-translation and our self-guidance algorithm by inserting the original sentence into BERT_T and its back-translation into BERT_F when running our framework. In Table 4.8, we show that the fusion of the two techniques generally results in better performance,

shedding some light on our future research direction.

On the other hand, we have developed our method based on the intuition that fully exploiting different layers in BERT is helpful for building fruitful sentence representations in the perspective that it facilitates considering various aspects of language including syntax. However, we have not yet verified whether there is a direct correlation between the method's strong results and its sensitivity to linguistic information. It is therefore desirable to make more effort in probing the inner workings of pre-trained language models, which becomes the main topic of the next two chapters.

4.7 Summary

We have proposed a contrastive learning method with self-guidance for improving BERT sentence embeddings. Through extensive experiments, we have demonstrated that our method can enjoy the benefit of contrastive learning without relying on external procedures such as data augmentation or back-translation, succeeding in generating higher-quality sentence representations compared to competitive baselines. Furthermore, our method is efficient at inference because it does not require any post-processing once its training is completed, and is relatively robust to domain shifts.

Chapter 5

Syntactic Analysis of Sentence Representation Models

5.1 Introduction

"What I cannot create, I do not understand." — Richard Feynman

From this chapter, we open a new episode of our discussion where we regard syntax as an analytic tool that can be utilized to explore the working mechanism of neural models for sentence representations. In particular, we pay our attention to pre-trained Transformer (Vaswani et al., 2017) language models (PLMs), which are self-attention based models that have hundreds of millions of parameters and are trained with large-scale plain text corpora in a self-supervised fashion. Stimulated by the above saying of a famous American physicist Richard Feynman, we attempt to estimate the extent to which PLMs are aware of syntactic concepts by generating parse trees relying on the internal knowledge of the models. In detail, we introduce some techniques existing in the literature of grammar induction and unsupervised parsing, and apply them

to the intermediate hidden representations of PLMs to induce parse trees.

Grammar induction, which is closely related to unsupervised parsing and latent tree learning, allows one to associate syntactic trees, i.e., constituency and dependency trees, with sentences. As grammar induction essentially assumes no supervision from gold-standard syntactic trees, existing approaches for this task mainly rely on unsupervised objectives, such as language modeling (Shen et al., 2018b, 2019; Kim et al., 2019c,d) and cloze-style word prediction (Drozdov et al., 2019) to train their task-oriented models. On the other hand, there is a trend in the NLP community of leveraging pre-trained language models (PLMs), e.g., ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019), as a means of acquiring contextualized word and sentence representations. These representations have proven to be surprisingly effective, playing key roles in recent improvements in various models for NLP tasks.

In this chapter, inspired by the fact that the training objectives of both the approaches for grammar induction and for training PLMs are identical, namely, (masked) language modeling, we investigate whether PLMs can also be utilized for grammar induction/unsupervised parsing, especially *without* training. Specifically, we focus on extracting constituency parse trees from PLMs without fine-tuning or introducing another task-specific module, at least one of which is usually required in other cases where representations from PLMs are employed. This restriction provides us with some advantages: (i) it enables us to derive strong baselines for grammar induction with reduced time and space complexity, offering a chance to reexamine the current status of existing grammar induction methods, and (ii) it facilitates an analysis on how much and what kind of syntactic information each PLM contains in its intermediate representations and attention distributions in terms of phrase-structure grammar.

5.2 Related Work

Grammar induction is a task whose goal is to infer from sequential data grammars which generalize and are able to account for unseen data (Lari and Young (1990); Clark (2001); Klein and Manning (2002b, 2004), to name a few). Traditionally, this was done by learning explicit grammar rules (e.g., context free rewrite rules), though more recent methods employ neural networks to learn such rules implicitly, focusing more on the induced grammars' ability to generate or parse sequences as unsupervised parsers. We therefore use the terms grammar induction and unsupervised parsing interchangeably.

Shen et al. (2018b) proposed Parsing-Reading-Predict Network (PRPN) where the concept of *syntactic distance* is first introduced. They devised a neural model for language modeling where the model is encouraged to recognize syntactic structure. The authors also probed the possibility of inducing constituency trees without access to gold-standard trees by adopting an algorithm that recursively splits a sequence of words into two parts, the split point being determined according to correlated syntactic distances; the point having the biggest distance becomes the first target of division. Shen et al. (2019) presented a model called Ordered Neurons (ON), which is a revised version of LSTM (Long Short-Term Memory; Hochreiter and Schmidhuber (1997)) which reflects the hierarchical biases of natural language and can be used to compute syntactic distances. Shen et al. (2018c) trained a supervised parser relying on the concept of syntactic distance.

Other studies include Drozdov et al. (2019), who trained deep inside-outside recursive autoencoders (DIORA) to derive syntactic trees in an exhaustive way with the aid of the inside-outside algorithm, and Kim et al. (2019c) who proposed Compound Probabilistic Context-Free Grammars (compound PCFG),

showing that neural PCFG models are capable of producing promising unsupervised parsing results. Li et al. (2019) proved that an ensemble of unsupervised parsing models can be beneficial, while Shi et al. (2019) utilized additional training signals from pictures related with input text. Dyer et al. (2016) proposed Recurrent Neural Network Grammars (RNNG) for both language modeling and parsing, and Kim et al. (2019d) suggested an unsupervised variant of the RNNG. There also exists another line of research on task-specific latent tree learning (Yogatama et al., 2017; Choi et al., 2018; Havrylov et al., 2019; Maillard et al., 2019). The goal here is not to construct linguistically plausible trees, but to induce the trees fitted to improving target performance. Naturally, the induced performance-based trees need not resemble linguistically plausible trees, and some studies (Williams et al., 2018a; Nangia and Bowman, 2018) examined the apparent fact that performance-based and linguistically plausible trees bear little resemblance to one another.

Pre-trained language models (Peters et al. (2018); Devlin et al. (2019); Radford et al. (2019); Yang et al. (2019); Liu et al. (2019b), *inter alia*)—particularly those employing the Transformer architecture (Vaswani et al., 2017)—have proven to be helpful for diverse NLP downstream tasks. In spite of this, there is no vivid picture for explaining what particular factors contribute to their superior performance, even though some recent work has attempted to shed light on this question. In detail, one group of studies (Raganato and Tiedemann (2018); Clark et al. (2019); Ethayarajh (2019); Hao et al. (2019); Voita et al. (2019), *inter alia*) has focused on dissecting the intermediate representations and attention distributions of PLMs, while the another group of publications (Mareček and Rosa (2018); Goldberg (2019); Hewitt and Manning (2019); Liu et al. (2019a); Rosa and Mareček (2019), to name a few) delve into the question of the existence of syntactic knowledge in Transformer-based mod-

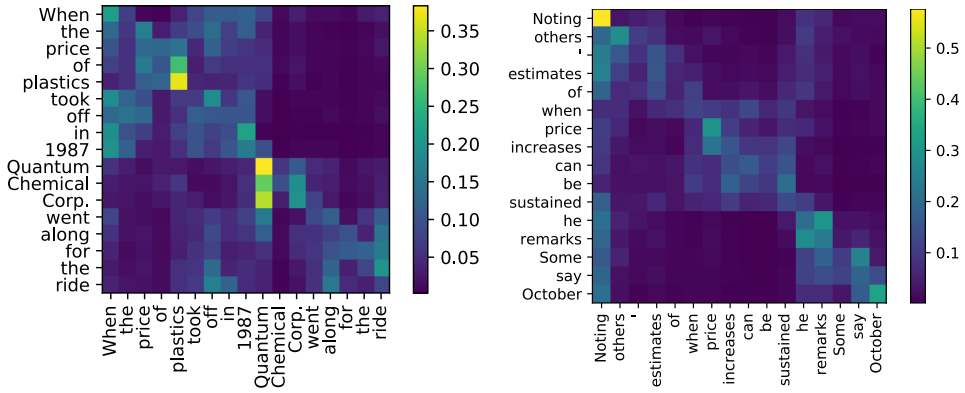

Figure 5.1 Self-attention heatmaps from two different PLMs. (Left) A heatmap for the average of attention distributions from the 7th layer of XLNet-base. (Right) A heatmap for the average of attention distributions from the 9th layer of BERT-base. We can spot the chunks of words on the two heatmaps that are correlated with the constituents of the input sentences, e.g., (Left) 'the price of plastics', 'took off in 1987', 'Quantum Chemical Corp.', (Right) 'when price increases can be sustained', and 'he remarks'.

els. Particularly, Mareček and Rosa (2019) proposed an algorithm for extracting constituency trees from Transformers trained for machine translation, which is similar to our approach.

5.3 Motivation

As having been pointed out in the literature, the multi-head self-attention mechanism (Vaswani et al., 2017) is a key component in Transformer-based language models, and it seems this mechanism empowers the models to capture certain semantic and syntactic information existing in natural language. Among a diverse set of knowledge they may capture, we concentrate on phrase-structure

grammar by seeking to extract constituency trees directly from their attention information and intermediate weights.

In preliminary experiments, where we manually visualized and investigated the intermediate representations and attention distributions of several PLMs given input, we found some evidence which suggests that the PLMs exhibit syntactic structure akin to constituency grammar in some degree. Specifically, we noticed some patterns which are often displayed in self-attention heatmaps as explicit horizontal lines, or groups of rectangles of various sizes. As an attention distribution of a word over other words corresponds to a row in a heatmap matrix, we can say that the appearance of these patterns indicates the existence of groups of words where the attention distributions of the words in the same group are relatively similar. Interestingly, we also discovered the fact that such groups of words are fairly correlated with the constituents of the input sentence, as shown in Figure 5.1.

Even though we have identified some patterns which match with the constituents of sentences, it is not enough to conclude that PLMs are aware of syntactic phrases as found in phrase-structure grammars. To demonstrate the claim, we attempt to obtain constituency trees in a zero-shot learning fashion, relying only on the knowledge from the PLMs. To this end, we suggest the following, inspired from our finding: two words in a sentence are syntactically *close* to each other (i.e., the two words belong to the same constituent) if their attention distributions over words in the sentence are also *close* to each other. Note that this implicitly presumes that each word is more likely to attend more on the words in the same constituent to enrich its representation in PLMs. Finally, we utilize the assumption to compute syntactic distances between each pair of adjacent words in a sentence, from which the corresponding constituency tree can be built.

Methodology	Unsupervised Parsing	CPE-PLM
Training data	In-domain data (e.g., raw text from PTB)	General corpora (e.g., Wikipedia)
Architecture	Task-oriented (e.g., RNNG, PCFG)	Transformer
Modeling	$p(S, T)$ (T is marginalized or implicitly modeled)	$p(S)$ (T is not considered in modeling)

Table 5.1 Comparison between typical unsupervised parsing and constituency parse extraction from pre-trained language models (CPE-PLM).

5.4 Method

5.4.1 CPE-PLM

In this chapter, we focus on a variant of unsupervised constituency parsing, which we call **C**onstituency **P**arse **E**xtraction from **P**re-trained **L**anguage **M**odels (**CPE-PLM**). We specify the characteristics of CPE-PLM in Table 5.1, comparing them with those of general unsupervised parsing methods.

Typical unsupervised parsers consist of task-oriented architectures (e.g., RNNG (Kim et al., 2019d) and PCFG (Kim et al., 2019c)) which are designed to model both a sentence S and the corresponding tree T (i.e., $p(S, T)$) and are trained with in-domain plain text.[1] On the other hand, CPE-PLM simply employs off-the-shelf Transformer PLMs, which only model the probability of a sentence $p(S)$, as their core component and do not require additional training—the PLMs are frozen and no trainable component is augmented on top of them, meaning *parameter-free*. Instead, CPE-PLM methods take advantage of implicit

[1] As it is mostly infeasible to directly model the tree T without supervision from gold annotations, unsupervised parsers usually make use of different approximation or marginalization techniques such as variational inference and sampling.

syntactic knowledge residing in PLMs to reconstruct parses, by computing syntactic distances (Shen et al., 2018b) between words in a sentence using features from the PLMs. We describe their algorithmic details in the following subsections. CPE-PLM's independence from training also makes it being distinct from syntactic probes for PLMs (Hewitt and Manning, 2019; Chi et al., 2020) which demand training probing modules to investigate the latent knowledge of PLMs.

5.4.2 Top-down CPE-PLM

We leverage the concept of syntactic distance proposed by Shen et al. (2018c,b) to draw constituency trees from raw sentences in an intuitive way. Formally, given a sequence of words in a sentence, $w_1, w_2, \ldots, w_n$, we compute $\mathbf{d} = [d_1, d_2, \ldots, d_{n-1}]$ where d_i corresponds to the syntactic distance between w_i and w_{i+1}. Each d_i is defined as follows:

$$d_i = f(g(w_i), g(w_{i+1})),$$

where $f(\cdot, \cdot)$ and $g(\cdot)$ are a distance measure function and representation extractor function, respectively. The function g converts each word into the corresponding vector representation, while f computes the syntactic distance between the two words given their representations. Once $\mathbf{d}$ is derived, it can be easily converted into the target constituency tree by Algorithm 1 following Shen et al. (2018c).[2] As Algorithm 1 recursively constructs a parse tree in a top-down manner, we name our method as **top-down CPE-PLM**.

Although previous studies attempted to explicitly train the functions f and

[2]Our parsing algorithm is an unbiased method in contrast to one (named as $\overline{\text{COO}}$ parser by Dyer et al. (2019)) employed in most previous studies (Shen et al., 2018b, 2019; Htut et al., 2018; Li et al., 2019; Shi et al., 2019). This choice enables us to investigate the exact extent to which PLMs contribute to their performance on unsupervised parsing, considering the fact revealed recently by Dyer et al. (2019) that the $\overline{\text{COO}}$ parser has potential issues that it prefers right-branching trees and does not cover all possible tree derivations. Furthermore, we can directly adjust the right-branching bias using our method in Section 5.4.5 if needed.

