Co-training a Weakly Supervised Constituency Parser of Natural Language

Nickil Maveli

Master of Science by Research

Institute for Language, Cognition and Computation

School of Informatics

University of Edinburgh

2021

Abstract

We develop a method for unsupervised parsing that depends on bootstrapping classifiers: an *inside* classifier that operates on a span, and an *outside* classifier that operates on everything excluding the given span, to determine if a node dominates a specific span in a sentence. To effectively learn the 2-way interactions between these classifiers, we subject the two classifiers through self-training and co-training procedures that help the parser adeptly capture the span boundaries of likely constituents and distituents in a sentence. In addition, through the paradigm of weak supervision, we inject a strong inductive bias into our parser primarily based on two modes of supervision: (1) prior branching knowledge of a known language (left/right-branching) to train these classifiers using a novel seed bootstrapping technique, and (2) rule-based heuristics, to achieve 63.1 F_1 on the English (PTB) test set. Furthermore, we demonstrate the generalization capabilities of our parsing framework by evaluating on treebanks for Chinese (CTB) and Japanese (KTB) using simply mode (1) as weak supervision to set new state-of-the-art results. Our code is available at https://github.com/nickil21/weakly-supervised-parsing.

Acknowledgements

It feels surreal to complete my thesis, from start to finish, in the middle of the Covid-19 pandemic. There have been numerous people who helped me along the way on this journey. I want to take this opportunity to thank them.

First and foremost, I am immensely thankful to my advisor, Prof. Shay Cohen, for making this thesis a reality through his constant guidance. If it hadn't been for him, who agreed to take me in his group as an MScR student, I probably wouldn't be researching to this day. He knew precisely when and how to either breathe down my neck or give me the freedom to explore at my own pace. Further, I appreciate his kind gesture to offer support when I was looking for a job as an applied researcher towards the end of my studies.

Prior to starting my studies at Edinburgh, I was an NLP software developer at Niki, my first real-world exposure to building NLP products. I am grateful to Ananth for the invaluable mentorship I received, which helped shape my critical thinking skills useful for carrying out research.

More broadly, I like to thank the members of the Cohort with whom I had the most interactions: Zheng Zhao, Marcio Fonseca, Ronald Cardenas, Esma Balkır, Javad Hosseini, Chunchuan Lyu, Jiangming Liu, Waylon Li, Haoran (Gavin) Peng, Yichao Liang, Yifu Qiu, and Yftah Ziser, for being a fantastic group and making our Monday reading group meetings a joy to attend. There was always something new to learn from the discussions and an opportunity for me to give presentations as well. Many thanks to EdinburghNLP for hosting great seminars and inviting distinguished speakers from time to time. Special thanks to Lexi Birch for agreeing to review and provide feedback on my thesis. I am also grateful to my committee members: Ankur Parikh (external) and Frank Keller (internal), for examining my thesis and providing constructive feedback.

Finally, I want to thank my parents for instilling a strong passion for learning since childhood and doing anything and everything to put me on the right path. I also want to thank my teachers, especially in my formative years at school, for helping me build up the character and foundations needed to succeed.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Nickil Maveli

(*Nickil Maveli*)

To my parents, for their unconditional love and constant support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Human beings communicate using language in the form of a sequence of words canonically vocalized as speech. They typically learn by observing their surroundings and innately acquire language properties, such as the acquisition of word meaning, phonology, morphology, and even syntax. Figure 1.1 shows the parse diagram under a set of conventions for the sentence, *The boy who was sitting at the edge of his chair fell down*, which is reasonably straightforward for most humans to understand; however, it is challenging for many Natural Language Understanding (NLU) systems to comprehend successfully, such as, to construe additional links (coreference between mentions) and the underlying relationship between different words in the sentence. However, in the field of Natural Language Processing (NLP), human language experts create syntactically annotated sentences to primarily tackle the problem of structure induction using constituency parsing. Parse trees (resulting from parsing) have found their usage as an intermediate representation in several downstream applications:

- **Named Entity Recognition:** To identify the named entities in the sentence, *were allies of the European Football Union with ties to John Doe*, the parser, after correctly identifying the phrase *European Football Union* in the constituency tree can use the surrounding context to infer that it belongs to an organization and not a person (Finkel and Manning, 2009).

- **Machine Translation:** To translate a sentence from a source to a target language, the operations on the intermediate parse trees can help learn specific productions of grammar and word-order variations to reconstruct the parse of the target language (Wu and Wong, 1998).

- **Speech Recognition:** To recognize the transcription, *place the card in the drawer*

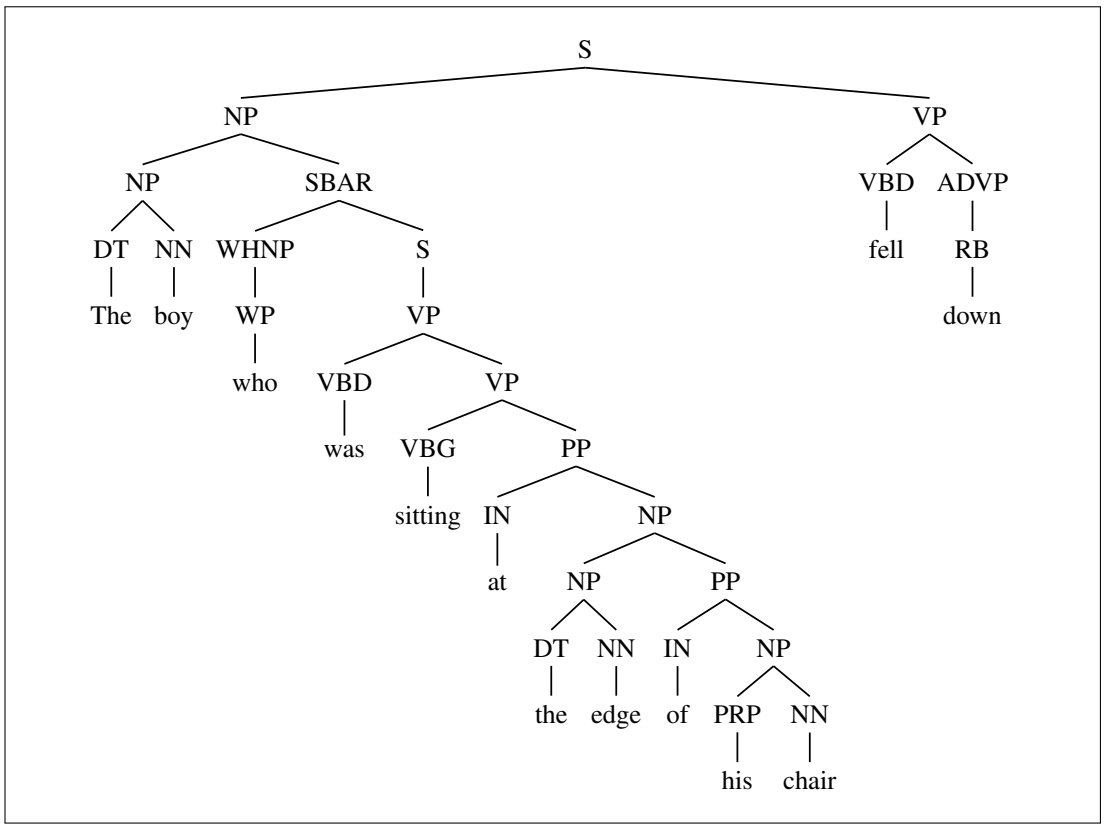

Figure 1.1: A sample parse tree representation. A listener must comprehend the situation in which the chair modifies the boy in a relative clause and decipher that the boy has fallen rather than the linearly proximate chair.

and give a higher probability (based on every joint sequence of words) than a less likely transcription *place the card and the drawer* (Chelba and Jelinek, 1998).

- **Question Answering:** To answer the question: *Which trains to Edinburgh arrive before the Manchester train?*, there is a need first to understand that the questioner wants a list of trains going to Edinburgh, not trains going to Manchester. Subsequently, the parse structure (knowing that *to Edinburgh* modifies trains and *which trains to Edinburgh* is the subject of the arrival) can help us (Shen et al., 2005).

Parsing, although having a myriad of practical uses, is not an unchallenging task. A common problem a syntactic parser faces is structural ambiguity, which occurs when the grammar tends to assign multiple parses to a sentence. Two common forms of ambiguity are attachment ambiguity (attach a constituent to the parse tree at more than one place) and coordination ambiguity (conjoining of a phrase by conjunction). Typical

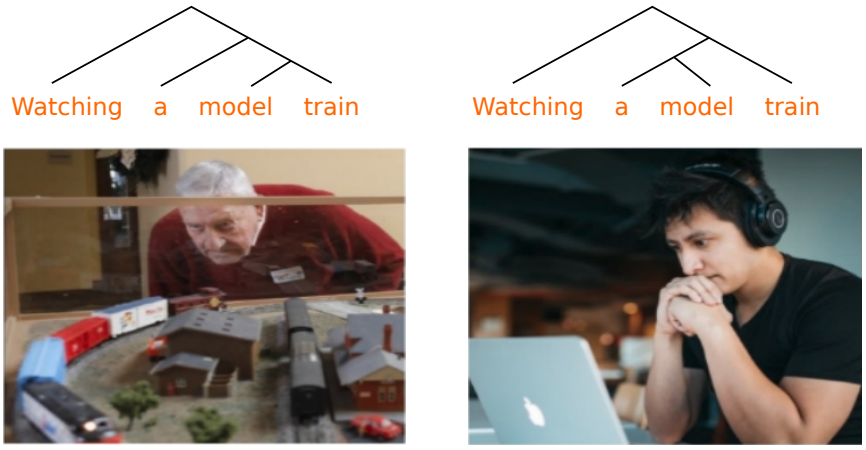

Figure 1.2: Two parse trees for a sentence having structural ambiguity that requires context to resolve where only one of the readings is plausible, and it is the parser's job to determine which one. The parse on the left corresponds to the reading in which *model train* (noun) refers to the toy train. The parse on the right corresponds to the reading in which *model train* denotes the model artifact, common in the machine learning terminology, subjected to a training process (verb). Image courtesy: https://twitter.com/wluper_/status/1133725988368637953.

ambiguities originating from coordination, relative clause attachment, PP attachment, etc., can result in alternative possibilities which tend to multiply when chained together. Figure 1.2 illustrates a scenario that is impossible to disambiguate for both humans and machines without additional contextual information. At times, it can also involve factors such as the possession of world knowledge and general reasoning capabilities which makes disambiguation much harder.