Algorithm 1 Syntactic Distance to Binary Constituency Tree

$S = [w_1, w_2, \ldots, w_n]$: Words in a sentence of length n.
$\mathbf{d} = [d_1, d_2, \ldots, d_{n-1}]$: A vector, each of whose element d_i is the syntactic distance between w_i and w_{i+1}.
function D2T(S, $\mathbf{d}$)
    if $\mathbf{d} = []$ **then**
        node $\leftarrow$ Leaf($S[0]$)
    else
        $i \leftarrow \arg\max_i(\mathbf{d})$
        child$_l$ $\leftarrow$ D2T($S_{\leq i}, \mathbf{d}_{<i}$)
        child$_r$ $\leftarrow$ D2T($S_{>i}, \mathbf{d}_{>i}$)
        node $\leftarrow$ Node(child$_l$, child$_r$)
    end if
    return node
end function

g with supervision (with access to gold-standard trees, Shen et al. (2018c)) or to obtain them as a by-product of training particular models that are carefully designed to recognize syntactic information (Shen et al., 2018b, 2019), we stick to simple distance metric functions for f and PLMs for g, forgoing any training process. In other words, we focus on investigating the possibility of PLMs possessing constituency information in a form that can be readily extracted with straightforward computations. If the trees induced by the syntactic distances derived from the PLMs are similar enough to gold-standard syntax trees, we can reasonably claim that the LMs resemble phrase-structure.

5.4.3 Pre-trained Language Models

We consider four types of recently proposed language models. These are: BERT (Devlin et al., 2019), GPT-2 (Radford et al., 2019), RoBERTa (Liu et al., 2019b), and XLNet (Yang et al., 2019). They all have in common that they are based on the Transformer architecture and have been proven to be effective in natural language understanding (Wang et al., 2019b) or generation. We handle two variants for each LM, varying in the number of layers, attention heads,

and hidden dimensions, resulting in eight different cases in total. In particular, each LM has two variants. (1) base: consists of l=12 layers, a=12 attention heads, and d=768 hidden dimensions, while (2) large: has l=24 layers, a=16 attention heads, and d=1024 hidden dimensions.[3] We deal with a wide range of PLMs, unlike previous work which has mostly analyzed a specific model, particularly BERT. For details about each LM, we refer readers to the respective original papers.

In terms of our formulation, each LM instance provides two categories of representation extractor functions, G^v and G^d. Specifically, G^v refers to a set of functions $\{g^v_j | j = 1, \ldots, l\}$, each of which simply outputs the intermediate hidden representation of a given word on the jth layer of the LM. Likewise, G^d is a set of functions $\{g^d_{(j,k)} | j = 1, \ldots, l, k = 1, \ldots, a+1\}$, each of which outputs the attention distribution of an input word by the kth attention head on the jth layer of the LM. Even though our main motivation comes from the self-attention mechanism, we also deal with the intermediate hidden representations present in the PLMs by introducing G^v, considering that the hidden representations serve as storage of collective information taken from the processing of the PLMs. Note that k ranges up to $a+1$, not a, implying that we consider the average of all attention distributions on the same layer in addition to the individual ones. This averaging function can be regarded as an ensemble of other functions in the layer which are specialized for different aspects of information, and we expect that this technique will provide a better option in some cases as reported in the previous work (Li et al., 2019).

One remaining issue is that all the PLMs we use regard each input sentence as a sequence of *subword* tokens, while our formulation assumes words cannot

[3]In case of GPT-2, 'GPT2' corresponds to the 'base' variant while 'GPT2-medium' to the 'large' one.

Function (f)	**Equation**
Functions for intermediate representations (F^v)	
$\mathrm{Cos}(\mathbf{r}, \mathbf{s})$	$\big(\mathbf{r}^\top \mathbf{s} / \big((\sum_{i=1}^{d} {r_i}^2)^{\frac{1}{2}} \cdot (\sum_{i=1}^{d} {s_i}^2)^{\frac{1}{2}}\big) + 1\big)/2$
$\mathrm{L1}(\mathbf{r}, \mathbf{s})$	$\sum_{i=1}^{d} \lvert r_i - s_i \rvert$
$\mathrm{L2}(\mathbf{r}, \mathbf{s})$	$(\sum_{i=1}^{d} (r_i - s_i)^2)^{\frac{1}{2}}$
Functions for attention distributions (F^d)	
$\mathrm{JSD}(P \Vert Q)$	$((D_{\mathrm{KL}}(P \Vert M) + D_{\mathrm{KL}}(Q \Vert M))/2)^{\frac{1}{2}}$
	where $M = (P + Q)/2$
	and $D_{\mathrm{KL}}(A \Vert B) = \sum_{w \in S} A(w) \log(A(w)/B(w))$
$\mathrm{HEL}(P, Q)$	$\frac{1}{\sqrt{2}} (\sum_{i=1}^{n} (\sqrt{p_i} - \sqrt{q_i})^2)^{\frac{1}{2}}$

Table 5.2 The definition of distance measure functions for computing syntactic distances between two adjacent words in a sentence. Note that $\mathbf{r} = g^v(w_i)$, $\mathbf{s} = g^v(w_{i+1})$, $P = g^d(w_i)$, and $Q = g^d(w_{i+1})$, respectively. d: hidden embedding size, n: the number of words (w) in a sentence (S).

be further divided into smaller tokens. To resolve this difference, we tested certain heuristics that guide how subword tokens for a complete word should be exploited to represent the word, and we have empirically found that the best result comes when each word is represented by an average of the representations of its subwords.[4] Therefore, we adopt the above heuristic for cases where a word is tokenized into more than two parts.

5.4.4 Distance Measure Functions

For the distance measure function f, we prepare three options (F^v) for G^v and two options (F^d) for G^d. Formally, $f \in F^v \cup F^d$, where $F^v = \{\mathrm{COS}, \mathrm{L1}, \mathrm{L2}\}$, $F^d = \{\mathrm{JSD}, \mathrm{HEL}\}$. COS, L1, L2, JSD, and HEL correspond to Cosine, L1, and L2, Jensen-Shannon, and Hellinger distance respectively. The functions in F^v are only compatible with the elements of G^v, and the same holds for F^d and

[4]We also tried other heuristics following previous work (Kitaev and Klein, 2018), e.g., using the first or last subword of a word as representative, but this led to no performance gains.

G^d. The exact definition of each function is listed in Table 5.2.

5.4.5 Injecting Bias into Syntactic Distances

One of the main advantages we obtain by leveraging syntactic distances to derive parse trees is that we can easily inject inductive biases into our framework by simply modifying the values of the syntactic distances. Hence, we investigate whether the extracted trees from our method can be further refined with the aid of additional biases. To this end, we introduce a well-known bias for English constituency trees—the *right-skewness* bias—in a simple linear form.[5] Namely, our intention is to modify the induced trees such that they are moderately right-skewed following the nature of gold-standard parse trees in English.

Formally, we compute $\hat{d}_i$ by appending the following linear bias term to every d_i:

$$\hat{d}_i = d_i + \lambda \cdot \mathrm{AVG}(\mathbf{d}) \times (1 - 1/(m-1) \times (i-1)),$$

where $\mathrm{AVG}(\cdot)$ outputs an average of all elements in a vector, λ is a hyperparameter, and i ranges from 1 to $m = n - 1$. We write $\hat{\mathbf{d}} = [\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_m]$ in place of $\mathbf{d}$ to signify the biased syntactic distances.

The main purpose of introducing such a bias is examining what changes are made to the resulting tree structures rather than boosting quantitative performance *per se*, though it is of note that it serves this purpose as well. We believe that this additional consideration is necessary based on two points. First, English is what is known as a head-initial language. That is, given a selector and argument, the selector has a strong tendency to appear on the left, e.g., 'eat food', or 'to Canada'. Head-initial languages therefore have an in-built preference for right-branching structures. By adjusting the bias injected into syntactic

[5]It is necessary to carefully design biases for other languages as they have their own properties.

distances derived from PLMs, we can figure out whether the LMs are capable of inducing the right-branching bias, which is one of the main properties of English syntax; if injecting the bias does not influence the performance of the LMs on unsupervised parsing, we can conjecture they are inherently capturing the bias to some extent. Second, as mentioned before, we have witnessed some previous work (Shen et al., 2018b, 2019; Htut et al., 2018; Li et al., 2019; Shi et al., 2019) where the right-skewness bias is implicitly exploited, although it could be regarded as not ideal. What we intend to focus on is the question about which benefits the bias provides for such parsing models, leading to overall performance improvements. In other words, we look for what the exact contribution of the bias is when it is injected into grammar induction models, by explicitly controlling the bias using our framework.

5.5 Experiments

5.5.1 General Configurations

Datasets. We conduct unsupervised constituency parsing on two datasets. The first dataset is WSJ Penn Treebank (PTB, Marcus et al. (1993)), in which human-annotated gold-standard trees are available. We use the standard split of the dataset—2-21 for training, 22 for validation, and 23 for test. The second one is MNLI (Williams et al., 2018b), which is originally designed to test natural language inference but often utilized as a means of evaluating parsers. It contains constituency trees produced by an external parser (Klein and Manning, 2003). We leverage the union of two different versions of the MNLI development set as test data following convention (Htut et al., 2018; Drozdov et al., 2019), and we call it the MNLI test set. Moreover, we randomly sample 40K sentences from the training set of the MNLI to utilize them as a validation set. To preprocess the datasets, we follow the setting of Kim et al. (2019c) with the minor

exceptions that words are not lower-cased and number characters are preserved instead of being substituted by a special character.

Implementation Details. For implementation, to compare PLMs in an unified manner, we resort to the HuggingFace's `Transformers` (Wolf et al., 2019) that supports all the models we consider. For each LM, we tune the best combination of f and g functions using the validation set. Then, we derive a set of $\mathbf{d}$ for sentences in the test set using the chosen functions, followed by the resulting constituency trees converted from each $\mathbf{d}$ by the tree construction algorithm in Section 5.4.2. In addition to the sentence-level F1 (S-F1) score, we report label recall scores for six main categories: SBAR, NP, VP, PP, ADJP, and ADVP. We also present the results of utilizing $\hat{\mathbf{d}}$ instead of $\mathbf{d}$, empirically setting the bias hyperparameter λ as 1.5. We do not fine-tune PLMs on domain-specific data, as we here focus on finding their universal characteristics.

We take four naïve baselines into account, random (averaged over 5 trials), balanced, left-branching, and right-branching binary trees. In addition, we present two more baselines which are identical to our models except that their g functions are based on a randomly initialized XLNet-base rather than pre-trained ones. To be concrete, We provide 'Random XLNet-base (F^v)' which applies the functions in F^v on random hidden representations and 'Random XLNet-base (F^d)' that utilizes the functions in F^d and random attention distributions, respectively. Considering the randomness of initialization and possible choices for f, the final score for each of the baselines is calculated as an average over 5 trials of each possible f, i.e., an average over 5×3 runs in case of F^v and 5×2 runs for F^d. These baselines enable us to estimate the exact advantage we obtain by pre-training LMs, effectively removing additional unexpected gains that may exist. Furthermore, we compare our parse trees against ones from

Models	f	L	A	S-F1	SBAR	NP	VP	PP	ADJP	ADVP
Baselines										
Random Trees	-	-	-	18.1	8%	23%	12%	18%	23%	28%
Balanced Trees	-	-	-	18.5	7%	27%	8%	18%	27%	25%
Left Branching Trees	-	-	-	8.7	5%	11%	0%	5%	2%	8%
Right Branching Trees	-	-	-	39.4	**68%**	24%	**71%**	42%	27%	38%
Random XLNet-base (F^v)	-	-	-	19.6	9%	26%	12%	20%	23%	24%
Random XLNet-base (F^d)	-	-	-	20.1	11%	25%	14%	19%	22%	26%
PLMs (λ=0)										
BERT-base	JSD	9	AVG	32.4	28%	42%	28%	31%	35%	63%
BERT-large	HEL	17	AVG	34.2	34%	43%	27%	39%	37%	57%
GPT2	JSD	9	1	37.1	32%	47%	27%	55%	27%	36%
GPT2-medium	JSD	10	13	39.4	41%	51%	21%	**67%**	33%	44%
RoBERTa-base	JSD	9	4	33.8	40%	38%	33%	43%	42%	57%
RoBERTa-large	JSD	14	5	34.1	29%	46%	30%	37%	28%	40%
XLNet-base	HEL	9	AVG	40.1	35%	56%	26%	38%	47%	68%
XLNet-large	L2	11	-	38.1	36%	51%	26%	41%	45%	**69%**
PLMs (λ=1.5)										
BERT-base	HEL	9	AVG	42.3	45%	46%	**49%**	43%	41%	65%
BERT-large	HEL	17	AVG	44.4	55%	48%	48%	52%	41%	62%
GPT2	JSD	9	1	41.3	43%	49%	38%	**58%**	27%	43%
GPT2-medium	HEL	2	1	42.3	54%	50%	39%	56%	24%	41%
RoBERTa-base	JSD	8	AVG	42.1	51%	44%	44%	55%	40%	66%
RoBERTa-large	JSD	12	AVG	42.3	40%	50%	43%	44%	**48%**	56%
XLNet-base	HEL	7	AVG	**48.3**	**62%**	53%	**50%**	**58%**	**49%**	**74%**
XLNet-large	HEL	11	AVG	46.7	57%	50%	54%	50%	**57%**	**73%**
Other models										
PRPN(tuned)† ‡	-	-	-	47.3	50%	59%	46%	57%	44%	32%
ON(tuned)† ‡	-	-	-	48.1	51%	**64%**	41%	54%	38%	31%
Neural PCFG†	-	-	-	**50.8**	52%	**71%**	33%	**58%**	32%	45%
Compound PCFG†	-	-	-	**55.2**	**56%**	**74%**	41%	**68%**	40%	52%

Table 5.3 Results on the PTB test set. **Bold** numbers correspond to the top 3 results for each column. L: layer number, A: attention head number (AVG: the average of all attentions). †: Results reported by Kim et al. (2019c). ‡: Approaches in which $\overline{\text{COO}}$ parser is utilized.

existing grammar induction models.