1.1 Motivation

While an effective solution, parsing does not provide a sound intuition about how children would address the phenomenon of inducing structure from observed data alone. Recently, substantial progress has been achieved in the field of unsupervised parsing, i.e., inducing parse tree structures from observed sentences alone without supervision. Consequently, there is a strong desire to study unsupervised parsing. Although constituency parsing has been an integral part of nearly every NLP system, yet the current supervised parsers only operate on the commonly spoken languages and are incapable of generalizing to out-of-domain data. In addition, the annotation of syntactic

tree structures by human language experts is both expensive and time-consuming. At times, in the case of low-resource languages, no proper annotation scheme is available. Hence, the study of unsupervised parsing and grammar induction is an important endeavor in the right direction. From a scientific standpoint, success in this field can bring machines closer to the reasoning of how children process textual information to infer syntactic knowledge. Furthermore, from a psycho-linguistic standpoint, the advancements in unsupervised parsing and grammar induction can have a pivotal role to play in supplementing empirical evidence against innate linguistic theories and arguments such as *poverty of the stimulus* (Chomsky et al., 2006) and *universal grammar hypothesis* (White, 1990).

In this thesis, we focus on the task of **unsupervised constituency parsing**. Broadly, we pose the problem of identifying the constituents, i.e., spans (including single-word and multi-word) that function as a single unit, and distituents, i.e., spans that do not belong to constituents, in a sentence as a binary classification task. First, we devise a procedure to convert the unlabeled sentences into a sequence classification task and then train a model by fine-tuning a transformer-based pre-trained language model on the unlabeled sentences to differentiate between the true and false inside strings of the pseudo-constituents. Later, we utilize the highly-confident inside strings to generate the outside strings. Finally, using self-training and co-training approaches, we learn by optimizing and integrating both the inside and outside string passes to enhance the model's ability to recognize the breakpoints in a sentence. Our best parser attains 63.1 sentence F_1 averaged over multiple runs with random seed on the Penn Treebank (PTB) test set using the branching knowledge and minimal heuristics acting as weaker forms of supervision. In addition, we report strong results on the test sets for the Japanese Treebank (KTB) as well as the Chinese Treebank (CTB).

1.2 Thesis Outline

In Chapter 2, we commence by providing an outline of the history and recent developments in the field of unsupervised parsing. Next, we formally delineate the problem formulation and describe different evaluation methods. Later, we state the possible utilization of few shot parsing as a middle-ground between fully unsupervised and supervised systems. We then argue the need to follow a set of universal conventions while designing parsers. Lastly, we briefly discuss the distinction between unsupervised parsing and grammar induction.

In Chapter 3, we introduce our neural unsupervised parsing framework and describe its training algorithm, fundamentally having three stages in an increasing order of complexity: learning using the inside strings; learning using the inside and outside strings; and a joint co-training based model training. Lastly, we describe the inference algorithm to extract the maximum scoring parse tree.

In Chapter 4, we discuss the implementation of our multi-view learning strategy. In particular, we elucidate the seed bootstrapping process to pose the challenge of inducing syntactic structures as a supervised classification task guided by pseudo-labels. Additionally, we outline the benefit of adding heuristics to the overall system to achieve higher gains. Lastly, we explain the training procedure behind the inside string, outside string, and the joint inside and outside strings.

In Chapter 5, we present our experimental results on treebanks for languages of different branching tendencies. We show strong results on right-branching languages such as English (PTB) and Chinese (CTB), as well as left-branching languages such as Japanese (KTB). Furthermore, we conduct linguistic error analysis to identify some of the more frequent errors made by the parser and provide linguistically motivated explanations. Lastly, we wrap up with a discussion about some of the limitations of our approach.

We will finally conclude in Chapter 6 as well as suggest potential directions for future work.

1.3 Contributions

The contributions of this thesis are summarized as follows:

- We devise a strategy to pose the unsupervised learning as a synthetically created sequence classification task that can generalize to any language parameterize by the direction of branching (left/right) using weak supervision.
- We set the groundwork for the research direction of employing semi-supervised learning techniques, i.e., self-training and co-training, as a core component of unsupervised parsing to model interactions between the inside and outside classifiers.

- We make an effort to identify common parsing errors and propose further corrective strategies to address these errors. Finally, our approach sets a new state-of-the-art result on three treebanks comprising both right-branching and left-branching languages.

Chapter 2

Background

In this chapter, we aim to provide the readers with an overview of unsupervised parsing. First, we describe some of the early works, to the more recent resurgence of neural network-based (deep learning) approaches in Section 2.1. We then formally define the unsupervised parsing task under both generative as well as discriminative settings in Section 2.2 and describe the evaluation metrics that typically previous works have used for this task. Later, we suggest a better alternative to perform training in the few-shot learning setting (Section 2.3).[1] We next argue for the strong requirement of following a standard protocol for all unsupervised parsers to make meaningful evaluations and improve model explainability from a futuristic perspective (Section 2.4). Finally, we explain in Section 2.5, that unsupervised parsing and grammar induction, although closely related to one another, are not synonymous.

2.1 History

In this section, we review a series of previous approaches to address the problem of unsupervised parsing — ranging from the early usage of statistical methods to the more recent employment of neural network-based methods.

2.1.1 Early Approaches

In the early 1990s, much effort had been spent on inducing probabilistic context-free grammars (PCFGs) via the Expectation-Maximization (EM) algorithm (Lari and Young,

[1] Few-shot learning learns models given only a few labeled examples.

1990; Carroll and Charniak, 1992).[2] During this time, the conventional wisdom pointed towards EM not being suitable for inducing tree structures as it produced unsatisfactory results. The Constituent-Context Model (CCM) of Klein and Manning (2002) assigned scores to each parse as a product of span probabilities and its contextual subsequences. It was the first model that outperformed the right-branching baseline, however, only unpunctuated sentences of sentence length upto 10 were considered during evaluation. In particular, the CCM model of Klein and Manning (2002) was the backbone for much subsequent works (Klein and Manning, 2004; Huang et al., 2012; Golland et al., 2012).

A non-parametric model was suggested by Bod (2006) known as Unsupervised Data-Oriented Parsing (UDOP) which used a large random subset of subtrees from all possible binary trees to estimate the most likely parse. Seginer (2007) adopted an incremental and a (locally) greedy parsing strategy which used a new link representation to create constituents. Ponvert et al. (2011) combined finite-state models in a cascaded fashion to produce constituent structures.

Few models exploited the availability of parallel corpora in multiple resource rich languages to perform unsupervised POS induction (Cohen et al., 2011; Das and Petrov, 2011). Parikh et al. (2014) proposed a spectral learning algorithm based on additive tree metrics and achieved improved results compared to the CCM model (Klein and Manning, 2002) on longer sentences (length > 10) without much initialization.

2.1.2 Neural Network based: The Deep Learning Era

Over the last couple of years, the resurgence of deep learning has significantly changed the landscape of unsupervised parsing. In comparison to the traditional, statistical, and hand-designed feature-based models, the neural models have shown to be effective in learning rich linguistic structures capable of outperforming the previously established benchmarks by a substantial margin. Applications employing neural networks often require less expert analysis and fine-tuning since they are trained rather than programmed. Moreover, they can exploit the tremendous amount of textual data widely available and can be re-trained using a custom dataset in contrast to traditional systems which tend to be more domain-specific. Numerous approaches have been proposed using neural networks which showed promising results on latent tree induction directly from words.

[2] In this approach, the grammar structure is fixed and only parameters are induced depending on the inclusion or exclusion of rule-based queries.

2.1.2.1 Recurrent Neural Network based Approaches

By injecting inductive bias into the recurrent neural network (RNN), Parsing-Reading-Predict Network (PRPN) model proposed by Shen et al. (2018b) attempted to induce trees by solving a language modeling task. The parsing network computed the syntactic distance (Shen et al., 2018a) to make soft constituent decisions. The previous state was composed using self-attention and the attention range was regulated by the syntactic distance, which was equivalent to the depth of the tree. In a follow up work, Shen et al. (2019) proposed ordered neurons LSTM (ON-LSTM) model to induce a hierarchy in the representation units that used a combination of gating mechanism and an activation function to recursively find breakpoints based on each neuron's updates and all the neurons that follow it sequentially. The early successful works on unsupervised parsing, for instance, conducted by Klein and Manning (2002), enforced strong conditional independence assumptions that enabled the model to discover desirable tree structures, but it came at the expense of language modeling performance. A new set of generative models, recurrent neural network grammars (RNNG) model (Dyer et al., 2016) made no independence assumptions. Instead, an RNN model encoded structural bias via shift and reduce operations after learning the joint distribution over sentences and parse trees. Even though the language modeling capabilities were improved substantially as a result of no independence assumption, the grammar induction abilities were not promising. To overcome this challenge, the unsupervised RNNG (URNNG) model introduced by Kim et al. (2019b) employed parameterized function over latent trees to handle intractable marginalization and injected strong inductive biases for the unsupervised learning of the RNNG model. A hybrid model, Parser and Language Model (PaLM; Peng et al. 2019) consisted of a RNN language model with a constituency parser that learned syntactical information implicitly using attention mechanisms over the spans of tokens to create a span-based parser from the attention weights.

2.1.2.2 PCFG with Neural Parameterization

Recent work (Jin et al., 2018a,b, 2019; Kim et al., 2019a; Zhu et al., 2020), showed that inducing PCFGs from raw text is possible if it used neural parameterization, i.e., utilizing neural networks to produce the rule probabilities. For instance, Jin et al. (2019) showed that a regularized PCFG inducer with a normalizing flow model (Dinh et al., 2015) could produce meaningful tree structures if it used contextual embeddings. Compound PCFG (Kim et al., 2019a) utilized neural networks to parameterize the

PCFG's rule probabilities and found that the neural PCFG is capable of inducing trees using maximum likelihood estimation. The work of Kim et al. (2019a) is related to some degree of extending PCFGs to the latent variable setting (Matsuzaki et al., 2005; Petrov et al., 2006; Cohen et al., 2012). However, these approaches had relied on annotated treebanks for learning supervised parsers. To induce both constituents and dependencies within a single model, Zhu et al. (2020) proposed neural lexicalized PCFGs (L-PCFGs) that ameliorated the data sparsity problem through parameter sharing. To reduce the computational complexity from cubic to quadratic, Yang et al. (2021) extended PCFG based on tensor decomposition.

2.1.2.3 Leveraging Transformer Models

More recently, Pre-trained Language Models (PLMs) have shown tremendous success in learning universal language representations from large-scale raw texts, which are beneficial for downstream NLP tasks (Radford et al., 2019; Devlin et al., 2019; Dong et al., 2019). One such deep and powerful architecture is the Transformer (Vaswani et al., 2017). To comprehend language hierarchically, Wang et al. (2019) proposed Tree Transformer, which improved the interpretability of the attention heads of transformer encoder capable of inducing tree structures guided by a self-attention module. In another study, Kim et al. (2020) extracted trees from pre-trained transformers without training and found that the attention heads exhibited syntactic structure that resembled constituency grammar. Li et al. (2020a) built a purely unsupervised parser using attention heads of transformers that do no rely on the development set. Shen et al. (2021) proposed StructFormer model, that introduced strong inductive bias to enable transformers to perform unsupervised dependency and constituency parsing at once.