5.5.2 Experimental Results on PTB

In Table 5.3, we report the results of various models on the PTB test set. First of all, our method combined with PLMs shows competitive or comparable results in terms of S-F1 even without the right-skewness bias. This result implies that the extracted trees from our method can be regarded as a baseline for English

grammar induction. Moreover, PLMs show substantial improvements over Random Transformers (XLNet-base), demonstrating that training language models on large corpora, in fact, enables the PLMs to be more aware of syntactic information.

When the right-skewness bias is applied to the syntactic distances derived from PLMs, the S-F1 scores of the PLMs increase by up to ten percentage points. This improvement indicates that PLMs do not properly capture the largely right-branching nature of English syntax, at least when observed through the lens of our framework. By explicitly controlling the bias through our framework and observing the performance gap between our models with and without the bias, we confirm that the main contribution of the bias comes from its capability to capture subordinate clauses (SBAR) and verb phrases (VP). This observation provides a hint for what some previous work on unsupervised parsing desired to obtain by introducing the bias to their models. It is intriguing to see that all of the existing grammar induction models are inferior to the right-branching baseline in recognizing SBAR and VP (although some of them already utilized the right-skewness bias), implying that the same problem—models do not properly capture the right-branching nature—may also exist in current grammar induction models. One possible assumption is that the models do not need the bias to perform well in language modeling, although future work should provide a rigorous analysis about the phenomenon.

On the other hand, the existing models show exceptionally high recall scores on noun phrases (NP), even though our PLMs also have success to some extent in capturing noun phrases compared to naïve baselines. From this, we conjecture that neural models trained with a language modeling objective become largely equipped with the ability to understand the concept of NP. In contrast, the PLMs record the best recall scores on adjective and adverb phrases (ADJP

and ADVP), suggesting that the LMs and existing models capture disparate aspects of English syntax to differing degrees. To further explain why some PLMs are good at capturing ADJPs and ADVPs, we manually investigated the attention heatmaps of the sentences that contain ADJPs or ADVPs. From the inspection, we empirically found that there are some keywords—including 'two', 'ago', 'too', and 'far'—which have different patterns of attention distributions compared to those of their neighbors, and that these keywords can be a clue for our framework to recognize the existence of ADJPs or ADJPs. It is also worth mentioning that ADJPs and ADVPs consist of a relatively smaller number of words than those of SBAR and VP, indicating that the PLMs combined with our method have strength in correctly finding small chunks of words, i.e., low-level phrases.

Meanwhile, in comparison with other LM models, GPT-2 and XLNet based models demonstrate their effectiveness and robustness in unsupervised parsing. Particularly, the XLNet-base model serves as a robust baseline achieving the top performance among LM candidates. One plausible explanation for this outcome is that the training objective of XLNet, which considers both autoencoding (AE) and autoregressive (AR) features, might encourage the model to be better aware of phrase structure than other LMs. Another possible hypothesis is that AR objective functions (e.g., typical language modeling) are more effective in training syntax-aware neural models than AE objectives (e.g., masked language modeling), as both GPT-2 and XLNet are pre-trained on AR variants. However, it is hard to conclude what factors contribute to their high performance at this stage.

Interestingly, there is an obvious trend that the functions in F^d—the distance measure functions for attention distributions—lead most of the LM instances to the best parsing results, indicating that deriving parse trees from

attention information can be more compact and efficient than extracting them from the LMs' intermediate representations, which should contain linguistic knowledge beyond phrase structure. In addition, the results in Table 5.3 show that large parameterizations of the LMs generally increase their parsing performance, but this improvement is not always guaranteed. Meanwhile, as we expected in Section 5.4.3 and as seen in the 'A' (attention head number) column of Table 5.3, the average of attention distributions in the same layer often provides better results than individual attention distributions.

5.5.3 Experimental Results on MNLI

We present the results of various models on the MNLI test set in Table 5.4. We observe trends in the results which mainly coincide with those of the PTB dataset. Particularly, (1) right-branching trees are strong baselines for the task, especially showing their strengths in capturing SBAR and VP clauses/phrases, (2) our method resorting to PLMs is also comparable to the right-branching trees, demonstrating its superiority in recognizing different aspects of phrase categories including prepositional phrases (PP) and adverb phrases (ADVP), and (3) attention distributions seem more effective for distilling the phrase structures of sentences than intermediate representations.

However, there are some issues worth mentioning. First, the right-branching baseline seems to be even stronger in the case of MNLI, recording a score of over 50 in sentence-level F1. We conjecture that this result comes principally from two reasons: (1) the average length of sentences in MNLI is much shorter than in PTB, giving a disproportionate advantage to naïve baselines, and (2) our data preprocessing, which follows Kim et al. (2019c), removes all punctuation marks, unlike previous work (Htut et al., 2018; Drozdov et al., 2019), leading to an unexpected advantage for the right-branching scheme. Moreover, it deserves

Models	f	L	A	S-F1	SBAR	NP	VP	PP	ADJP	ADVP
Baselines										
Random Trees	-	-	-	21.4	11%	25%	16%	22%	22%	27%
Balanced Trees	-	-	-	20.0	8%	29%	11%	20%	22%	32%
Left Branching Trees	-	-	-	8.4	6%	13%	1%	4%	1%	8%
Right Branching Trees	-	-	-	51.9	**65%**	28%	**75%**	47%	45%	30%
Random XLNet-base (F^v)	-	-	-	22.0	12%	26%	15%	22%	22%	25%
Random XLNet-base (F^d)	-	-	-	23.5	14%	26%	18%	22%	22%	25%
PLMs (λ=0)										
BERT-base	HEL	9	10	36.1	36%	37%	34%	45%	26%	42%
BERT-large	JSD	17	10	37.0	38%	32%	34%	50%	22%	39%
GPT2	JSD	1	10	44.0	43%	**53%**	31%	60%	24%	40%
GPT2-medium	JSD	3	12	49.1	57%	32%	61%	44%	35%	37%
RoBERTa-base	JSD	10	9	36.2	26%	35%	34%	50%	23%	44%
RoBERTa-large	JSD	3	6	39.8	20%	28%	35%	30%	28%	27%
XLNet-base	HEL	1	6	39.0	25%	39%	28%	59%	35%	44%
XLNet-large	HEL	1	15	42.2	32%	49%	27%	**62%**	32%	**49%**
PLMs (λ=1.5)										
BERT-base	HEL	2	12	52.7	**64%**	35%	70%	50%	**46%**	30%
BERT-large	HEL	4	4	51.7	63%	31%	**71%**	49%	**46%**	30%
GPT2	HEL	1	10	52.2	57%	**53%**	49%	**62%**	32%	42%
GPT2-medium	HEL	2	1	**53.9**	53%	**57%**	50%	**62%**	29%	44%
RoBERTa-base	HEL	2	3	52.0	**64%**	31%	**72%**	49%	**47%**	30%
RoBERTa-large	L1	23	-	52.7	55%	40%	65%	53%	43%	41%
XLNet-base	L2	8	-	**54.9**	57%	49%	61%	55%	44%	**57%**
XLNet-large	L2	12	-	**53.5**	54%	47%	59%	51%	**48%**	**60%**
Other models										
PRPN-UP†‡	-	-	-	48.6*	-	-	-	-	-	-
PRPN-LM†‡	-	-	-	50.4*	-	-	-	-	-	-
DIORA†	-	-	-	51.2*	-	-	-	-	-	-
DIORA(+PP)†	-	-	-	59.0*	-	-	-	-	-	-

Table 5.4 Results on the MNLI test set. **Bold** numbers correspond to the top 3 results for each column. L: layer number, A: attention head number (AVG: the average of all attentions). †: Results reported by Htut et al. (2018) and Drozdov et al. (2019). ‡: Approaches in which $\overline{\text{COO}}$ parser is utilized. $*$: These results are not strictly comparable to ours, due to the difference in data preprocessing.

to consider the fact that the gold-standard parse trees in MNLI are not human-annotated, rather automatically generated.

Second, in terms of consistency in identifying the best choice of f and g for each LM, we observe that most of the best combinations of f and g tuned for PTB do not correspond well to the best ones for MNLI. Does this observation imply that a specific combination of these functions and the resulting

performance do not generalize well across different data domains? To clarify, we manually investigated the performance of some combinations of f and g, which are tuned on PTB but tested on MNLI instead. As a result, we discover that particular combinations of f and g which are good at PTB are also competitive on MNLI, even though they fail to record the best scores on MNLI. Concretely, the union of f^d(JSD) and $g^d_{(9,13)}$—the best duo for the XLNet-base on PTB—achieves 39.2 in sentence-level F1 on MNLI, which is very close to the top performance (39.3) we can obtain when leveraging the XLNet-base. It is also worth noting that GPT-2 and XLNet are efficient in capturing PP and ADVP respectively, regardless of the data domain and the choice of f and g.

5.6 Analysis

5.6.1 Performance Comparison by Layer

To take a closer look at how different the layers of PLMs are in terms of parsing performance, we retrieve the best sentence-level F1 scores from the lth layer of each LM from all the combinations of f and g_l, with regard to the PTB dataset, and then we plot the scores in Figure 5.2. Each score is from the models to which the bias is *not* applied.

From the graphs, we observe several patterns. First, XLNet-based models outperform other competitors across most of the layers. Second, the best outcomes are largely shown in the middle layers of the LMs akin to the observation from Shen et al. (2019), except for some cases where the first layers (especially in case of MNLI) record the best. Interestingly, GPT-2 shows a decreasing trend in its output values as the layer becomes high, while other models generally exhibit the opposite pattern. Moreover, we discover from raw statistics that regardless of the choice of f and g_l, the parsing performance reported as S-F1 is moderately correlated with the layer number l. In other words, it seems

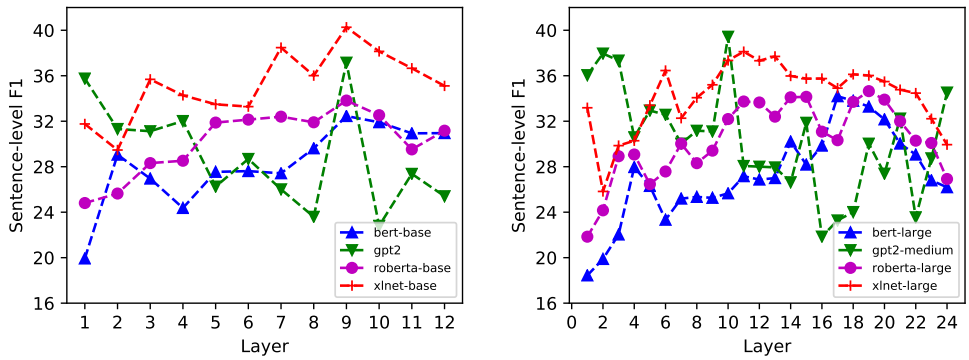

Figure 5.2 The best layer-wise S-F1 scores of PLMs on the PTB test set. (Left) The performance of the X-'base' models. (Right) The performance of the X-'large' models.

that there are some particular layers in the LMs which are more sensitive to syntactic information.

5.6.2 Estimating the Upper Limit of Distance Measure Functions

Although we have introduced effective candidates for f, we explore the potential of extracting more sophisticated trees from PLMs, supposing we are equipped with a pseudo-optimum f, call it f_{ideal}. To obtain f_{ideal}, we train a simple linear layer on each layer of the PLMs *with supervision* from the gold-standard trees of the PTB training set, while g remains unchanged—the PLMs are frozen during training. We choose the XLNet-base model as a representative for PLMs. We assume f_{ideal} is only compatible with the functions in G^v, as the functions in G^d are not suitable for training as the sizes of the representations provided by G^d are variable according to the length of an input sentence. To train the pseudo-optimal function f_{ideal}, we minimize a pair-wise learning-to-rank loss following

Model	f	L	A	S-F1	SBAR	NP	VP	PP	ADJP	ADVP
Baselines (from Table 1)										
Random XLNet-base (F^v)	-	-	-	19.6	9%	26%	12%	20%	23%	24%
Random XLNet-base (F^d)	-	-	-	20.1	11%	25%	14%	19%	22%	26%
XLNet-base (λ=0)	JSD	9	AVG	40.1	35%	56%	26%	38%	47%	68%
XLNet-base (λ=1.5)	HEL	7	AVG	48.3	**62%**	53%	50%	58%	49%	**74%**
Trained models (gold trees)										
Random XLNet-base	f_{ideal}	-	-	41.2	28%	58%	29%	50%	35%	41%
XLNet-base (worst case)	f_{ideal}	1	-	58.0	47%	75%	56%	71%	50%	61%
XLNet-base (best case)	f_{ideal}	7	-	**65.1**	61%	**82%**	**67%**	**78%**	**55%**	73%

Table 5.5 Results of training a pseudo-optimum f_{ideal} with PTB and XLNet-base model.

previous work (Burges et al., 2005; Shen et al., 2018c):

$$L_{\text{dist}}^{\text{rank}} = \sum_{i,j>i} [1 - \text{sign}(d_i^{\text{gold}} - d_j^{\text{gold}})(d_i^{\text{pred}} - d_j^{\text{pred}})]^+,$$

where d^{gold} and d^{pred} are computed from the gold tree and our predicted one, respectively. $[x]^+$ is defined as $max(0, x)$. We train the f_{ideal} with the PTB training set for 5 epochs. Each batch of the training set contains 16 sentences. We use an ADAM optimizer (Kingma and Ba, 2014) with the learning rate 5e-4. We train the variations of f_{ideal} differentiated by the choice of g in G^v and report the best result in the Table 5.5. Each f_{ideal} is chosen based on its performance on the PTB validation set. Considering the randomness of training, every result for f_{ideal} is averaged over 3 different trials.

In Table 5.5, we present three new results using f_{ideal}. As a baseline, we report the performance of f_{ideal} with a randomly initialized XLNet-base. Then, we list the worst and best result of f_{ideal} according to g, when it is combined with the pre-trained LM. We here mention some findings from the experiment. First, comparing the results with the pre-trained LM against one with the random LM, we reconfirm that pre-training an LM apparently enables the model to capture some aspects of grammar. Specifically, our method is comparable

to the linear model trained on the gold-standard trees. Second, we find that there is a tendency for the performance of f_{ideal} relying on different LM layers to follow one we already observed in Section 5.6.1—the best result comes from the middle layers of the LM while the worst from the first and last layer. Third, we identify that the LM has a potential to show improved performance on grammar induction by adopting a more sophisticated f. However, we emphasize that our method equipped with a simple f without gold-standard trees is remarkably reasonable in recognizing constituency grammar, being especially good at catching ADJP and ADVP.

5.6.3 Constituency Tree Examples

We visualize several gold-standard trees from PTB and the corresponding tree predictions for comparison. We randomly select four sentences from PTB and visualize their trees from Figure 5.3 to Figure 5.6, where the resulting group of trees for each sentence consists of a gold constituency tree and two induced trees (one *without* the right-skewness bias and the other *with* the bias) from our best model—XLNet-base. The 'T' character in the induced trees indicates a dummy tag.

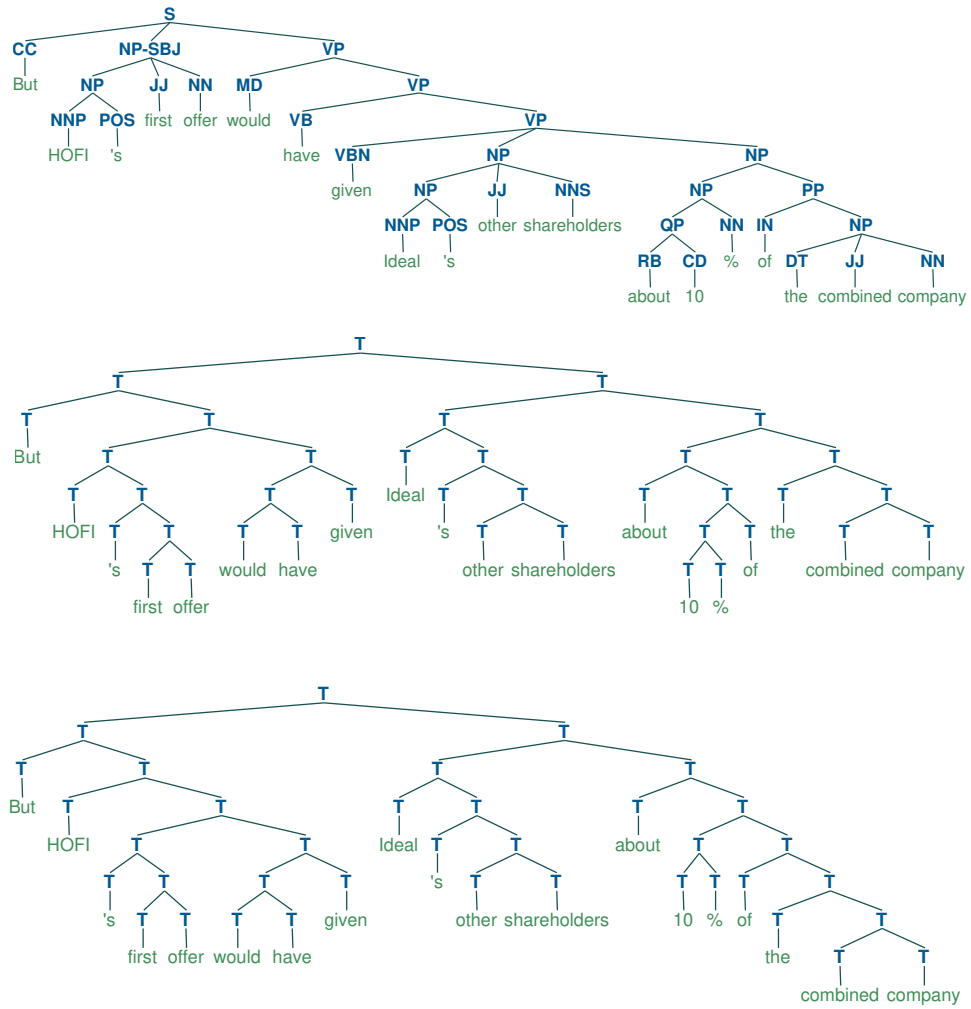

Figure 5.3 Gold (**top**) and predicted trees (one *without* the bias in the **middle**, the other with the bias at the **bottom**) for the sentence 'But HOFI 's first offer would have given Ideal 's other shareholders about 10 % of the combined company'.

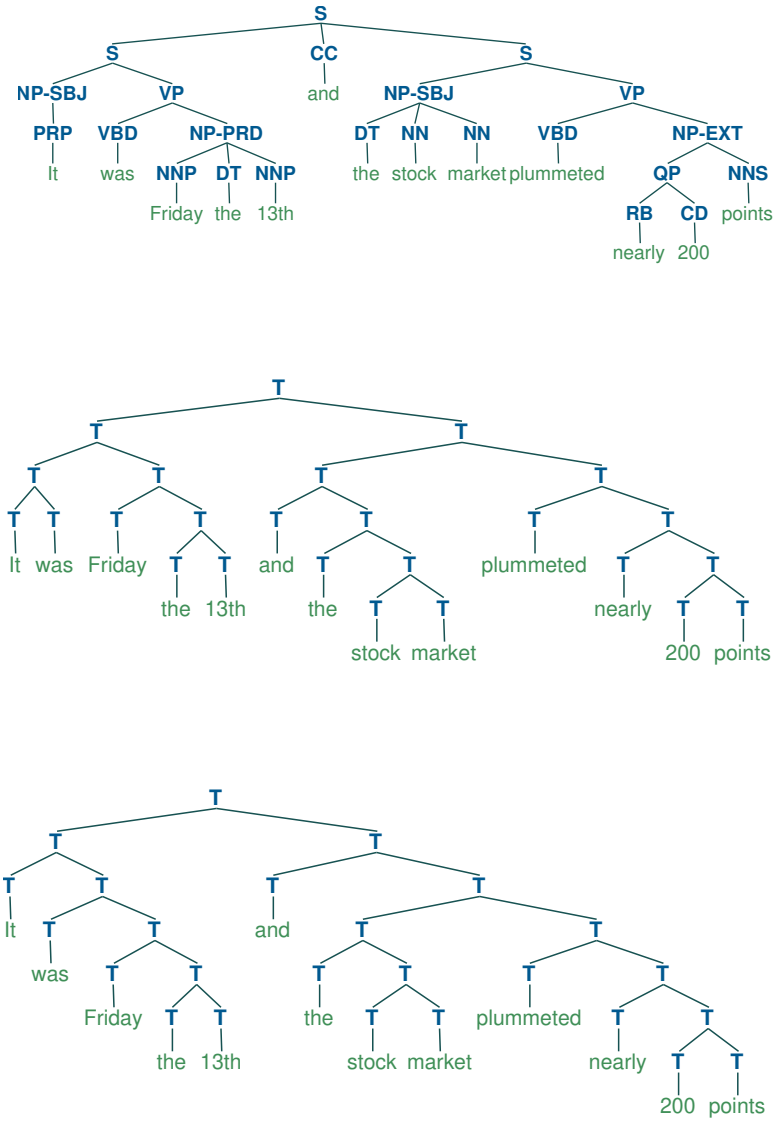

Figure 5.4 Gold (**top**) and predicted trees (one *without* the bias in the **middle**, the other with the bias at the **bottom**) for the sentence 'It was Friday the 13th and the stock market plummeted nearly 200 points'.

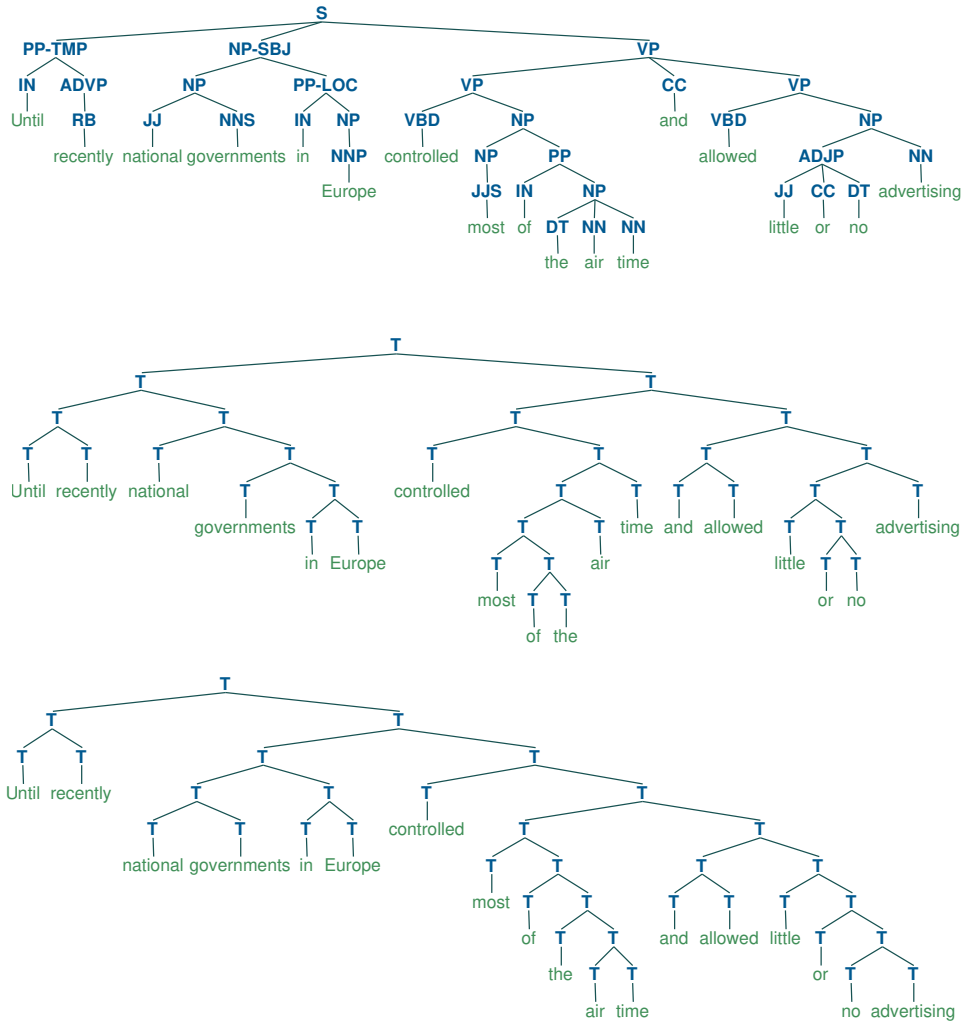

Figure 5.5 Gold (**top**) and predicted trees (one *without* the bias in the **middle**, the other with the bias at the **bottom**) for the sentence 'Until recently national governments in Europe controlled most of the air time and allowed little or no advertising'.

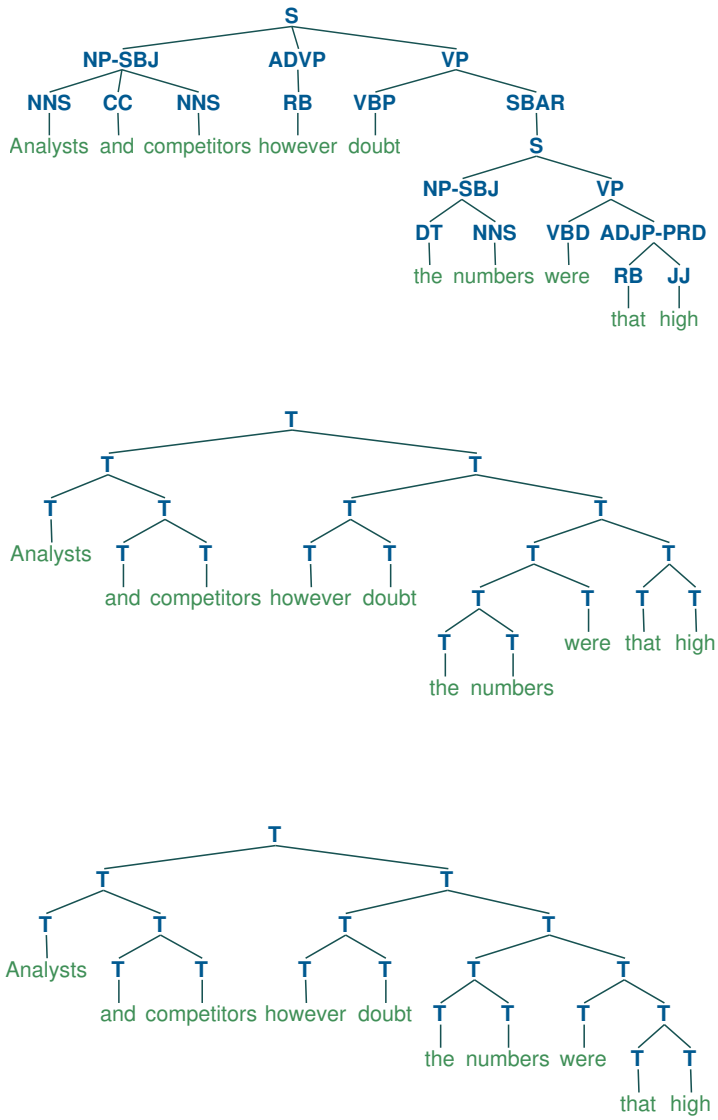

Figure 5.6 Gold (**top**) and predicted trees (one *without* the bias in the **middle**, the other with the bias at the **bottom**) for the sentence 'Analysts and competitors however doubt the numbers were that high'.