2.1.2.4 Guiding by Weak Supervision

The main idea surrounding weak supervision is the application of labelling functions (such as linguistic constraints, gazetteers, external databases, etc.) as an alternative to the gold-standard annotations. To induce syntactic structures from BERT-based models using distant supervision data, Shi et al. (2021) extracted naturally-occurring bracketings such as answer fragments and webpage hyperlinks. Their analysis re-asserted Spitkovsky et al.'s (2010) findings that a significant portion of HTML markups do conform to constituents. Based on the weak supervision inferred from the linguistic notion of constituency tests and further refinement of the transformer model's

grammaticality judgments, Cao et al. (2020) were able to achieve strong results.

2.1.2.5 Combining Autoencoders with the Inside-outside Algorithm

The goal of standard autoencoders is to map a sentence to lower-dimensional representation (encoding), which is an intermediate stage, and then reconstruct the observed sentence from the decompressed intermediate representation (decoding). The Deep Inside-Outside Recursive Autoencoder (DIORA) of Drozdov et al. (2019) incorporated the inside-outside algorithm (Baker, 1979), and with the help of latent tree chart parsers, trained to predict each word from its surrounding context. During the bottom-up inside pass, the inside representations corresponding to each span of the input sentence was responsible in encoding only the current subtree. Whereas, during the top-down outside pass, the outside representations was responsible in encoding the context of a given subtree. To recover from local errors of DIORA, (Drozdov et al., 2020) proposed an improved variant, S-DIORA with majorly two modifications. The first modification was to replace DIORA's weights by performing a hard argmax operation to encode a single tree and thereby eliminating the vector averaging altogether. The second modification was to add a beam at each cell in the chart to mitigate the effect of local errors arising when using the context-free approach of the inside-outside algorithm. Since DIORA's training objective used only leaf-level spans, Hong et al. (2020) experimented with all-level spans training objective function and found better performance.

2.1.2.6 Reinforcement Learning through Downstream Tasks

Some approaches used reinforcement learning to induce syntactic structures using reward functions defined by downstream tasks. Yogatama et al. (2017) utilized reinforcement learning and took the performance on a downstream task as the reward signal to induce tree structures. Choi et al. (2018) proposed Gumbel Tree-LSTM that learned to compose tree structures from raw sentences. To combine the continuous PRPN (Shen et al., 2018b) with a Tree-LSTM model (Tai et al., 2015) having discrete parsing abilities, Li et al. (2019) adopted an imitation learning framework.

2.1.3 Multi-Modal Grammar Induction

The approaches mentioned in Section 2.1.1 as well as Section 2.1.2 were generally limited to relying only on textual elements and do not consider signals from other modalities of data. In this section, we offer methods aimed to induce syntactical

structures aided by its visual context. In this regard, Shi et al. (2019) proposed Visually Grounded Neural Syntax Learner (VG-NSL) that learned a parser by looking at natural images and reading aligned captions. They assumed that similar spans of phrases, for instance, verb or prepositional phrases, could be mapped to similar visual objects and these concrete spans in itself should act as constituents. Although it was able to accurately model the phrase structure syntax, it was observed that the parser focused more on shorter constituents such as Noun Phrases and performed poorly on longer ones such as Verb Phrases when compared to text-only parsing approaches. To overcome this drawback, Zhao and Titov (2020) introduced a fully-differentiable end-to-end visually grounded learning framework which was optimized through a language modeling objective by extending the Compound PCFG (Kim et al., 2019a) model. In another study, Kojima et al. (2020) constructed simpler model variants of the VG-NSL (Shi et al., 2019) and identified key components to reduce the expressivity of the model. Contrary to what was expected, they found that the less expressive variants performed equally or sometimes even better in comparison to the original VG-NSL model. In the same vein, Jin and Schuler (2020) incorporated visual information from images and achieved strong performance on multilingual induction datasets without any additional encoded information. Since images are static and lacked dynamic information about visual elements, Zhang et al. (2021) investigated video-aided grammar induction by leveraging aligned video-sentence pairs and noticed that their model was able to learn linguistic phenomena for verb related features effectively.

2.2 Task Definition

2.2.1 Problem Formulation

We first formally describe the task. Let the inputs be a sentence $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$, where T is its length in words, which will induce an unobserved structure $\mathbf{z} \in \mathcal{Z}_T$ (e.g. parse tree, part-of-speech sequence) over a sequence of length T. We compare generative and discriminative unsupervised structure learning models. In the generative learning setting, the models are optimized to maximize the log marginal likelihood over sentences and the parse trees: $\log p(\mathbf{x}) = \log \sum_{\mathbf{z} \in \mathcal{Z}_T} p(\mathbf{x}, \mathbf{z})$. Whereas, in the discriminative learning setting, the models are optimized to maximize the conditional probability of the parse trees given the sentences: $\sum_{\mathbf{z} \in \mathcal{Z}_T} \log p(\mathbf{z} \mid \mathbf{x})$.

2.2.2 Evaluation

Setting up a right evaluation framework towards unsupervised parsing is challenging as the linguistically motivated gold-standard trees could have syntactic analysis for syntactic constructions which is questionable. However, it has the notable advantage of providing hard empirical quantitative information and is, therefore, one of the better-known ways of evaluating trees. As opposed to supervised parsing, unsupervised parsers do not actually label the brackets in the tree since there is no way to match the induced symbols to the gold symbols using some correspondence linking. As a result, we only measure the unlabeled brackets. We can then define the following metrics:

- **True Positives** (TP) are the spans present in both predicted and gold-standard parse tree.
- **True Negatives** (TN) are the spans not present in both predicted and gold-standard parse tree.
- **False Positives** (FP) are the spans present in the predicted but not present in the gold-standard parse tree.
- **False Negatives** (FN) are the spans present in the gold-standard but not present in the predicted parse tree.

The standard formula for Precision, which is a proxy for the percentage of predicted constituents that are correct, can be written as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}.$$

The standard formula for Recall, which is a proxy for the percentage of gold constituents that are predicted, can be written as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}.$$

As a unification of these two quantities, we also report the unlabeled F_1, their harmonic mean:

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

However, there are two ways of averaging F_1 scores over multiple unseen test sentences, i.e., micro and macro average. In micro average (i.e., corpus-level score), we

aggregate all the span predictions at a corpus level and compare them with gold-standard spans to get the TP, FP, and FN values. Later, we compute the Precision and Recall and finally the resulting F_1 score. On the other hand, in macro average (i.e., sentence-level score), we compute the F_1 score at a sentence level (incorporating all its spans) and then average it over all the test sentences.

In addition, we make use of the standard PARSEVAL metrics (Black et al., 1991) whose canonical implementation is implemented by EVALB.[3] There does exist a slight difference in the way the F_1 score is calculated in relation to the standard and EVALB approaches. In the former case, we typically remove trivial (single-word and whole-sentence) spans and count duplicate spans (resulting from unary production) only once. Whereas, in the latter case, we do take into consideration the whole-sentence spans as well as the duplicated spans and do not discard them. Our experiments use the `evalb` parameters as shown in Table 2.1.

```
DEBUG 0
MAX_ERROR 1
CUTOFF_LEN 10
LABELED 0
DELETE_LABEL_FOR_LENGTH -NONE-
EQ_LABEL ADVP PRT
```

Table 2.1: The hyperparameters used for `evalb`

.

To estimate the perplexity of a language model used for unsupervised parsing more holistically, we perform syntactic evaluation by evaluating the grammaticality of the predictions (Marvin and Linzen, 2018); the model is given minimally different pairs of sentences, one grammatical and one ungrammatical, and must, in theory, expect to assign a higher probability score to the grammatical sentence. For e.g.:

The bankers embarrassed themselves.
*The bankers embarrassed herself.

In the above minimal pair, the last sentence is ungrammatical because the reflexive pronoun (herself) needs to agree in number (and gender) with its antecedent.

To determine if the unsupervised parsers can generalize well on both short and long sentences, we generally divide the test sentences into two sets: sentences having

[3] https://nlp.cs.nyu.edu/evalb/

length $\leq$ 10 (WSJ-10) and all sentences (WSJ-Full). We mainly follow the guidelines delineated by Li et al. (2020b). We report scores on both these sets. In order to avoid overfitting to a particular random seed, it is preferred to report mean and standard deviation based on multiple random seeds or restarts. In addition, a lot of previous work which recursively split larger constituents into smaller ones based on the notion of syntactic distance (Shen et al., 2018b, 2019; Htut et al., 2018; Li et al., 2019; Shi et al., 2019) have shown a marked bias for outputting right-branching structures, thereby inflating parsing performance on right-branching languages such as English (Dyer et al., 2019). Consequently, it is recommended to perform evaluation on treebanks of both right-branching and left-branching languages, such as the CTB and KTB respectively, in addition to the default, PTB.

2.3 Few-Shot Parsing: A Better Alternative

Recently proposed neural unsupervised parsing approaches consider the gold parse trees in the development set for either early stopping (Shen et al., 2018b, 2019; Drozdov et al., 2019, 2020) or hyperparameter tuning (Kim et al., 2019a). Based on a study conducted by Shi et al. (2020), they found that the size of the labeled samples used for tuning is indeed responsible for the strong performance as opposed to the fully unsupervised criteria. To be specific, about 15 labeled samples are sufficiently capable of achieving a comparable performance as those tuned on 1700 labeled samples. This study suggests we need far fewer labeled examples compared to what is being used in the current literature. It was found that training a supervised parser model (Kitaev and Klein, 2018) on these labeled samples gives substantial improvements. For instance, in their analysis, ON-LSTM (Shen et al., 2019) model showed 12.5 F_1 points gain compared to using fully unsupervised criteria. Consequently, Few-shot constituency parsing, which involves training a supervised constituency parser on limited examples can be a better substitute than using these few examples for model development and tuning.

An alternate option seeking a purely unsupervised solution can be to perform hyperparameter tuning with metrics such as Perplexity, Entropy, Kullback-Leibler (KL) divergence, etc., not based on gold parses. In the case of a multilingual setting, we can fix the best-found hyperparameter values for one language and evaluate on other languages to see the generalization capabilities.

2.4 Datasets and Models

One of the major challenges involved in assessing the true performance of a parser is the lack of a fixed set of protocols needed to be followed during the experimentation phase. Firstly, a lot of previous studies (Kim et al., 2019a; Shen et al., 2018b) collapse the low frequency word tokens to a single form which may be desirable in chaining the words together in a cluster to get substantial improvements (Shi et al., 2020). Hence, it is advisable to have a fixed vocabulary size across all models to have a fair comparison. Even reporting scores on incremental subsets of vocabulary size, i.e., 10K, 20K, etc., can be a good future step. Secondly, the use of additional datasets along with or without the actual training set can lead to unfair comparisons. For instance, DIORA (Drozdov et al., 2019) and S-DIORA (Drozdov et al., 2020) are trained on SNLI (Bowman et al., 2015) and MNLI (Williams et al., 2018) datasets. URNNG (Kim et al., 2019b) is trained on a portion of one billion words (Chelba et al., 2014). Thirdly, a lot of early systems, such as the Constituent-Context Model (CCM; Klein and Manning 2002), the Dependency Model with Valence (DMV; Klein and Manning 2005), and Unsupervised Maximum Likelihood estimator for Data-Oriented Parsing (UML-DOP; Bod 2006) are trained using POS tags which the recent neural-based approaches do not seem to consider as they are instead trained using raw words. Furthermore, several early approaches learn from training sentences of length ≤ 10, in contrast to the neural-based approaches which usually do not put a length limit during training. Finally, a lot of previous works which rely on transformer architectures (Wang et al., 2019; Cao et al., 2020; Shi et al., 2021) are capable of capturing structural information about language in their intermediate layers (Goldberg, 2019; Jawahar et al., 2019) and therefore it is quite possible that the deep contextual models can encode parse trees in their word representations to some extent. For this reason, it is unfair to compare model variants especially as each one's syntactic abilities are different. A future step could be to design structural probes (Hewitt and Manning, 2019) to identify the amount of syntactical information preserved in the language model's word representation space before and after training.

2.5 Unsupervised Parsing vs. Grammar Induction

A grammar can be regarded as the specification, set or rules necessary to construct some meaning from a finite set of smaller components. Grammar Induction is the mechanism

of learning a formal grammar, i.e., determining the set of (rewrite) rules or productions and their associated probabilities from a set of sample sentences. Once learned, we can use the induced grammar to parse a set of unseen sentences and compare the predicted parse with the gold-standard parse to evaluate its capabilities on unsupervised parsing (Jin et al., 2019; Kim et al., 2019a; Zhu et al., 2020). As a result, it is worth mentioning that unsupervised parsing and grammar induction are different terms and thus cannot be used interchangeably. In reality, it is possible to design an unsupervised parser without actually learning the underlying grammar induced in a prior step (Klein and Manning, 2002; Smith and Eisner, 2005; Bod, 2006; Seginer, 2007; Parikh et al., 2014; Shen et al., 2018b, 2019; Drozdov et al., 2019, 2020; Cao et al., 2020).

In the rest of this thesis, we focus on the problem of unsupervised parsing while learning no underlying formal grammar.

Chapter 3

Neural Unsupervised Parsing Framework

In this chapter, we cover the essence of our framework: from the basic building blocks of our training algorithm to the decoding step comprising of a dynamic programming-based inference algorithm to perform a global search over the space of all valid parse trees. Before diving deep into the inner workings of our system, we give a brief introduction to several key concepts which serve as a backbone and play a crucial role in building our system (Section 3.1). Furthermore, with increasing complexity, we present three ways to bootstrap a score function that helps identify whether a node in the parse tree should dominate a given span (Section 3.2). Finally, we describe the chart parsing algorithm to extract the maximum scoring parse tree and compute its runtime complexity in Section 3.3.

3.1 Preliminaries

In the subsequent sections, we delineate a minimal set of key components and core concepts that form the basis of our parsing framework.

3.1.1 Semi-Supervised Learning

The first key idea is the use of a learning paradigm, semi-supervised learning (SSL), which lies in between supervised and unsupervised learning that attempts to use both labeled and unlabeled data to design models. Let $X = \{X_L, X_U\}$ denote the training dataset which includes a considerably small amount of labeled data set $X_L = \{(x_i, y_i)\}_{i=1}^{\mathbf{L}}$

Inputs: **L**: Labeled training set; **U**: Unlabeled training set; τ: Threshold for selection;
Algorithm:
Repeat (until no more predictions are confident):
Train a classifier m with training data **L**
Classify data in **U** with m
(for x $\in$ **U**)
Find a subset $\mathbf{U}'$ of **U** satisfying $\max m(x) > \tau$
$\mathbf{U}' = \{(x, p(x))\}$
$\mathbf{L} \leftarrow \mathbf{L} + \mathbf{U}'$
$\mathbf{U} \leftarrow \mathbf{U} - \mathbf{U}'$
Output: Generated final classifier m based on the new training set.

Figure 3.1: The self-training algorithm.

along with its corresponding labels $Y_L = (y_1, y_2, \cdots, y_L)$ and a reasonably large amount of unlabeled data set $X_U = \{(x_i)\}_{i=1}^{\mathbf{U}}$, and $\mathbf{L} \ll \mathbf{U}$. For a simple binary classification task having 2 classes, the first **L** examples within X are labeled by $\{y_i\}_{i=1}^{\mathbf{L}} \in (0,1)$. Formally, SSL aims to minimize the classification error:

$$\min_{\theta} \sum_{(x,y)\in X_L} \mathcal{L}_{\mathcal{S}}(x,y,\theta) + \alpha \sum_{(x)\in X_U} \mathcal{L}_{\mathcal{U}}(x,\theta) + \beta \sum_{(x)\in X} \mathcal{R}(x,\theta), \tag{3.1}$$

where $\mathcal{L}_{\mathcal{S}}$ denotes the supervised loss (cross-entropy loss), $\mathcal{L}_{\mathcal{U}}$ denotes the unsupervised loss, and $\mathcal{R}$ denotes the regularization loss, for each training instance. Finally, θ refers to the model parameters and α, β denote scalar hyperparameters.

3.1.1.1 Self-training

Self-training (Yarowsky, 1995; McClosky et al., 2006, 2008) (sometimes also referred to as "self-learning") is a technique that leverages the previously learned model's prediction on unlabeled data to train a model. The process repeats until no more unlabeled data remain or we observe no significant improvement in the model performance on the validation set. Given a set of labeled training set **L** and an unlabeled data set **U**, self-training works as follows: At each iteration, we train a model m using **L**, and further classify **U** with m which gives predictions $m(x)$ formulated as a probability distribution over all the classes in the dataset. We choose a subset $\mathbf{U}' \subset \mathbf{U}$ if the probability of the most likely class is greater than a predefined threshold τ. Next, we add $\mathbf{U}'$ to **L** along with $p(x) = \arg\max m(x)$ as its pseudo-label, and remove $\mathbf{U}'$ from **U**. We repeat the procedure until the algorithm converges and no more predictions on the unlabeled data

Inputs: $\mathbf{L}_1$, $\mathbf{L}_2$: Two views of labeled training set; $\mathbf{U}$: Unlabeled training set; τ: Threshold for selection;

Algorithm:

Loop for k iterations:

Train a classifier m_1 with training data $\mathbf{L}_1$

Train a classifier m_2 with training data $\mathbf{L}_2$

Classify data in $\mathbf{U}$ with m_1 and m_2

(for x $\in \mathbf{U}$)

Find a subset $\mathbf{U}'_1$ of $\mathbf{U}$ satisfying $\max m_1(x) > \tau$ and $\max m_2(x) < \tau$

$\mathbf{U}'_1 = \{(x, p_1(x))\}$

Find a subset $\mathbf{U}'_2$ of $\mathbf{U}$ satisfying $\max m_2(x) > \tau$ and $\max m_1(x) < \tau$

$\mathbf{U}'_2 = \{(x, p_2(x))\}$

$\mathbf{L}_2 \leftarrow \mathbf{L}_2 + \mathbf{U}'_1$

$\mathbf{L}_1 \leftarrow \mathbf{L}_1 + \mathbf{U}'_2$

$\mathbf{U} \leftarrow \mathbf{U} - \mathbf{U}'_1 - \mathbf{U}'_2$

Output: Generated final classifiers m_1 and m_2 based on the new training set.

Figure 3.2: The multi-view co-training algorithm.

remain confident. Figure 3.1 illustrates the general bootstrapping process involved in self-training.

In recent times, the use of self-training has gained wide attention and has been proven to substantially improve results for unsupervised parsing (Mohananey et al., 2020; Shi et al., 2020). Moreover, Steedman et al. (2003) in their experiments have shown that self-training can indeed enhance the performance of a parser by leveraging the unlabeled raw sentences. Self-training has the advantage of being used as a wrapper method to other existing classifiers. A presumably downside of this approach is that errors can reinforce over successive iterations.

3.1.1.2 Co-training

When dealing with multiple supervised classifiers, we can extend the self-training algorithm to train on multiple views, for e.g., subsets of features of a given data, through the use of the co-training (Blum and Mitchell, 1998) algorithm. It also makes the primary assumptions: (1) For any given data, each feature set should be sufficient to train a classifier and achieve a good performance, and (2) The feature set should be conditionally independent given the class. Given a set of labeled training set $\mathbf{L}$ which

can be divided into two conditionally independent feature sets $\mathbf{L}_1$, $\mathbf{L}_2$ and an unlabeled dataset $\mathbf{U}$, co-training proceeds as follows: Initially, we train m_1 and m_2 on views $\mathbf{L}_1$ and $\mathbf{L}_2$ respectively. At each iteration, the most confident predictions based on a chosen threshold τ according to precisely one of the two models is added to the set of labeled data for the other model's view. Likewise, we can repeat the same procedure for the other model too. In this manner, one model supplies the pseudo labels to the inputs on which the other model is less confident. The pseudo-code for the co-training procedure is shown in Figure 3.2.

The feature splits in co-training which mainly optimizes for sufficiency and independence conditions does lead to a good model especially if the predictions of the base learners are not too strongly correlated. However, a potential downside of co-training is the presence of a high sampling bias one can typically observe during the sample selection step when there is a strong tendency to ignore the distributional bias between the labeled and unlabeled datasets.

3.1.2 Spectral Learning

The second important component is the use of the spectral learning algorithm (Cohen et al., 2012, 2013, 2014) and the unsupervised estimation of probabilistic context-free grammars (PCFGs; Clark and Fijalkow, 2020). More specifically, the notion of *inside* and *outside* trees has a strong influence on our approach. For a given parse tree, the inside tree corresponding to a node contains the entire subtree below that node; the outside tree comprises everything in the tree except for the inside tree. Figure 3.3 illustrates the concept using an example.