5.7 Summary

We have proposed a simple but effective method of inducing constituency trees from pre-trained language models in a zero-shot learning fashion. Furthermore, we have reported a set of intuitive findings observed from the extracted trees, demonstrating that PLMs exhibit some properties similar to constituency grammar. In addition, we have shown that our method can serve as a strong baseline for English grammar induction when combined with (or even without) appropriate linguistic biases.

On the other hand, there are still remaining issues that can be good starting points for future work. First, although we analyzed our method based on two popular datasets, we focused only on English grammar induction. As each language has its own properties (and correspondingly would need individualized biases), it is desirable to expand this work to other languages. In the next chapter, we overcome this limitation by considering eight additional languages in evaluation. Second, it would also be desirable to investigate whether further improvements can be achieved by directly grafting the PLMs onto existing grammar induction models. Lastly, by verifying the usefulness of the knowledge from the PLMs and linguistic biases for grammar induction, we want to point out that there is still much room for improvement in the existing grammar induction models.

Chapter 6

Multilingual Syntactic Analysis with Enhanced Techniques

6.1 Introduction

In the previous chapter, we have presented a novel methodology of inducing constituency parse trees from pre-trained language models (PLMs) and thus paving the way for investigating the inner workings of the models. We improve upon this paradigm by proposing a novel chart-based method and an effective top-K ensemble technique. Moreover, we demonstrate that we can broaden the scope of application of the approach into multilingual settings.

Constituency **P**arse **E**xtraction from **P**re-trained **L**anguage **M**odels (**CPE-PLM**) relies on the combination of (i) simple distance metric functions and (ii) the representations obtained from PLMs. The core assumption underlying the methodology is that PLMs hold syntactic knowledge enough to be utilized for predicting parse trees by themselves. Although the CPE-PLM framework has demonstrated that non-trivial trees resembling gold-standard annotations can

be extracted from general PLMs even without fine-tuning on treebanks, it has been also reported that CPE-PLM's freedom from task-specific training comes at the cost of its performance inferior to that of unsupervised parsers. Furthermore, it has yet to be verified that CPE-PLM is also effective for languages other than English.

In this chapter, we first attempt to narrow the performance gap between unsupervised parsers and CPE-PLM by introducing a novel method inspired by neural chart-based algorithms (Durrett and Klein, 2015; Stern et al., 2017; Kitaev and Klein, 2018). In contrast to the top-down CPE-PLM method in the previous chapter, which focuses on detecting the boundary of two subspans in a phrase relying only on the knowledge from the two words around the boundary, our chart-based method considers all components in a phrase to judge how plausible the phrase is. Furthermore, we introduce a simple but effective ensemble technique that utilizes the pre-defined set of attention heads which are confirmed as being effective by their performance on the validation set. We show that our chart-based method outperforms or is competitive to the top-down method and that the top-K ensemble plays a key role in boosting their performance.

Second, the limitation of most previous studies for both unsupervised parsing and CPE-PLM is that they are heavily English-centric, leaving an open question whether they are universally applicable. To investigate this problem, we test CPE-PLM on several other languages. Specifically, we propose to introduce *multilingual* PLMs (Conneau and Lample, 2019; Conneau et al., 2020) into CPE-PLM to grant the framework an ability to deal with multiple languages simultaneously. We show that the CPE-PLM methods built upon multilingual PLMs are able to induce reasonable parses for sentences in nine languages in an integrated and language-agnostic manner, achieving figures superior or com-

parable to ones from neural PCFGs (Kim et al., 2019c; Zhao and Titov, 2021). In supplementary analyses, we provide intuitive explanations about the inner workings of our method. For instance, we confirm the existence of *universal* attention heads which seem to be responsible for capturing syntactic information irrespective of the input language.

6.2 Related work

Pre-trained language models (PLMs) now lie at the heart of many studies in the literature. Following the trend, much effort has been made to develop English PLMs (Devlin et al. (2019); Liu et al. (2019b); Radford et al. (2019); Yang et al. (2019), *inter alia*), to construct non-English PLMs (Martin et al. (2019); *i.a.*), and to train multilingual variants (Conneau and Lample, 2019; Conneau et al., 2020). We explore the potential use of these PLMs as zero-shot parsers. The trees induced by our method can also be leveraged as a tool for probing PLMs, similar to the recent work that attempts to explore some knowledge in PLMs (Clark et al., 2019; Jawahar et al., 2019). In particular, Chi et al. (2020) extended Hewitt and Manning (2019) to multilingual settings, analogous to our work. Still, it is different from ours in that it requires explicit supervision and devotes itself to dependency grammar.

Mareček and Rosa (2019) developed an approach similar to ours, but they focused on Transformers trained for machine translation rather than language models. Work on neural unsupervised parsing (Shen et al. (2018b, 2019); Kim et al. (2019c); Shi et al. (2020), *inter alia*) also seeks to generate parse trees without supervision from gold-standard trees. It is worth noting that some work such as Kann et al. (2019) and Zhao and Titov (2021) attempts to evaluate unsupervised parsers in multilingual settings, akin to our work. The difference

between ours and theirs is that our method does not require training a parser for each language, instead relying on the off-the-shelf PLMs.

6.3 Method

6.3.1 Chart-based CPE-PLM

Although the top-down method in the previous chapter has shown its effectiveness in extracting non-trivial phrase structures from PLMs, there still remains much room for improvement, considering that this method by nature operates in a greedy fashion rather than taking account of the probabilities of all possible subtrees. In other words, the top-down CPE-PLM method only relies on the information obtained from the representations of two words to estimate the likelihood of the space between the two words becoming a target to be split. To overcome this limitation, we propose a novel approach based on chart parsing which executes an exact inference to find the most probable parse while effectively considering all possibilities with dynamic programming.

Following the previous work on chart parsing (Stern et al., 2017; Kitaev and Klein, 2018), we assign a real-valued score $s_{tree}(T)$ for each tree candidate T, which decomposes as $s_{tree}(T) = \sum_{(i,j)\in T} s_{span}(i,j)$, where $s_{span}(i,j)$ is a score (or cost) for a constituent that is located between positions i and j ($1 \leq i \leq j \leq n$) in a sentence. Specifically, $s_{span}(i,j)$ is defined as follows:

$$s_{span}(i,j) = \begin{cases} s_{comp}(i,j) + \min_{i \leq k < j} s_{split}(i,k,j) & \text{if } \ i < j \\ 0 & \text{if } \ i = j \end{cases}$$

where $s_{split}(i,k,j) = s_{span}(i,k) + s_{span}(k+1,j)$. In other words, $s_{comp}(i,j)$ measures the validity or compositionality of the span (i,j) itself while $s_{split}(i,k,j)$ indicates how plausible it is to divide the span (i,j) into two subspans (i,k)

Algorithm 2 Chart to Syntactic Distance

n: Length of an input sentence S.
$C \in \mathbb{R}^{n \times n}$: Chart matrix whose elements are $s_{span}(i, j)$.
$P \in \mathbb{R}^{n \times n}$: Matrix, whose $(i, j)^{th}$ element is the split point of the span (i, j) of the sentence S.
s: Start position, initialized as 1.
e: End position, initialized as n.
function C2D(C, P, s, e)
    if $s = e$ **then**
        return [] (empty vector)
    else
        $v \leftarrow C[s][e]$
        $p \leftarrow P[s][e]$
        return [C2D(C, P, s, p); v; C2D(C, P, p+1, e)]
        ([·; ·]: vector concatenation)
    end if
end function

and $(k + 1, j)$. We choose the most probable k that brings us the minimum cost of $s_{split}(i, k, j)$. Note that each constituent is by definition evaluated with the scores of its children in addition to its own score. Once $s_{comp}(\cdot, \cdot)$ is properly defined, it is straightforward to compute every $s_{span}(i, j)$ by utilizing the CKY algorithm (Cocke, 1969; Kasami, 1966; Younger, 1967). In the following subsections, we formulate two variants of $s_{comp}(\cdot, \cdot)$ in detail.

Finally, our parser outputs $\hat{T}$, the tree that requires the lowest score (cost) to build, as a prediction for the parse tree of the input sentence: $\hat{T} = \text{argmin}_T \, s_{tree}(T)$. Technically, after completing filling in the chart matrix, each of whose element is $s_{span}(i, j)$, we derive $\hat{T}$ by applying Algorithm 2 and Algorithm 1 (Chapter 5) in order.

Pair Score Function

The methodology introduced above abstracted over the choice of $s_{comp}(\cdot, \cdot)$; in what follows we propose two candidates for it. First, we propose a pair score function $s_p(\cdot, \cdot)$ which is defined as follows:

$$s_p(i, j) := \binom{j - i + 1}{2}^{-1} \sum_{(w_x, w_y) \in pair(i,j)} f(g(w_x), g(w_y)),$$

where $pair(i, j)$ returns a set consisting of all combinations of two words from a span (i, j)—e.g., $pair(1, 3) = \{(w_1, w_2), (w_1, w_3), (w_2, w_3)\}$. The intuition behind this formulation is that every pair of words in a constituent should have similar attention distributions so that the pair's embeddings become similar to each other in the subsequent layers of PLMs. For $f(\cdot, \cdot)$ and $g(\cdot)$, we again take advantage of F^d and G^d specified in Section 5.4.4.[1]

Characteristic Score Function

We also propose another candidate for $s_{comp}(\cdot, \cdot)$, namely a characteristic score function $s_c(\cdot, \cdot)$. Instead of measuring the similarities of all pairs of attention distributions, we pre-define $\boldsymbol{c}$ as the characteristic vector of a given constituent and evaluate the cost of each word in the constituent with regard to this value. Although $\boldsymbol{c}$ can be realized in many ways, for simplicity, we here use the average of all the attention distributions of words in a constituent. As a consequence, $s_c(i, j)$ is formalized as follows:

$$s_c(i, j) := \frac{1}{j - i + 1} \sum_{i \leq x \leq j} f(g(w_x), \boldsymbol{c}),$$

where $\boldsymbol{c} = \frac{1}{j-i+1} \sum_{i \leq y \leq j} g(w_y)$.

[1] This implies that we make only use of the attention distributions of PLMs, as it is verified by the previous work (Kim et al., 2020) and our preliminary experiments that attention distributions offer more useful signals in this setting.

6.3.2 Top-K Ensemble for CPE-PLM

The part remaining ambiguous so far in clarifying CPE-PLM algorithms is about how to properly select the distance measure function f and representation extractor function g from the set of candidates, i.e., F^d and G^d. Basically, we can consider a typical case where we acquire the best combination of f and g using the validation set and apply it to the test set. In addition, we introduce one more option, called *top-K ensemble*, that enables us to integrate the knowledge from several attention heads.

Specifically, we first pick an arbitrary candidate for f, dubbed $\hat{f}$.[2] Then, we compute the parsing performance of every possible combination of $\hat{f}$ and $g \in G^d$ on the validation set and sort G^d according to the performance of its elements. After that, we simply choose the top K elements from the sorted G^d and allow all of them (G^d_{topK}) to participate in parsing instead of just leveraging the best single one. At test time, given an input sentence, we predict K separate trees using every element from G^d_{topK}, and then convert the trees into corresponding syntactic distance vectors (Algorithm 2). Finally, we compute the average of the syntactic distance vectors and translate this averaged vector into the final tree prediction (Algorithm 1). By introducing the top-K ensemble technique, it becomes possible to obtain a more accurate tree prediction while seamlessly combining diverse syntactic signals provided by different attention heads.

6.4 Experiments

6.4.1 General Configurations

To evaluate, we prepare the PTB (Marcus et al., 1993) dataset for English and the SPMRL (Seddah et al., 2013) dataset for the eight other languages: Basque,

[2]In practice, we test every element $f \in F^d$ exhaustively to select the best one as F^d consists of only two elements.

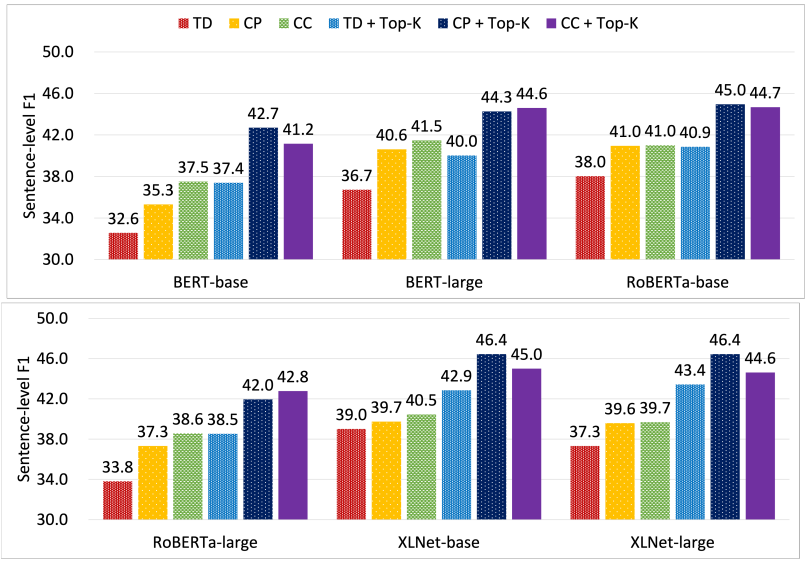

Figure 6.1 Performance of CPE-PLM methods on PTB. Chart-based (CP and CC) approaches show superior figures in most cases compared to TD. The top-K ensemble also provides orthogonal improvements.