The spectral learning algorithm mainly comprises two steps: In the first step, we learn a fixed dimensional representation of inside and outside trees using feature functions in conjunction with a projection defined through singular value decomposition (SVD). In the second step, we perform the parameter estimation of the model. Our learning algorithm identifies the presence or absence of a node dominating a substring in the parse tree as a latent variable. The patterns of co-occurrence of the string that the node dominates (which is represented by the inside string) and the rest of the sentence (which is represented by the outside string) help to determine if a node exists or not. The concept of inside *trees* versus outside *trees* is important in the case of spectral learning for latent-variable PCFGs (L-PCFGs; Cohen et al., 2012). However, for our use case, since no trees are present during learning, there is a strong desire to enhance it

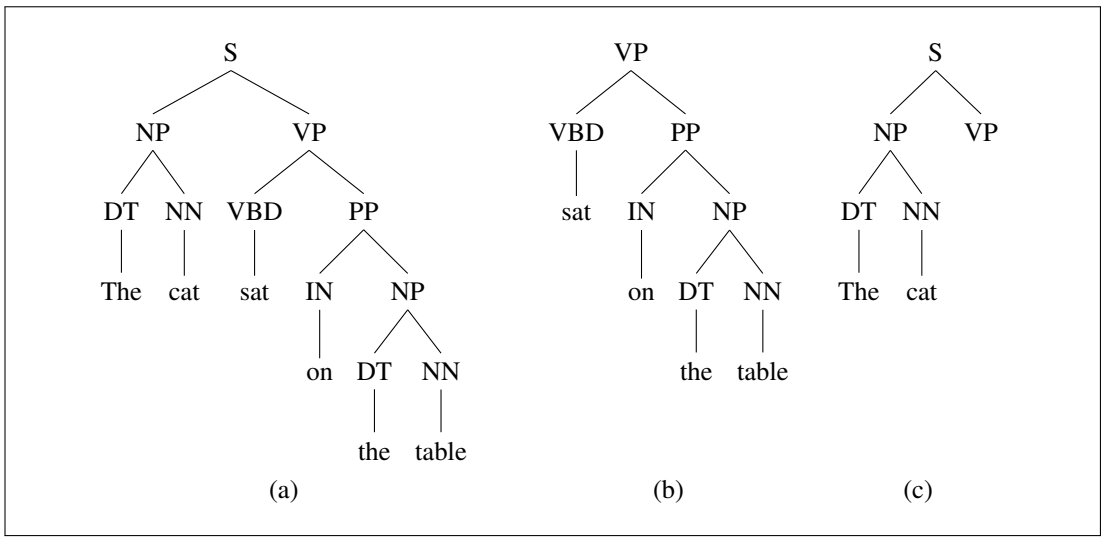

Figure 3.3: The extraction of inside and outside trees at a given node for a sample sentence, *"the cat sat on the table"*. (a) shows the obtained parse tree, (b) inside tree corresponding to the nonterminal VP, and (c) outside tree corresponding to the nonterminal VP.

further to derive information only from the raw strings.

We note that our learning approach is related to a previous work on incorporating spectral learning for unsupervised parsing by Parikh et al. (2014) to a certain degree. For instance, both the methods rely on two central notions: the usage of word embeddings to create a vector representation for encoding each word in the sentence, and heuristics based on punctuation marks as a useful signal to extract the fencepost positions of constituent spans. Furthermore, we base our approach on the idea of correlation driving the identity of a latent state, for e.g., our objective is to minimize the inter-correlation between inside and outside vectors. In contrast, their approach uses the additive tree metric property to backward compute the distance among latent variables parameterized by the distances among the observed variables.

3.2 Training Algorithm

We will use two salient notions inspired by spectral learning throughout the thesis: *inside* and *outside* strings. For a given input sentence, $x = x_1 \cdots x_n$ and a span (i, j), the sequence $x_i \cdots x_j$ denote the inside string corresponding to the span (i, j), while the pair of sequences $(x_1 \cdots x_{i-1}, x_{j+1} \cdots x_n)$ denote the outside string corresponding to the span

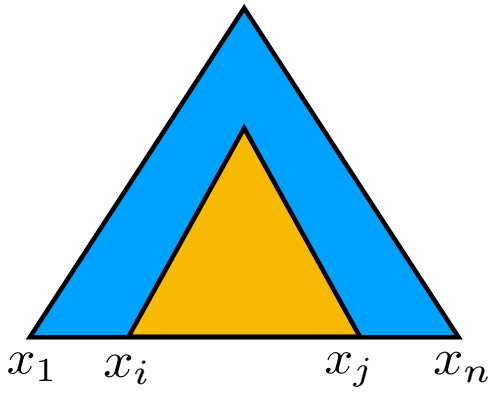

Figure 3.4: A depiction of a syntax tree, with the inside string as depicted by the sequence $x_i \cdots x_j$ and the outside string as depicted by the sequence $(x_1 \cdots x_{i-1}, x_{j+1} \cdots x_n)$ that provides external context for the inside representations.

(i, j). Let us consider the diagram of a syntax tree in Figure 3.4 which is decomposed into two components. Each of these components denoted by orange and blue colours is a "view" of the whole parse tree that provides information on the identity of the node that spans the words $x_i \cdots x_j$. In the scenario of unsupervised parsing which expects the tree being unobserved during training, we have to rely on the substrings spanning either the blue or the orange part to hypothesize if a node is present at that point or not. We denote by $\mathbf{h}_{\text{in}}(i, j)$ representations for inside strings and $\mathbf{h}_{\text{out}}(i, j)$ representations for outside strings which are both vectors. We associate each token constituting the inside and outside strings with a vector after extracting the contextualized word representations from a PLM, RoBERTa$_{\text{BASE}}$ (Liu et al., 2019). Henceforth, we will call $\mathbf{h}_{\text{in}}$ the *inside representation*, and $\mathbf{h}_{\text{out}}$ the *outside representation*. In our framework, the inside and outside representations are strictly complementary: which means that the outside representation for a given node x can be calculated without knowing beforehand the elements under that node.

The core essence behind using bootstrapping is to begin with a small seed set of training examples (x, i, j, b) where (i, j) denotes a span in a sentence x, and $b \in \{0, 1\}$, depending on whether the span (i, j) is dominated by a node in the syntax tree or not. We note that the availability of either the inside or outside strings is sufficient to bootstrap the seed set and the corresponding classifier trained on this seed set typically return a probability $p(b \mid x, i, j)$. The classifier after learning using the bootstrapping seed set can be then applied on the remaining training set. Furthermore, we only add the training examples to the existing seed set if it satisfies the condition of the classifier, which is to be able to predict with a certain threshold of confidence pertaining to the

Inputs: I represents the labeled inside set; **U** is a set of Unlabeled training sentences;

Algorithm:

- Loop for K iterations:
  1. Learn the inside classifier m_{in} based on $\mathbf{h}_{\text{in}}(i,j)$ derived from **I**
  2. Use m_{in} to label **U** to get the predicted inside strings $\hat{y}_{\text{in}}$
  3. If $\hat{y}_{\text{in}} > \tau_{\text{max}}$, extract c constituents randomly and add it to the set of pseudo-constituents $\mathcal{X}_c$
  4. If $\hat{y}_{\text{in}} < \tau_{\text{min}}$, extract d distituents randomly and add it to the set of pseudo-distituents $\mathcal{X}_d$
  5. $\mathbf{I} = \mathcal{X}_c \cup \mathcal{X}_d$
- Get outside strings for each **I**; Assign to the set of labeled output sentences **O**
- Learn outside model m_{out} based on $\mathbf{h}_{\text{out}}(i,j)$ derived from **O**

Output: inside model m_{in}, outside model m_{out}

Figure 3.5: Our self-training algorithm.

label b. We carry out this process in an iterative fashion.

In the following three sections, we present three learning algorithms of increasing complexity in their use of inside and outside strings.

3.2.1 Modeling Using Inside Strings

In order to compute the inside representation $\mathbf{h}_{\text{in}}$, we fine-tune a sequence classification model, that encodes a fixed-vector representation for each token in the inside string present in the training sentences. We compute the inside representations in a bottom-up manner from a node's children that encodes the phrase information of the inner content in the span. Once we build the seed bootstrapping set, the inside model m_{in} which is modeled at a sentence level computes an inside score, $score_{\text{in}}(i,j)$, from the inside representation $\mathbf{h}_{\text{in}}(i,j)$ corresponding to the span (i,j). In the same manner, we compute the inside score for all spans in the unlabeled input training sentence **U**. Subsequently, we take the most confident inside strings from **U** based on a predefined threshold τ. More specifically, the sentences whose confidence bounds is higher than τ_{max} are treated as pseudo-constituents while the sentences whose confidence bounds is

lower than τ_{min} are treated as pseudo-distituents. To reduce the training time, we choose a random sample of c pseudo-constituents and d pseudo-distituents which constitute the labeled inside set $\mathbf{I}$. We further subject the inside model to an iterative self-training procedure whose steps are detailed as shown in Figure 3.5. An alternative option for self-training using probability distribution can be thought of along the lines of minimizing the KL divergence (Kullback and Leibler, 1951) between teacher and student outputs. We suspect it would not yield similar performance in comparison to employing hard judgments using class labels.

3.2.2 Modeling Using Inside and Outside Strings

Once we build the inside model with self-training, we can then look to incorporate the outside string. For a given inside string, we can further divide the outside string into two parts: a left-outside to capture the first token of the outside string towards its immediate left side, and a right-outside to capture the first token of the outside string towards its immediate right side. We denote the sentence boundary and the given inside string under consideration by a `[MASK]` placeholder token. Finally, we can reduce the outside string to the form consisting of the triple (x_{i-1}, `[MASK]`, x_{j+1}) pertaining to a given inside string $x_i \cdots x_j$. In order to compute the outside representation $\mathbf{h}_{\text{out}}$, we extract the triple for every span in the training sentences and fine-tune another sequence classification model which encodes a fixed-vector representation for each triple. The outside model m_{out} computes an outside score, $score_{\text{out}}(i, j)$, from the outside representation $\mathbf{h}_{\text{out}}(i, j)$ corresponding to the span (i, j). We compute the outside representations in top-down manner from a node's parent and siblings that encodes the contextual information of the span.

3.2.3 An Iterative Co-training Algorithm

The main inspiration behind our final algorithm is the standard co-training algorithm (refer Section 3.1.1.2). To cover the global information, we consider the inside and outside strings to be the two views while performing the co-training procedure. We note that the co-training procedure requires the two views to be independent of each other conditioned on the label of the training instance. This is akin to the type of assumption that, for instance, PCFGs satisfy, when breaking a tree into an inside and outside tree: the two trees are conditionally independent given the nonterminal that connects them. In our case, we satisfy this assumption by creating inside and outside string representations

Inputs: $\mathbf{I}$ is the set of labeled inside sentences; $\mathbf{O}$ is the set of labeled outside sentences; $\mathbf{U}$ is a set of unlabeled sentences.

Algorithm: Loop for K iterations:

- Choose c pseudo-constituents and d pseudo-distituents from the most confidently predicted outside strings $\hat{y}_{\text{out}}$ from $\mathbf{U}$ based on τ
- Extract the inside strings $\hat{\mathbf{I}}$ corresponding to the c pseudo-constituents and d pseudo-distituents of outside
- $\mathbf{I} = \mathbf{I} \cup \hat{\mathbf{I}}$
- Train the inside model m_{in} based on $\mathbf{h}_{\text{in}}(i, j)$ derived from $\mathbf{I}$
- Choose c pseudo-constituents and d pseudo-distituents from the most confidently predicted inside strings $\hat{y}_{\text{in}}$ from $\mathbf{U}$ based on τ
- Extract the outside strings $\hat{\mathbf{O}}$ corresponding to the c pseudo-constituents and d pseudo-distituents of inside
- $\mathbf{O} = \mathbf{O} \cup \hat{\mathbf{O}}$
- Train the outside model m_{out} based on $\mathbf{h}_{\text{out}}(i, j)$ derived from $\mathbf{O}$

Output: Two models m_{in}, m_{out}, that predict the inside and outside scores for unlabeled sentences. We combine these predictions by multiplying together and optionally re-normalizing their class probability scores.

Figure 3.6: Our co-training algorithm.

separately. As a result, the sufficiency and independence of each view is guaranteed. Once we have the inside and outside models trained on their conditionally independent inside and outside feature sets, we can employ an iterative approach. At each iteration, we move only the inside strings $\hat{\mathbf{I}}$ that are confident to be likely the inside strings of pseudo-constituents and pseudo-distituents to the labeled training set of the inside model $\mathbf{I}$ whose decisions are guided by the outside model. Therefore, the outside model (which acts as a teacher agent) provides the labels to the inside strings on which the inside model (which acts as a student agent) is uncertain. Likewise, we move only the outside strings $\hat{\mathbf{O}}$ that are confident to be likely the outside strings of pseudo-constituents and pseudo-distituents to the labeled training set of the outside model $\mathbf{O}$ whose decisions are guided by the inside model. Therefore, the inside model (which acts as a teacher agent) provides the labels to the outside strings on which the outside model (which acts as a student agent) is uncertain. The steps to perform the overall iterative co-training procedure is outlined in Figure 3.6.

Eventually, we can combine the scores obtained by the inside and the outside models to get the score, $score(i, j)$, for each span:

$$score(i, j) = score_{\text{in}}(i, j) \cdot score_{\text{out}}(i, j). \tag{3.2}$$

3.3 Inference

Our decoding follows the chart-based parser approach, which is a Viterbi version of the CKY algorithm (Cocke, 1969; Kasami, 1966; Younger, 1967), to produce a globally optimized parse tree for each sentence. Let us consider a span for a sentence of length n denoted by the pair (i, j) with i and j referring to the beginning and ending positions of a span and k referring to the partition between i and j. Then, the pseudocode considering only the nested loop for the CKY algorithm can be written as:

```
for i in range(2, n):              # Length of span
    for j in range(1, n-i+1):      # Start of span
        for k in range(1, i-1):    # Partition of span
```

Mathematically, the above code block can be rewritten as:

$$t = \sum_{i=2}^{n} \sum_{j=1}^{n-i+1} \sum_{k=1}^{i-1} 1 \cdot \tag{3.3}$$

Upon simplifying, we get:

$$t = \frac{(n^3 - n)}{6} \cdot \tag{3.4}$$

Therefore, for a sentence of length n, the time complexity for the CKY inference is $O(n^3)$ and the space complexity is $O(n^2)$ (one cell for each substring).

A score function, $score(i, j)$, learned from our algorithm assigns a score to every span in the sentence. The score of a candidate tree is calculated by summing up the scores of all its spans. Finally, the optimal parse tree t^* having the highest score is chosen by:

$$t^* = \arg\max_{t \in \mathcal{T}} \sum_{(i,j) \in t} score(i, j), \tag{3.5}$$

where $\mathcal{T}$ is the set of possible binary trees over the sentence and $(i, j) \in t$ denotes that the span (i, j) appears in t. When $score(i, j)$ is the probability of a span (i, j) being in the correct tree, this formulation gives the tree with the highest expected number

of correct constituents (Goodman, 1996). This formulation has been used recently by several unsupervised constituency parsing algorithms (Kim et al., 2019b,a; Cao et al., 2020; Li et al., 2020a; Drozdov et al., 2019, 2020).

Chapter 4

Multi-View Learning

In this chapter, we present details about our implementation to perform unsupervised parsing. Our approach relies on optimizing two models — inside and outside models — with different views of the original input sentence and the resulting interactions help each model to contribute to the improvement in performance of the other in an iterative fashion. This chapter is organized as follows. We first give a high-level overview of transformer architecture-based PLMs and their syntactic abilities (Section 4.1). Next, we elucidate the technicalities involved in the seed bootstrapping procedure both for right-branching and left-branching languages (Section 4.2). Finally, we discuss the training aspects relating to the inside, outside, and joint model and the additional heuristics-based span refinement strategies acting as a weak form of supervision signal in Section 4.3.

4.1 Preliminaries

In this section, we provide a thorough background about PLMs, more specifically, the ones based on transformer architectures, which serve as the backbone model in our overall approach, and highlight recent studies that analyze the nature of syntactic information these PLMs possess.

4.1.1 Transformers

Although modeling a wide variety of NLP tasks is possible using the high versatility of recurrent neural networks, there are clear disadvantages: the inherent recurrent structures make them challenging to perform parallelization, and the handling of long

clauses causes an impediment to learning due to the vanishing gradient problem.

To counter these two limiting constraints, Vaswani et al. (2017) propose a new model architecture: the Transformer. It consists of a multi-layer, non-recurrent neural sequence model, encompassing a self-attention component to prepare contextual input representations. The use of self-attention allows parallel construction of the step representations of the input since it re-computes a state representation at every step from the overall input. These representations do not depend on a time-dependent state that is conventional in RNNs. As a result, transformers facilitate and accelerate the training of larger networks. The architecture of the vanilla transformer is shown in Figure 4.1. We investigate the various components in greater detail in the following sections.

4.1.1.1 Transformer Block

A Transformer consists of an encoder and a decoder, each of which is a stack of ℓ identical blocks.[1] Each transformer block applies the following transformations to the input of the block $\mathbf{h}^{\ell-1}$:

$$\tilde{\mathbf{g}}^{\ell} = \textsc{MultiAttn}\left(\mathbf{h}^{\ell-1}\right) \tag{4.1}$$

$$\mathbf{g}^{\ell} = \textsc{LayerNorm}\left(\tilde{\mathbf{g}}^{\ell} + \mathbf{h}^{\ell-1}\right) \tag{4.2}$$

$$\tilde{\mathbf{h}}^{\ell} = \text{FFN}\left(\mathbf{g}^{\ell}\right) \tag{4.3}$$

$$\mathbf{h}^{\ell} = \textsc{LayerNorm}\left(\tilde{\mathbf{h}}^{\ell} + \mathbf{g}^{\ell}\right), \tag{4.4}$$

where $\textsc{MultiAttn}(\cdot)$ denotes a multi-headed self-attention mechanism, $\text{FFN}(\cdot)$ denotes a two-layer position-wise feed-forward network, along with residual connections (He et al., 2016), and $\textsc{LayerNorm}(\cdot)$ denotes the layer normalization operation (Ba et al., 2016) typically one applies to the resulting output of self-attention and feedforward network. We note that some transformer implementations may not conform to the exact same equations (Yang et al., 2019), however, commonly all include a multi-head attention component and a feed-forward network.

4.1.1.2 Multi-head Attention

The multi-head attention is an extension of the standard attention mechanism which computes attention with $\mathbf{H}$ independent attention heads. As a result, information from different representation subspaces at various positions can be jointly attended to by

[1] Equations in Section 4.1.1 adapted from Bosselut (2020, Section 2.4).

the model. Consider a sequence of vectors $\mathbf{H} \in \mathbb{R}^{l \times d}$, where l and d denote the length and dimension of the input sequence, the self-attention projects $\mathbf{H}$ into three different matrices: the query matrix $\mathbf{Q}$, the key matrix $\mathbf{K}$, and the value matrix vector $\mathbf{V}$. The attention consists of multiple heads where each head computes a unique scaled dot product attention distribution over $\mathbf{V}$ using $\mathbf{Q}$ and $\mathbf{K}$ to get the output representation:

$$\mathbf{Q}, \mathbf{K}, \mathbf{V} = \mathbf{H}\mathbf{W}^{\mathbf{Q}}, \mathbf{H}\mathbf{W}^{\mathbf{K}}, \mathbf{H}\mathbf{W}^{\mathbf{V}} \tag{4.5}$$

$$\text{ATTENTION}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V}, \tag{4.6}$$

where $\mathbf{W}^{\mathbf{Q}}, \mathbf{W}^{\mathbf{K}}, \mathbf{W}^{\mathbf{V}}$ are learnable parameters and softmax() is performed row-wise. The multi-headed attention module adds an extension to the single head self-attention by enhancing the self-attention abilities through jointly modeling interactions from diverse representation spaces:

$$\text{head}_i = \text{ATTENTION}\left(\mathbf{H}\mathbf{W}_i^Q, \mathbf{H}\mathbf{W}_i^K, \mathbf{H}\mathbf{W}_i^V\right) \tag{4.7}$$

The outputs of the attention heads head_i can be later concatenated:

$$\text{MULTIH}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\text{head}_1; \ldots; \text{head}_H]\,\mathbf{W}^o, \tag{4.8}$$

where $\mathbf{W}^o$ denotes the output projection of the concatenated outputs of the attention heads, and $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, $\mathbf{W}_i^V$ represent the head-specific projections for $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ respectively.

4.1.1.3 Positional Embeddings

Since the transformer does not rely on recurrence or convolution and has no formal notion of ordering of tokens, there has to be a mechanism to inject information about the position of tokens. The original self-attention adds the absolute position embedding $\mathbf{p}_t$ for each absolute position in the sequence to the input token embedding for any input word $\mathbf{x}_t$:

$$\mathbf{x}_t = \mathbf{x}_t + \mathbf{p}_t, \tag{4.9}$$

where $\mathbf{p}_t, \mathbf{x}_t \in \mathcal{R}_x^d$. The absolute positional encodings can be fixed with the help of pre-defined sinusoidal and cosine functions or can be learned through training parameters.