French, German, Hebrew, Hungarian, Korean, Polish, and Swedish. We use the standard split of each dataset, and the datasets are preprocessed following Kim et al. (2019c) and Zhao and Titov (2021)—removing punctuation marks. We leverage the *unlabelled sentence-level F1* (percentage) score as a primary metric to evaluate the extent to which tree predictions resemble corresponding gold-standard trees. The hyperparameter K, which determines the number of attention heads engaging in the top-K ensemble, is decided by grid search on some reasonable candidates ($\{5, 10, 20, 30\}$). We empirically found that K=20 is versatile across different settings. From now on, we employ the abbreviations TD, CP, and CC to refer to the *top-down* (Chapter 5), *chart-pair* (the approach defined in this chapter and equipped with s_p), and *chart-characteristic* method (the approach defined in this chapter and with s_c) respectively.

Language	French			German			Korean			Swedish		
Method	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**
Naïve baselines												
Random		16.2			13.9			22.2			16.4	
Left-branching		5.7			10.0			18.5			8.4	
Right-branching		26.4			14.7			19.2			30.4	
PCFGs												
N-PCFG†		42.2			37.8			25.7			14.5	
C-PCFG†		40.5			37.3			27.7			23.7	
Mono. PLMs	41.4	42.4	**42.8**	38.4	39.6	**39.7**	**51.1**	47.3	47.4	35.6	38.4	**38.9**

Table 6.1 CPE-PLM's performance on French, German, Korean, and Swedish. The best score for each language is in **bold**. †: results from Zhao and Titov (2021).

6.4.2 Experiments on Monolingual Settings

We first assess CPE-PLM on the PTB dataset. We apply our methods to three different categories of English PLMs—BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), and XLNet (Yang et al., 2019).[3] We also test the effect of the top-K ensemble by combining it with each of the CPE-PLM method. In Figure 6.1, we confirm that our chart-based methods mostly outperform the top-down approach, showing an improvement of up to nearly five points (RoBERTa-large: 33.8 $\rightarrow$ 38.6). Moreover, we reveal that the top-K ensemble provides significant improvements on parsing performance in an orthogonal manner, regardless of the accompanying method. This result implies that the cues that can contribute to inducing parse trees may be distributed across different parts of PLMs, rather than concentrated on a specific attention head. We attain the 46.4 F1 score by combining XLNet, the CP method, and the top-K ensemble, which is six points higher than the best performance (40.1) reported in the previous chapter.

[3]We prepare two variants for each PLM: (i) *X-base* consists of 12 layers, 12 attention heads, and 768 hidden dimensions. (ii) *X-large* has 24 layers, 16 heads, and 1024 dimensions.

Next, we evaluate CPE-PLM methods on French, German, Korean, and Swedish, for which language-specific BERT variants are available.[4] As baselines, we prepare three naïve methods—random and left/right-branching trees—in addition to N(eural)-PCFG and C(ompound)-PCFG (Kim et al., 2019c), which are representative unsupervised parsers.[5] For CPE-PLM, we consider both the top-down and chart methods, all of which are combined with the top-K ensemble. In Table 6.1, the CPE-PLM methods demonstrate performance comparable (French / German) or superior (Korean / Swedish) to that of the strong baselines. In particular, CPE-PLM shows much better performance in Korean and Swedish, where PCFGs failed to obtain meaningful results. We conjecture this discrepancy in part comes from the availability of subword-level features, to which PLMs have access while PCFGs do not, considering that the SPMRL dataset is originally constructed for testing morphologically-rich languages. Meanwhile, CP and CC outperform TD on 3 out of 4 languages, albeit the gaps are relatively small compared to the English case. This outcome leads to two implications: (i) the effectiveness of a CPE-PLM method depends on the language where it is applied, and (ii) our top-K ensemble is broadly helpful for all the parsing methods, reducing the gap between their performance.

6.4.3 Experiments on Multilingual Settings

[4]The PLMs we utilize per language are listed as follows. **German:** bert-base-german (`https://deepset.ai/german-bert`). **French:** camembert (Martin et al., 2019). **Swedish:** bert-base-swedish (`https://github.com/huggingface/transformers/tree/master/model_cards/KB/bert-base-swedish-cased`). **Korean:** KoBERT-base (`https://github.com/SKTBrain/KoBERT`).

[5]We rely on Zhao and Titov (2021) to report PCFG models' performance. Note that the authors assume they have access to parses in the PTB dev set to select the best hyperparameters following Kim et al. (2019c). This condition is exactly the same for our English models and cross-lingual transfer settings in Table 6.2. Meanwhile, we additionally employ the SPMRL validation set in Table 6.1 and the Multi-ling. part of Table 6.2, which can give undesirable extra gains to CPE-PLM.

Language	English			Basque			French			German			Hebrew			Hungarian			Korean			Polish			Swedish		
Method	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**
PCFGs																											
N-PCFG†		50.8			30.2			42.2			37.8			41.0			37.9			25.7			31.7			14.5	
C-PCFG†		55.7			27.9			40.5			37.3			39.2			38.3			27.7			32.4			23.7	
Mono-ling.	42.9	46.4	45.0				41.4	42.4	42.8	38.4	39.6	39.7							51.1	47.3	47.4				35.6	38.4	38.9
Multi-ling.																											
M-BERT	40.3	**45.0**	44.8	40.0	40.9	**41.6**	42.9	44.6	**45.6**	39.3	**40.6**	40.3	42.3	**42.5**	42.0	38.2	39.1	**40.4**	**52.1**	50.9	49.8	41.6	**43.0**	42.9	37.3	37.4	**39.3**
XLM	44.2	**47.7**	46.2	43.3	44.1	**44.7**	43.7	**46.0**	**46.0**	39.2	**41.3**	40.4	43.9	44.2	**44.3**	40.8	**42.3**	42.2	**43.0**	41.6	41.0	44.3	**44.9**	44.4	39.0	40.1	**40.7**
XLM-R	45.5	46.7	**47.0**	43.7	43.8	**45.1**	**45.8**	44.2	45.5	41.4	**42.2**	41.6	45.0	43.2	**45.3**	42.4	**44.0**	43.4	**55.9**	55.7	54.3	43.1	43.7	**44.6**	39.5	40.6	**41.5**
XLM-R-L	41.7	44.6	**45.1**	44.3	44.1	**45.2**	39.5	42.4	**42.9**	38.9	**41.0**	40.7	43.7	44.1	**46.3**	39.8	**41.5**	41.3	51.8	**52.6**	51.8	41.7	43.5	**44.5**	36.2	38.6	**39.4**
Cross-ling.																											
M-BERT				39.8	39.8	**41.1**	42.2	44.6	**45.5**	37.7	**40.3**	39.3	39.7	**42.8**	**42.8**	36.2	**39.4**	38.0	**48.9**	47.0	45.7	39.8	41.9	**42.3**	36.0	**39.1**	38.7
(+,−)				(-0.2)	(-1.1)	(-0.5)	(-0.7)	(0.0)	(-0.1)	(-1.6)	(-0.3)	(-1.0)	(-2.6)	(+0.3)	(+0.8)	(-2.0)	(+0.3)	(-2.4)	(-3.2)	(-3.9)	(-4.1)	(-1.8)	(-1.1)	(-0.6)	(-1.3)	(+1.7)	(-0.6)
XLM				40.6	41.2	**42.1**	44.2	**46.3**	46.1	38.9	**41.5**	40.3	42.4	**45.8**	43.9	38.0	**42.2**	40.7	**40.1**	39.5	38.4	42.2	**44.5**	44.4	38.2	**40.9**	**40.9**
(+,−)				(-2.7)	(-2.9)	(-2.6)	(+0.5)	(+0.3)	(+0.1)	(-0.3)	(+0.2)	(-0.1)	(-1.5)	(+1.6)	(-0.4)	(-2.8)	(-0.1)	(-1.5)	(-2.9)	(-2.1)	(-2.6)	(-2.1)	(-0.4)	(0.0)	(-0.8)	(+0.8)	(+0.2)
XLM-R				43.4	42.1	**43.7**	45.4	45.1	**46.2**	41.5	**42.2**	41.5	45.5	45.2	**46.3**	41.3	**43.4**	41.9	**52.6**	49.6	48.9	44.3	**45.4**	44.8	40.4	41.0	**41.4**
(+,−)				(-0.3)	(-1.7)	(-1.4)	(-0.4)	(+0.9)	(+0.7)	(+0.1)	(0.0)	(-0.1)	(+0.5)	(+2.0)	(+1.0)	(-1.1)	(-0.6)	(-1.5)	(-3.3)	(-6.1)	(-5.4)	(+1.2)	(+1.7)	(+0.2)	(+0.9)	(+0.4)	(-0.1)
XLM-R-L				**43.9**	42.6	43.6	39.4	42.3	**43.2**	38.6	**40.6**	**40.6**	42.8	44.7	**45.4**	38.6	39.9	**40.7**	**51.6**	51.3	50.5	42.6	44.9	**45.1**	37.1	39.6	**40.0**
(+,−)				(-0.4)	(-1.5)	(-1.6)	(-0.1)	(-0.1)	(+0.3)	(-0.3)	(-0.4)	(-0.1)	(-0.9)	(+0.6)	(-0.9)	(-1.2)	(-1.6)	(-0.6)	(-0.2)	(-1.3)	(-1.3)	(+0.9)	(+1.4)	(+0.6)	(+0.9)	(+1.0)	(+0.6)

Table 6.2 Performance of CPE-PLM on 9 languages. **Mono-ling.**: CPE-PLM's performance in monolingual settings. **Multi-ling.**: the results when combined with multilingual PLMs. **Cross-ling.**: the performance when relying on cross-lingual transfer, in addition to the relative losses or gains (+,−) compared to the original results. The best score per PLM is in **bold** while the best for each language is underlined. †: results from Zhao and Titov (2021).

Theoretically, multilingual PLMs have a potential to be a core asset for CPE-PLM, given that they are able to deal with sentences from over a hundred languages simultaneously. However, it has not yet been investigated whether they can play a role as expected. To shed light on this issue, we conduct experiments with the CPE-PLM methods built upon multilingual PLMs. We apply four multilingual PLMs to nine languages in total. We use a multilingual version of the BERT-base model (M-BERT, Devlin et al. (2019)), the XLM model trained on 100 languages (XLM, Conneau and Lample (2019)), XLM-R, and XLM-R-L(arge) (Conneau et al., 2020). For baselines, we only consider PCFGs as we verified in Section 6.4.2 that they can subsume naïve baselines. We also list CPE-PLM's performance in monolingual settings for reference. Again, we utilize the TD, CP, and CC methods combined with the top-K ensemble.

In the **Multi-ling.** section of Table 6.2, we report CPE-PLM's performance with multilingual PLMs when the best attention heads are separately selected for each language, relying on the validation sets of respective languages. We observe that the CPE-PLM framework works pretty well across languages when it is built upon multilingual PLMs, outperforming PCFGs except for English. Surprisingly, we discover that for every language we consider, there exists at least one multilingual PLM that outperforms its monolingual counterpart. For instance, we achieve the F1 score of 47.7 in English with the XLM model, which is higher than all the scores we achieved for English in monolingual settings. In conclusion, we confirm that multilingual PLMs can serve as a core component for an integrated CPE-PLM framework that processes different languages simultaneously. Regarding the effect of parsing strategies, we identify that CP and CC generally outperform TD, and that the only exception occurs in Korean. We assume this is related to the linguistic properties of target languages, but we leave a thorough analysis on this as future work.

Next, we evaluate CPE-PLM in a harsher condition where the validation set is given only for English. Concretely, we attempt to parse sentences in eight other languages with the CPE-PLM methods optimized for English (i.e., the attention heads are chosen based on the PTB validation set), performing *zero-shot cross-lingual transfer* from English to others. Note that this constraint facilitates revealing the true value of CPE-PLM by answering the following research question: *given no access to parsers or gold-standard trees in target languages at all, can we induce non-trivial parse trees by solely relying on the knowledge residing in PLMs?*

In the **Cross-ling.** section, we present the performance of cross-lingual transfer and relative performance losses or gains (+,−) compared against the language-specific optimization (**Multi-ling.**). To our surprise, we reveal that the cross-lingual transfer leads to negligible losses or even small gains in most cases. This is also in line with the reports from related work (Pires et al. (2019); Cao et al. (2020); *i.a.*) that multilingual PLMs are effective in cross/multi-lingual NLP tasks. Our finding implies that there exist *universal* attention heads that are sensitive to the phrase structures of sentences irrespective of the input language. We seek to analyze this phenomenon in detail in the following section.

6.5 Analysis

In this section, we present several analyses that facilitate our understanding of the inner workings of CPE-PLM. We employ XLM-R, whose specification is elaborated in Conneau et al. (2020), as a backbone and the CC method as a parsing scheme.