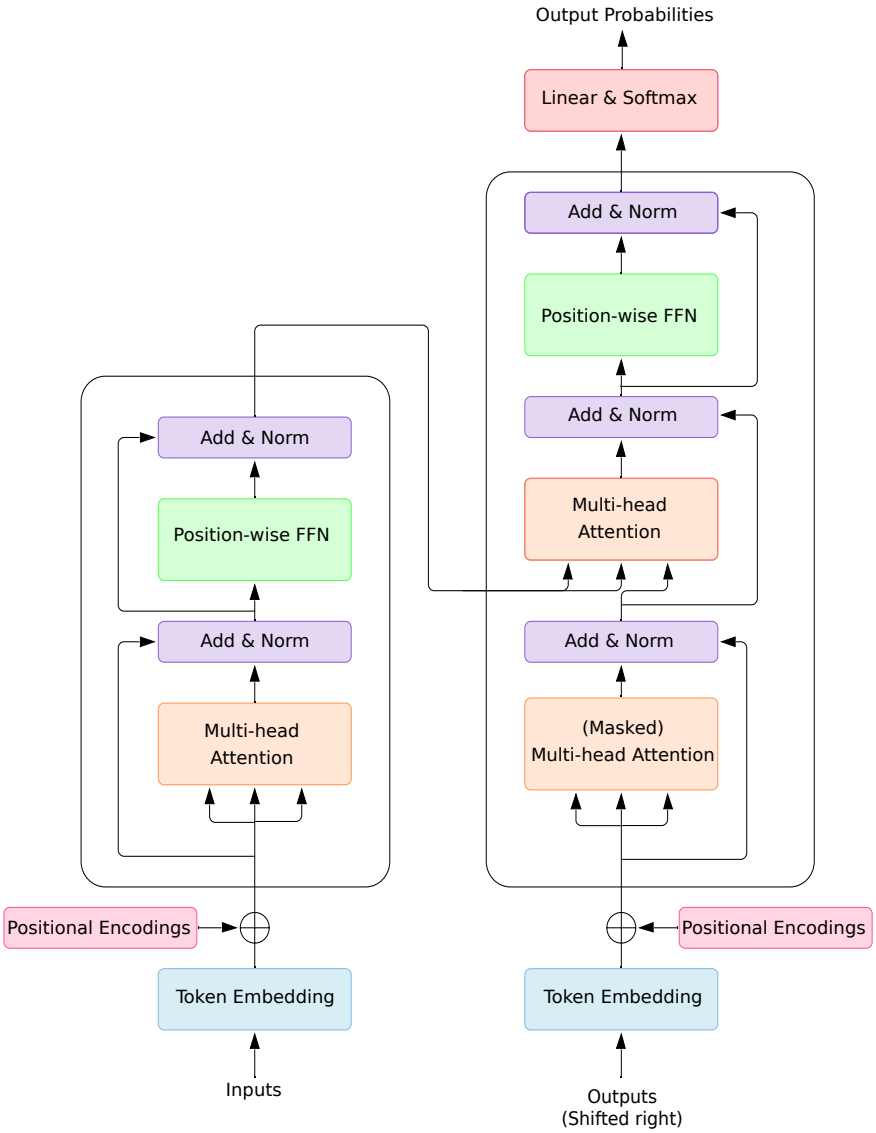

Figure 4.1: Block diagram of the Transformer architecture.

4.1.2 Bidirectional Transformer Language Models

Bidirectional transformers, as opposed to left-to-right language models (Radford et al., 2018, 2019; Brown et al., 2020), are capable of conditioning on both future as well as preceding tokens (corresponding to left and right context in all layers) when learning to predict a token at a specific position in a sequence. In the following sections, we succinctly describe the most widely adopted pre-training method in the NLP community for model initialization, BERT (Devlin et al., 2019), and its enhanced variant, RoBERTa (Liu et al., 2019).

4.1.2.1 BERT

BERT is a transformer-based PLM which is pre-trained on 3.3 billion tokens of English text. The BERT's training consists of two kinds of objective functions: (1) *Masked language modeling* (MLM), where we replace 15% of the input tokens in a sentence with a special token $[MASK]$ and train BERT to predict them with their surrounding words, and (2) *Next sentence prediction* (NSP), which is a binary classification loss to predict whether two segments are adjacent in the original document or is taken from a different document. For this purpose, the training corpus consists of tuples $([CLS], x_1, \ldots, x_N, [SEP], y_1, \ldots, y_M, [EOS])$, with learnable special tokens $[CLS]$ to classify whether $x_1, \ldots, x_N$ and $y_1, \ldots, y_M$ follow each other and $[SEP]$ to segment two sentences. With a probability of 50%, we replace the second input with a random input.

4.1.2.2 RoBERTa

An optimized variant of BERT pre-training, RoBERTa model, proposes to match or exceed BERT's performance by incorporating the following four modifications: (1) training the model longer with bigger batches and over more unlabeled data, (2) discarding the NSP objective, (3) training on sequences of large sequence length, and (4) dynamically changing the masking pattern applied to the training data instead of a single static mask. We use RoBERTa as the base model for all our experiments.

4.1.3 Syntactic knowledge in PLMs

Lin et al. (2019) show that BERT's representations encode hierarchical information rather than linear. In terms of how syntax is represented, it appears that self-attention weights do not directly encode syntactic structure. Htut et al. (2019) find that the attention weights present in the different layers/heads of the BERT models are not representative of syntactically meaningful parse trees. Moreover, they find that fine-tuning does not have much impact on the syntax inducing ability of the attention heads. Instead, BERT token representations can be an alternative option to recover the syntactic information.

Furthermore, there have been many studies dealing with designing syntactic probing experiments to look for evidence pertaining to the knowledge encoded in BERT weights. Hewitt and Manning (2019) design a structural probe using BERT's token embeddings for finding syntax and recover parse trees on the English PTB with the help of low-rank transformation matrices. Wu et al. (2020) introduce a parameter-free

probing approach and their results show that the syntax which BERT learns and the linguistic annotations have substantial disagreements. In addition, the fill-in-the-gap probes of MLM reveal that BERT takes into consideration subject-verb agreement when carrying out the cloze task (Goldberg, 2019; van Schijndel et al., 2019). Even though BERT exhibits strong ability to attach subject nouns with their hypernyms, it fails miserably for cases when the presence of negation reverses the truth of those hypernyms (Ettinger, 2020).

4.2 Seed Bootstrapping

In this section, we pose the task of identifying the plausible constituents and distituents from unlabeled sentences as a sequence classification task. While we do use a fixed template to perform the seed bootstrapping process, we argue that the method is still "unsupervised" at its core, as we are not using the *annotated labels* or *POS tags* for parsing purposes, and the amount of engineered rules are minimal, inducing more the inductive bias of the algorithm (inductive bias is necessary for any learning algorithm) than the actual algorithm. In our analysis, we assume the language is already known before and therefore its structure (left/right-branching), a form of our weak supervision.

4.2.1 Formulation for Right-branching Languages

We know from the linguistic theory that single words in a sentence (with an exception of certain contractions and possessives) are constituents. In addition, complete sentences are constituents and also the largest among all of its other constituents. We exploit this idea to devise a plan for the preparation of the *constituent* and *distituent* classes. To generate the constituent class, we take the complete sentence `(start:end)`. To generate the distituent class, we take the `(start:end-1)`, $\cdots$, `(start:end-6)` slices, where `start` and `end` denote the 0^{th} and N^{th} position (sentence length) respectively. The intuition behind choosing the slicing range for distituents is governed by the fact that the longer the sentence, there would be an unlikely chance that the constituent spans extend right to the very end of the sentence. After creating the dataset constituting the synthetically obtained constituent and distituent classes from the seed bootstrapping technique, we divide it into train/validation splits. All the string slicing decisions are made based on the feedback from this validation split iteratively until we see the degradation in performance (measured using F_1 score) on the synthetic set of trees.

In our experiments, both PTB and CTB follow the aforementioned procedure. In the case of PTB, we make use of additional signals such as casing-specific information by adding contiguous title-case words while allowing only the apostrophe mark, since apostrophes separate different tokens (for e.g., *Britain's* becomes *Britain 's*). We simply follow this tokenization. To account for the bias due to the casing of spans, we identify the most common first word and produce lowercase equivalents of contiguous title-case words which start with that word. Moreover, since all of the sentences in the PTB training set for the constituent class start with capital letters, it acts as a mitigation step by calibrating the model's aggressiveness towards the casing of the starting letter.

4.2.2 Formulation for Left-branching Languages

We design our parsing scheme slightly differently compared to the one mentioned in Section 4.2.1, although along the same style, to accommodate the change in the position of heads. We choose the slice `(start:end)` in the sentence to label the constituent class and `(start+1:end)`, $\cdots$, `(start+4:end)` slices are chosen to label the distituent class. In our experiments, KTB follows the aforementioned procedure. Furthermore, we break the sentences on "*" mark and consider the resulting fragmented parts also as constituents. Additionally, we reiterate that we do not rely on the development set with the gold-standard trees whatsoever.

Figure 4.2 illustrates the underlying working flow of our algorithm in an end-to-end fashion.

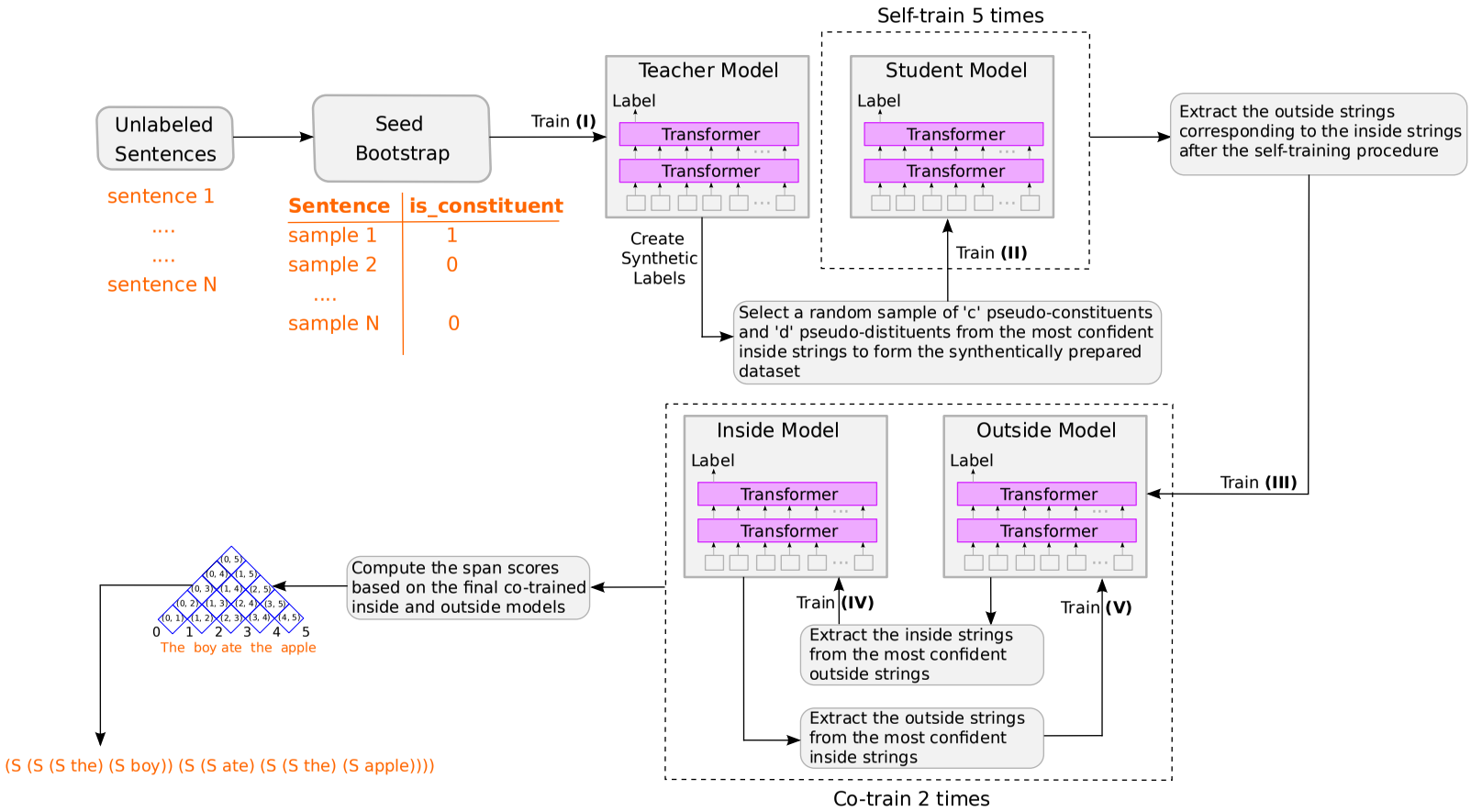

Figure 4.2: Block diagram detailing our approach. We perform the self-training procedure for five iterations which follow multiple steps; **(I)**: Fine-tune a RoBERTa$_{\text{BASE}}$ model (teacher) on a downstream task using a cross-entropy loss after seed bootstrapping; **(II)**: Synthetically annotate this data using the teacher model and select top K samples corresponding to each class to form the final synthetic dataset; We fine-tune a RoBERTa$_{\text{BASE}}$ model (student) on this dataset using hard labels and retrieve the outside strings from the most confident insides; **(III)**: Train the outside classifier on these outside strings; We perform the co-training procedure for two iterations which follow a two-fold optimizing step; **(IV)**: Retrieve the inside strings from the most confident outsides and train the inside classifier; **(V)**: Retrieve the outside strings from the most confident insides and train the outside classifier.