Language	English			Basque			French			German			Hebrew			Hungarian			Korean			Polish			Swedish		
Method	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**	**TD**	**CP**	**CC**
Pre-training data																											
Tokens (M)†	300.8			2.0			56.8			66.6			31.6			58.4			54.2			44.6			12.1		
Size (GiB)†	55608			270			9780			10297			3399			7807			5644			6490			77		
Val. & Test data																											
Size (Validation)	1700			948			1235			5000			500			1051			2066			821			494		
Size (Test)	2416			946			2540			4999			716			1009			2287			822			666		
XLM-R	45.5	46.7	**47.0**	43.7	43.8	**45.1**	**45.8**	44.2	45.5	41.4	**42.2**	41.6	45.0	43.2	**45.3**	42.4	**44.0**	43.4	**55.9**	55.7	54.3	43.1	43.7	**44.6**	39.5	40.6	**41.5**
Cross-lingual				43.4	42.1	**43.7**	45.4	45.1	**46.2**	41.5	**42.2**	41.5	45.5	45.2	**46.3**	41.3	**43.4**	41.9	**52.6**	49.6	48.9	44.3	**45.4**	44.8	40.4	41.0	**41.4**
(+,−)				(-0.3)	(-1.7)	(-1.4)	(-0.4)	(+0.9)	(+0.7)	(+0.1)	(0.0)	(-0.1)	(+0.5)	(+2.0)	(+1.0)	(-1.1)	(-0.6)	(-1.5)	(-3.3)	(-6.1)	(-5.4)	(+1.2)	(+1.7)	(+0.2)	(+0.9)	(+0.4)	(-0.1)

Table 6.3 Factor correlation analysis. The first section describes the statistics of the data utilized for training XLM-R. The second section displays the characteristics of the validation and test sets. †: from Conneau et al. (2020).

6.5.1 Factor Correlation Analysis

First, we attend to two factors that may affect CPE-PLM's performance: (i) the amount of the data consumed to train a PLM, and (ii) the number of sentences in the validation and test sets. In Table 6.3, we do not notice a clear relationship between the amount of pre-training data and performance. We conjecture this result is rooted in the sampling technique exploited when pre-training XLM-R. Specifically, the technique readjusts the probability of sampling a sentence from each language, increasing the number of tokens sampled from low-resource languages while mitigating the bias towards high-resource languages (Conneau et al., 2020). On the other hand, we discover that the languages for which the size of the validation sets are relatively small (i.e., Hebrew, Polish, and Swedish) tend to benefit from cross-lingual transfer, implying that the insufficient number of examples in the validation set might cause some noise or lead to the suboptimal result in the selection of attention heads.

6.5.2 Visualization of Attention Heads

We revealed in Section 6.4.3 that CPE-PLM's performance for most languages does not suffer much from cross-lingual transfer, suggesting that there would exist significant overlaps among the sets of the attention heads selected for respective languages. To verify our hypothesis, in Figure 6.2, we visualize the language-specific sets of the top 20 heads existing in XLM-R. We observe that the heads effective for CPE-PLM are distributed over the middle-to-upper (6-12) layers of XLM-R, implying that phrase-level information is pervasive in the upper layers rather than the lower ones. In addition, we discover that most of the heads detected as sensitive to syntax respond to multiple languages simultaneously and that there exist a few heads proven to be important for dealing with all the nine languages we consider. Our finding of the existence of such

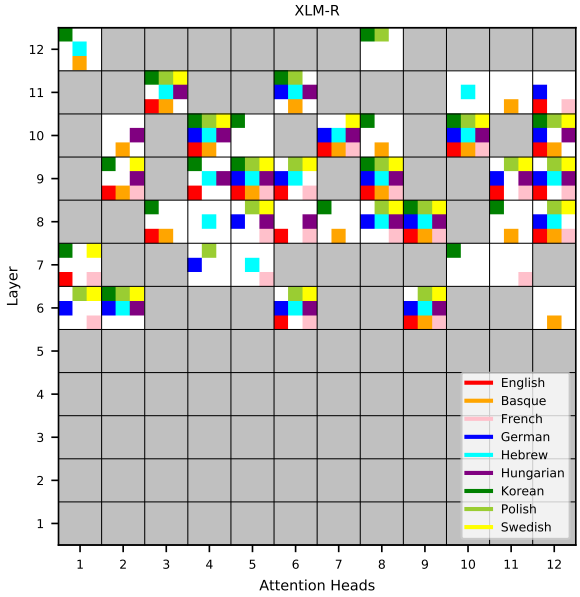

Figure 6.2 Visualization of the sets of the top 20 attention heads (in XLM-R) for 9 languages. Each cell is filled with the color assigned for a language if the corresponding head is responsible for parsing the language.

universal attention heads explains why CPE-PLM is robust to cross-lingual transfer in multilingual settings, in addition to providing a partial clue on why multilingual PLMs are excel at cross-lingual transfer as reported in previous work (Pires et al. (2019); Cao et al. (2020); *inter alia*).

6.5.3 Recall Scores on Noun and Verb Phrases

To assess CPE-PLM's performance in a more fine-grained manner and probe the extent to which it detects the core components of sentences, we present its recall scores on gold-standard noun and verb phrases. We only target the languages whose gold-standard trees contain proper tags in their test sets. In Figure 6.3, we confirm that compared to the random baseline, CPE-PLM has a decent ability to identify noun phrases, succeeding in retrieving more than half of NPs for

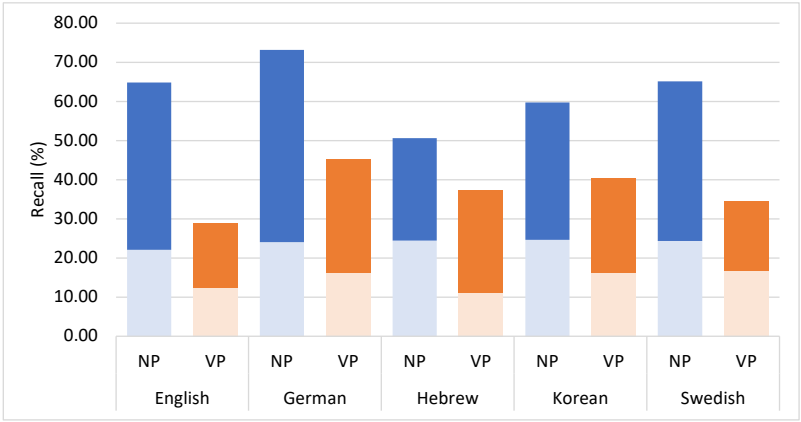

Figure 6.3 Recall scores on gold-standard NPs and VPs. The light bars indicate the random baseline's performance while the dark ones show that of the CC method.

every language. On the contrary, CPE-PLM seems relatively weak in recognizing VPs which are generally longer and more complex than NPs. This implies that CPE-PLM might struggle with grasping the whole structure of sentences (e.g., VPs), although it successfully perceives small phrasal components (e.g., NPs).

6.6 Limitations and Future Work

We here mention a few limitations of our approach and propose avenues for future work. First, analogous to several unsupervised parsers (Shi et al., 2020) including PCFGs (Zhao and Titov, 2021), the current form of our method relies on a few gold-standard annotations from the validation set to determine the best hyperparameters. This dependency makes it hard to say that our approach is entirely unsupervised, although it steps aside from a typical way of learning parsers with supervision. A next, promising yet challenging, step will be therefore to develop a remedy that enables our method to be free from the annotations. Note that our cross-lingual transfer experiments also shed some

light on how to relieve such dependency.

While we have shown that CPE-PLM can be superior or comparable to PCFGs and that it can function as an effective tool for analyzing PLMs, its performance still falls short of expectations in terms of whether it can practically substitute standard parsers, similar to the case of unsupervised parsers. To improve its performance, we have a plan as future work to design an ensemble method that gathers diverse information from heterogeneous PLMs.

6.7 Summary

In this chapter, we have studied the extension of the CPE-PLM framework. To this end, we have introduced a chart-based method and top-K ensemble for improving performance, and extended the range of application of the paradigm to different languages by applying multilingual PLMs. We have also verified that our approach is robust to cross-lingual transfer. Finally, we have provided analyses on the inner workings of our method. For instance, we have discovered universal attention heads which are consistently sensitive to syntactic information irrespective of the input language.

Chapter 7

Conclusion

Sentence representation learning is a fundamental step in natural language processing that defines the way of deriving the integrated meaning of a higher-level concept (i.e., sentence) from its constituents—words and phrases. In this dissertation, we have concentrated on the fact that syntax, which is a subfield in linguistics where the order of combining its elements to construct a sentence is theoretically studied, can suggest a plausible intuition on developing a new method for training and analyzing neural models for sentence representations.

In Chapter 3, we have proposed a neural model, called **SATA Tree-LSTM**, that takes better advantage of explicit supervision from constituency parse trees to derive sentence representations. Its merit comes from the introduction of structure-aware tag representations, which are implemented such that they can reflect the surrounding context of target tags in parse trees.

In Chapter 4, we have assumed that syntactic information is implicitly distributed over the weights of already trained Transformers, blended with other sentence-level knowledge, and have suggested to consider all the intermediate

layers of the Transformers to capture the holistic meaning of an input sentence. To this end, we have introduced a **self-guided contrastive learning** method that fine-tunes those pre-trained Transformers in an self-supervised fashion.

On the other hand, we have presented a system for inducing phrase-structure parse trees from pre-trained language models (PLMs), dubbed as **CPE-PLM** (**C**onstituency **P**arse **E**xtraction from **P**re-trained **L**anguage **M**odels). In Chapter 5, we have introduced one of the instances of the framework called **top-down CPE-PLM**, which utilizes simple top-down parsing mechanism to generate parse trees only sticking to the internal knowledge of PLMs. By analyzing the parse trees derived from our method, we have examined the lower bound of the degree to which PLMs understand syntactic concepts.

In Chapter 6, we have developed a revised version of the CPE-PLM framework, i.e., **chart-based CPE-PLM**, which is reminiscent of chart parsers in the typical parsing literature. With the aid of this approach, we have demonstrated that the syntactic capabilities of PLMs are superior to what was known before. We have also verified that CPE-PLM is applicable for languages other than English, and that there exist universal attention heads in PLMs which are sensitive to the syntactic aspects of input sentences irrespective of their source language.

Although we have partially validated its utility in Chapter 3, it remains as a topic of active research whether explicit modeling of syntax is vital in sentence representation learning, considering the fact in particular that pre-training gigantic models with a colossal number of data guarantees some extent of understanding of syntax as observed in Chapter 5 and Chapter 6. It might not be necessary anymore to explicitly consider syntactic structure in neural network design, especially when it is our top priority to improve the performance of sentence representations in downstream tasks.

Nevertheless, it is worth noting that indeed, syntax-inspired models still have their merits in data and model efficiency. In Chapter 3, we have already demonstrated that tree-LSTMs can achieve performance competitive to that of pre-trained models which require a lot more parameter budgets and training data. This point can be much more attractive when we are in low-resource settings, where it is impossible to collect a sufficient amount of labeled data or even a satisfactory amount of plain text for pre-training. Therefore, as future work, it is desired to broaden our experimental environments into more diverse and extreme cases—beyond common benchmarks—where we may discover more practical advantages of taking syntax into account in network modeling.

On the other hand, even though we did not cover the case, Transformers can also accept additional syntactic knowledge as recursive neural network. For instance, Strubell et al. (2018) have shown that it is beneficial in the case of semantic role labeling to encourage one of the attention heads in Transformer to reflect the syntactic structure of input sentences. Moreover, several other research groups are also actively conducting their research on improving the Transformer architecture relying on syntactic intuitions (Wang et al. (2019c); Bai et al. (2021); *inter alia*). As it is still controversial whether such modifications are crucial in making a remarkable breakthrough in performance improvement, we expect that more effort from the community will be devoted to solving this conundrum.

Finally, we would like to highlight that syntax is an essential ingredient in enhancing the degree of our comprehension with respect to the inner workings of the current form of neural language models which are inherently black boxes. As we begin to rely more on neural models in resolving a number of problems in real-world scenarios, the interpretability of neural networks is becoming increasingly important, particularly in the perspective that we should guarantee

their correctness. It is also critical to master the working mechanism of our neural models in a practical point of view, as this can lead to more effective utilization of them as in Chapter 4. In these regards, syntax-oriented models such as one in Chapter 3—the models that are created to precisely follow syntactic instructions in building their computation graphs—can be appealing, as their inner workings are relatively more transparent compared against those of typical neural architectures. Moreover, as demonstrated in Chapter 5 and Chapter 6, we can utilize syntactic approaches in revealing the extent to which the way a neural model deals with sentences agrees with ours—i.e., the way we process sentences in our brain. To conclude, we anticipate that syntax will function as a core component in bridging research on human language processing and one on language processing of machines.

Bibliography

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *Proceedings of ICLR*.

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2020. Better fine-tuning by reducing representational collapse. *arXiv preprint arXiv:2008.03156*.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings of SemEval*.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of SemEval*.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 task

1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of SemEval*.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of SemEval*.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Proceedings of *SEM*.

Guillaume Alain and Yoshua Bengio. 2016. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Jiangang Bai, Yujing Wang, Yiren Chen, Yaming Yang, Jing Bai, Jing Yu, and Yunhai Tong. 2021. Syntax-bert: Improving pre-trained transformers with syntax trees. In *Proceedings of EACL*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155.

Leonard Bloomfield. 1926. A set of postulates for the science of language. *Language*, 2(3):153–164.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves,

Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation*.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*.

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of ACL*.

Peter F Brown, Vincent J Della Pietra, Peter V Desouza, Jennifer C Lai, and Robert L Mercer. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–480.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of ICML*.

Steven Cao, Nikita Kitaev, and Dan Klein. 2020. Multilingual alignment of contextual word representations. In *Proceedings of ICLR*.

Fredrik Carlsson, Evangelia Gogoulou, Erik Ylipää, Amaru Cuba Gyllensten, and Magnus Sahlgren. 2021. Semantic re-tuning with contrastive tension. In *Proceedings of ICLR*.

Andrew Carnie. 2012. *Syntax: A generative introduction*. John Wiley & Sons.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of SemEval*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Qian Chen, Zhen-Hua Ling, and Xiaodan Zhu. 2018. Enhancing sentence embedding with generalized pooling. In *Proceedings of COLING*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of ICML*.

Xinlei Chen and Kaiming He. 2020. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*.

Zhou Cheng, Chun Yuan, Jiancheng Li, and Haiqin Yang. 2018. Treenet: Learning sentence representations with unconstrained tree structure. In *Proceedings of IJCAI*.

Ethan A. Chi, John Hewitt, and Christopher D. Manning. 2020. Finding universal grammatical relations in multilingual BERT. In *Proceedings of ACL*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *Proceedings of AAAI*.

Noam Chomsky. 1957. *Syntactic structures*. Mouton, The Hague.

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. MIT press.

Noam Chomsky. 1975. *The Logical Structure of Linguistic Theory*. Plenum.

Alexander Clark. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *Proceedings of the 2001 workshop on Computational Natural Language Learning-Volume 7*, page 13.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

John Cocke. 1969. *Programming languages and their compilers: Preliminary notes*. New York University.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An evaluation toolkit for universal sentence representations. In *Proceedings of LREC*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of EMNLP*.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of ACL*.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Proceedings of NeurIPS*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2014. Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *Proceedings of AAAI*.

Andrew Drozdov, Patrick Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised latent tree induction with deep inside-outside recursive auto-encoders. In *Proceedings of NAACL*.

Greg Durrett and Dan Klein. 2015. Neural CRF parsing. In *Proceedings of ACL-IJCNLP*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of NAACL*.

Chris Dyer, Gábor Melis, and Phil Blunsom. 2019. A critical analysis of biased parsers in unsupervised parsing. *arXiv preprint arXiv:1909.09428*.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of ACL*.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of EMNLP-IJCNLP*.

Allyson Ettinger, Ahmed Elgohary, and Philip Resnik. 2016. Probing for semantic evidence of composition by means of simple classification tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*.

Hongchao Fang and Pengtao Xie. 2020. Cert: Contrastive self-supervised learning for language understanding. *arXiv preprint arXiv:2005.12766*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Alan Henderson Gardiner. 1922. The definition of the word and the sentence. *British Journal of Psychology: General Section*, 12(4):352–361.

John M Giorgi, Osvald Nitski, Gary D Bader, and Bo Wang. 2020. Declutr: Deep contrastive learning for unsupervised textual representations. *arXiv preprint arXiv:2006.03659*.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of International Conference on Neural Networks (ICNN)*.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Beliz Gunel, Jingfei Du, Alexis Conneau, and Veselin Stoyanov. 2021. Supervised contrastive learning for pre-trained language model fine-tuning. In *Proceedings of ICLR*.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2019. Visualizing and understanding the effectiveness of bert. *arXiv preprint arXiv:1908.05620*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple customization of recursive neural networks for semantic relation classification. In *Proceedings of EMNLP*.

Serhii Havrylov, Germán Kruszewski, and Armand Joulin. 2019. Cooperative learning of disjoint syntax and semantics. In *Proceedings of NAACL*.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of CVPR*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of ICCV*.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of NAACL*.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of NAACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of ACL*.

Phu Mon Htut, Kyunghyun Cho, and Samuel Bowman. 2018. Grammar induction with neural language models: An unusual replication. In *Proceedings of the EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *Proceedings of ICML*.

Minlie Huang, Qiao Qian, and Xiaoyan Zhu. 2017. Encoding syntactic knowledge in neural networks for sentiment classification. *ACM Transactions on Information Systems (TOIS)*, 35(3):26.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of ICML*.

Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. 2020. A survey on contrastive self-supervised learning. *arXiv preprint arXiv:2011.00362*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of ACL*.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of ACL*.

Dan Jurafsky. 2000. *Speech & language processing*. Pearson Education India.

Katharina Kann, Anhad Mohananey, Samuel R. Bowman, and Kyunghyun Cho. 2019. Neural unsupervised parsing beyond English. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*.

Tadao Kasami. 1966. An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report no. R-257*.

Najoung Kim, Roma Patel, Adam Poliak, Patrick Xia, Alex Wang, Tom McCoy, Ian Tenney, Alexis Ross, Tal Linzen, Benjamin Van Durme, et al. 2019a. Probing what different nlp tasks teach machines about function word comprehension. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (* SEM)*.

Taeuk Kim, Jihun Choi, Daniel Edmiston, Sanghwan Bae, and Sang-goo Lee. 2019b. Dynamic compositionality in recursive neural networks with structure-aware tag representations. In *Proceedings of AAAI*.

Taeuk Kim, Jihun Choi, Daniel Edmiston, and Sang-goo Lee. 2020. Are pre-trained language models aware of phrases? simple but strong baselines for grammar induction. In *Proceedings of ICLR*.

Taeuk Kim, Bowen Li, and Sang-goo Lee. 2021a. Multilingual chart-based constituency parse extraction from pre-trained language models. *arXiv preprint arXiv:2004.13805*.

Taeuk Kim, Kang Min Yoo, and Sang-goo Lee. 2021b. Self-guided contrastive learning for BERT sentence representations. In *Proceedings of ACL-IJCNLP*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*.

Yoon Kim, Chris Dyer, and Alexander Rush. 2019c. Compound probabilistic context-free grammars for grammar induction. In *Proceedings of ACL*.

Yoon Kim, Alexander Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019d. Unsupervised recurrent neural network grammars. In *Proceedings of NAACL*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of NeurIPS*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of ACL*.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL*.

Dan Klein and Christopher D Manning. 2002a. A generative constituent-context model for improved grammar induction. In *Proceedings of ACL*.

Dan Klein and Christopher D. Manning. 2002b. A generative constituent-context model for improved grammar induction. In *Proceedings of ACL*.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*.

Reinhard Kneser and Hermann Ney. 1993. Improved clustering techniques for class-based statistical language modelling. In *Proceedings of Third European Conference on Speech Communication and Technology*.

Karim Lari and Steve J Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56.

Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. 2020. Contrastive representation learning: A framework and review. *IEEE Access*.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. In *Proceedings of EMNLP*.

Bowen Li, Lili Mou, and Frank Keller. 2019. An imitation learning approach to unsupervised parsing. In *Proceedings of ACL*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of COLING*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *Proceedings of NAACL*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017a. Adaptive semantic compositionality for sentence modelling. In *Proceedings of IJCAI*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017b. Dynamic compositional neural networks over tree structure. In *Proceedings of IJCAI*.

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

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *Proceedings of ICLR*.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*.

Jean Maillard, Stephen Clark, and Dani Yogatama. 2019. Jointly learning sentence embeddings and syntax with unsupervised tree-lstms. *Natural Language Engineering*, 25(4):433–449.

Christopher Manning and Hinrich Schutze. 1999. *Foundations of statistical natural language processing*. MIT press.

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

David Mareček and Rudolf Rosa. 2018. Extracting syntactic trees from transformer encoder self-attentions. In *Proceedings of the EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

David Mareček and Rudolf Rosa. 2019. From balustrades to pierre vinken: Looking for syntax in transformer self-attentions. In *Proceedings of the ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC*.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah, and Benoît Sagot. 2019. Camembert: a tasty french language model. *arXiv preprint arXiv:1911.03894*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proceedings of NeurIPS*.

Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. *Proceedings of NeurIPS*.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-

supervised learning. *IEEE transactions on pattern analysis and machine intelligence.*

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. *arXiv preprint arXiv:2006.04884.*

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of ACL.*

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. In *Proceedings of EMNLP.*

Tsendsuren Munkhdalai and Hong Yu. 2017. Neural semantic encoders. In *Proceedings of ACL.*

Nikita Nangia and Samuel Bowman. 2018. Listops: A diagnostic dataset for latent tree learning. In *Proceedings of NAACL: Student Research Workshop.*

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook fair's wmt19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation.*

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP (RepEval).*

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL.*

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of LREC*.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of ACL*.

Jordan B Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46(1-2):77–105.

Qiao Qian, Bo Tian, Minlie Huang, Yang Liu, Xuan Zhu, and Xiaoyan Zhu. 2015. Learning tag embeddings and tag-specific composition functions in recursive neural network. In *Proceedings of ACL*.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Alessandro Raganato and Jörg Tiedemann. 2018. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of EMNLP-IJCNLP*.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Rudolf Rosa and David Mareček. 2019. Inducing syntactic trees from bert representations. *arXiv preprint arXiv:1906.11511*.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages (SPMRL)*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of ACL*.

Tao Shen, Zhou Tianyi, Long Guodong, Jiang Jing, Wang Sen, and Zhang Chengqi. 2018a. Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling. *arXiv preprint arXiv:1801.10296*.

Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. 2018b. Neural language modeling by jointly learning syntax and lexicon. In *Proceedings of ICLR*.

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron Courville, and Yoshua Bengio. 2018c. Straight to the tree: Constituency parsing with neural syntactic distance. In *Proceedings of ACL*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *Proceedings of ICLR*.

Haoyue Shi, Karen Livescu, and Kevin Gimpel. 2020. On the role of supervision in unsupervised constituency parsing. In *Proceedings of EMNLP*.

Haoyue Shi, Jiayuan Mao, Kevin Gimpel, and Karen Livescu. 2019. Visually grounded neural syntax acquisition. In *Proceedings of ACL*.

Richard Socher, John Bauer, Christopher D Manning, et al. 2013a. Parsing with compositional vector grammars. In *Proceedings of ACL*.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y Ng, and Christopher D Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of ICML*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of ACL*.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NeurIPS*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*.

Zhiyang Teng and Yue Zhang. 2017. Head-lexicalized bidirectional tree lstms. *TACL*, 5:163–177.

Junfeng Tian, Zhiheng Zhou, Man Lan, and Yuanbin Wu. 2017. ECNU at SemEval-2017 task 1: Leverage kernel-based traditional NLP features and neural networks to build a universal model for multilingual and cross-lingual semantic textual similarity. In *Proceedings of SemEval*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NeurIPS*.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of ACL*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Proceedings of NeurIPS*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of ICLR*.

Bin Wang and C-C Jay Kuo. 2020. Sbert-wk: A sentence embedding method by dissecting bert-based word models. *arXiv preprint arXiv:2002.06652*.

Kexin Wang, Nils Reimers, and Iryna Gurevych. 2021. Tsdae: Using transformer-based sequential denoising auto-encoder for unsupervised sentence embedding learning. *arXiv preprint arXiv:2104.06979*.

Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019c. Tree transformer: Integrating tree structures into self-attention. In *Proceedings of EMNLP*.

Yizhong Wang, Sujian Li, Jingfeng Yang, Xu Sun, and Houfeng Wang. 2017. Tag-enhanced tree-structured neural networks for implicit discourse relation classification. In *Proceedings of IJCNLP*.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. 2018a. Do latent tree learning models identify meaningful structure in sentences? *TACL*, 6:253–267.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018b. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. 2020. Clear: Contrastive learning for sentence representation. *arXiv preprint arXiv:2012.15466*.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. In *Proceedings of NeurIPS*.

Baosong Yang, Derek F Wong, Tong Xiao, Lidia S Chao, and Jingbo Zhu. 2017. Towards bidirectional hierarchical representations for attention-based neural machine translation. In *Proceedings of EMNLP*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Proceedings of NeurIPS*.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang

Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *Proceedings of ICLR*.

Daniel H Younger. 1967. Recognition and parsing of context-free languages in time n3. *Information and control*, pages 189–208.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *Proceedings of IJCAI*.

Yanpeng Zhao and Ivan Titov. 2021. An empirical study of compound pcfgs. *arXiv preprint arXiv:2103.02298*.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *Proceedings of COLING*.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. Freelb: Enhanced adversarial training for natural language understanding. In *Proceedings of ICLR*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of ICCV*.

초록

구문론(syntax)은 언어학의 한 갈래로써, 자연어 문장의 형성 과정에 내포되어 있는 원리와 그로 인해 촉발되는 여러 언어적 현상을 규정하고 이를 검증하는 연구 분야를 총칭한다. 구문론은 단어, 구 및 절과 같은 문장 내의 구성 요소로부터 해당 문장의 의미를 점진적으로 구축해 나가는 과정에 대한 체계적인 이론적 절차를 제공하며, 따라서 이는 자연어처리에서 문장 표현 학습 및 분석을 위한 방법론을 구상하는데 있어 활용될 수 있는 잠재성을 지니고 있다.

본 논문에서는 신경망 기반의 문장 표현 방법론을 개발하는 데 있어 구문론을 활용하는 두 측면에 관하여 논한다. 먼저, 언어학적인 파스 트리의 형태로 표현되어 있거나 혹은 타 신경망 모델의 파라미터에 암시적으로 저장되어 있는 구문론적 지식을 도입하여 더 나은 문장 표현을 만드는 보다 직접적인 방법론을 제시한다. 이에 더하여, 구문론에 바탕한 문법적 체계를 이용하여 학습이 완료된 신경망 기반 문장 표현 모델들의 작동 원리를 규명하고 이들의 개선점을 찾는데 도움을 줄 수 있는 분석적 접근법 또한 소개한다. 실제 환경에서의 다각적인 실험과 검증을 통하여 규칙 및 통계 기반 자연어처리에서 귀중한 자원으로 간주되었던 구문론이 신경망 기반의 모델이 대중적으로 사용되고 있는 현재의 자연어처리에서도 보완재로써 기능할 수 있음을 보인다. 구체적으로, 구문론이 고성능의 문장 표현을 위한 신경망 모델 혹은 이를 위한 학습 방법론을 개발하는데 있어 효과적인 직관을 제공할 수 있음을 실증하고, 문장 표현 신경망 모델이 자체적으로 파스 트리를 복원해낼 수 있는 능력을 평가함으로써 구문론을 내부 작동 체계가 불명확한 신경망 모델의 작동 원리에 대한 이해도를 증진시키는 분석 도구로 활용한다.

주요어: 자연어처리, 기계 및 심층 학습, 신경망, 문장 표현, 구문론, 비지도 파싱, 문장 분류, 문장 유사도 측정, 재귀신경망, 트랜스포머, 사전학습 언어모델
학번: 2016-21197