4.3 Model Training

In this section, we lay out details concerning the practical implementation necessary for training using the inside, outside, and the joint models.

4.3.1 Inside Model

Our inside model consists of a RoBERTa$_{\text{BASE}}$ transformer variant fine-tuned on the bootstrapped dataset with a sequence classification head on top using a standard cross-entropy loss. We do not explicitly perform hyperparameter search and hence keep our optimization minimal. Instead, we build on the default values for all the hyperparameters as mentioned in HuggingFace Transformers[2] (Wolf et al., 2020) due to its previous superior performance in sequence classification tasks on the GLUE benchmark.[3] Accordingly, we choose the Adam optimizer with a learning rate of $3e-5$, batch size of 32, epochs of 3, and maximum sequence length of 128, for all our models. Additionally, to monitor the validation loss and prevent overfitting, we apply the early stopping regularization technique on the train/validation random split of 80/20 after the seed bootstrapping procedure. We set the patience value at 3. We do note that, however, we do not touch the development set of PTB. Furthermore, we carry out model checkpointing, as well as logging, after every 100 steps. We evaluate the inside model on MCC (Matthews Correlation Coefficient) as well as F_1 since the classes are imbalanced. After fine-tuning using the pre-trained RoBERTa$_{\text{BASE}}$ model, we find that our best inside model achieves 0.28 MCC and 0.42 F_1 on the internal validation set. Lastly, we fine-tune the inside model on the unlabeled training sentences that generates an inside score $score_{\text{in}}(i, j)$ for every span.

For the Chinese monolingual experiment, we employ `bert-base-chinese` which is trained on cased Chinese simplified and traditional text, and for Japanese monolingual experiment, we employ `cl-tohoku/bert-base-japanese` which is trained on Japanese Wikipedia available at https://huggingface.co/models.

In order to augment the distituent class to create more training instances, we experiment with several strategies, however, nothing gave us concrete benefits. Some of those include — word deletion (select random tokens in the sentence and replace them by a special token), span deletion (same as word deletion, although puts more focus on deleting spans), reordering (sample randomly several pairs of span and swap them

[2] https://huggingface.co/transformers/v2.3.0/examples.html#glue
[3] https://gluebenchmark.com/

pairwise), and substitution (sample few words and replace them with their synonyms).

Eventually, we subject the inside model to the implementation details as mentioned in Section 3.2.1.

4.3.1.1 Role of Heuristics

To further improve the parser's performance, we inject additional bias into our algorithm through the use of certain heuristics for PTB. Moreover, for CTB and KTB, it is not necessary to incorporate such rules as our models show superior performance without them. Once we compute the inside score, $score_{\text{in}}(i, j)$, we make use of the following post-hoc span refinement strategies to prune out false constituents, another form of our weak supervision:

- We treat punctuation characters to mark the boundaries of a span and penalize any span that crosses its demarcated punctuation region (indicated by a span) by assigning a negative penalty of 0.25. Furthermore, the tokenization scheme of PTB considers punctuation as a separate token, and thus if a token consists only of a punctuation mark, then we consider it a constituent boundary.

- We delete any constituent if it starts or ends with the most common word succeeding the comma punctuation.

- We extract the most common starting word and check if its accompanying word does not belong to either the stop word or is present in the top 20 most frequent tokens of the PTB training set. Finally, we assign the scores of these corresponding spans in the CYK chart cell to the maximum value.

- Intuitively, from the linguistic definition of constituents, we refrain from bracketing if we identify a contiguous group of rare words (tokens not in the top 1000 most frequent list).

We argue that these heuristics are not language-specific biases, that contribute only to a certain extent in making the parser robust and should be considered a standard post-processing step. Overall, we observe about 3.8 F_1 improvements contingent on the inside model. Based on our experiments, the role of heuristics is smaller in comparison to the combined self-training and co-training gains. Their effect becomes negligible after successive iterations of the self-training process due to the predictions roughly

obeying the templated rules. As outlined in Figure 3.1, we conduct self-training on the inside model for three iterations.[4]

4.3.2 Outside Model

Once we calculate the inside scores, we extract the outside strings of spans having the inside scores satisfying a predetermined cutoff value and choose the threshold values corresponding to the lower and upper bounds to ensure the distribution of class labels is about 1:10 (with the distituent class being the majority). As a result, the model predictions do not lean aggressively towards constituents in the case of an equal split. Moreover, from a linguistic standpoint, one can be certain that the count of distituents in a sentence must necessarily exceed its constituents. We assume the outside strings satisfying the upper and lower bounds of the threshold as *gold-standard outside* of constituents and distituents, respectively. As done previously, we fine-tune the outside model (similar in architecture to the inside model) on the unlabeled training sentences that generates an outside score $score_{\text{out}}(i, j)$ for every span. Lastly, we subject the outside model to the implementation details as mentioned in Section 3.2.2.

4.3.3 Jointly Learning with Inside and Outside Models

Up until this stage, we have two models that operate separately on the inside and outside strings. Once we compute the outside scores using the outside model, we run it on the unlabeled training sentences and choose the outside strings obeying the threshold bounds as determined previously (see Section 4.3.2). Next, we extract the corresponding inside strings and re-train the inside model on the *old highly confident inside strings* (from Section 4.3.1) along with the *new inside strings obtained from the highly confident outside strings*. Similarly, we apply the same procedure on the outside model to augment its input data. As mentioned in Figure 3.2, we repeat the co-training process twice.

Subsequently, we subject the inside and outside models to the implementation details as mentioned in Section 3.2.3. Finally, we calculate the score corresponding to a given span, $score(i, j)$, as shown in Equation 3.2 and select the best parse tree using Equation 3.5.

[4] We only use the top 5K inside strings for self-training to cover maximum possible iterations as it is representative of the whole training set in terms of the average sentence length and punctuation marks. Accordingly, we set τ_{min} as 0.0005 and τ_{max} as 0.995.

Chapter 5

Empirical Study

In this chapter, we aim to provide readers with an empirical comparison of different unsupervised parsing systems. It is important to set up a unanimously agreed-upon pipeline when designing unsupervised parsers to draw meaningful comparisons. For instance, several studies used part-of-speech tags as an additional signal or did not specifically create splits for training and testing purposes. In Section 5.1, we provide details concerning the experimental settings mainly comprising datasets and baselines we use to draw comparisons. Further, we evaluate our approach against previous systems for both right-branching as well as left-branching languages (Section 5.2). Additionally, we conduct a thorough analysis to study the effect of distituent selection on the amount of bias it injects during the seed bootstrapping procedure. Moreover, we analyze the impact of self-training and co-training on the overall performance broken down at every stage of the training as well as perform error analysis to inspect the parser's misclassifications. We also briefly touch upon the generation of phrase labels (Section 5.3). Finally, we enumerate the noticeable failings of our approach in Section 5.4.

5.1 Experimental Setup

In this section, we outline the datasets we use to test our methodology on. Our evaluation strategy closely follows as in recent works (Kim et al., 2019a; Cao et al., 2020; Li et al., 2020b). To summarize, we ignore punctuation during evaluation while we do keep them during training. Finally, we compute the sentence-level F_1 for right-branching languages and Evalb F_1 for left-branching languages. Refer Section 2.2.2 for more comprehensive details about the various evaluation methods.

5.1.1 Input Corpora

In grammar induction, treebanks supply the input data, which contain both sentences and their annotated POS tags. Early approaches typically considered both these sources of information, whereas, recent approaches used only raw text to obtain an improved performance. The most frequently used corpora for English language experiments is the Wall Street Journal (WSJ) section of the Penn Treebank (PTB; Marcus et al. 1993). with standard splits of section 02-21 for training, 22 for validation and 23 for testing purposes. For our Chinese experiments, the Chinese Treebank (CTB; Xue et al. 2005) corresponding to version 5.1 with the same splits as in Chen and Manning (2014) was used. For our Japanese experiments, we used the Japanese Keyaki Treebank (KTB; Butler et al. 2012) with 80% of the sentences for training, 10% for validation, and 10% for testing after shuffling the corpus. In addition, for multilingual experiments, Statistical Parsing of Morphologically Rich Languages (SPMRL; Seddah et al., 2013) dataset can be used to evaluate grammar induction capabilities for nine languages. A significant drawback of relying on treebanks is that it is relatively disparate in nature compared to how humans converse in their day-to-day lives but rather typical of the structure of language commonly parsed by supervised parsing systems.

5.1.2 Trivial Baselines

We consider three naïve baselines consisting of right-branching, left-branching, and balanced binary trees. A balanced tree is one in which all subtrees' right and left branches contain the same number of elements. We construct the balanced tree baseline by recursively dividing a group of n leafs into two groups each of size $\lceil \frac{n}{2} \rceil$ adjoining one another until there exists one leaf node per group. To construct the right-branching baseline, we combine two nodes from right to left, whereas in the left-branching baseline, we combine them from left to right. In addition, we also specify the oracle baseline, which is the highest possible score with binarized trees because we compare them to non-binarized gold trees according to the convention, as most unsupervised parsing methods output fully binary trees.

5.2 Results

In this section, we list the experimental results and report the mean/max scores resulting from 4 runs of the chosen model with different random seeds for both right-branching

and left-branching languages.

5.2.1 Right-branching Languages

For PTB analysis, we choose to evaluate a total of nine models after deciding to group strong baselines containing open-source implementation broadly under four categories: (1) relying on the concept of syntactic depth or making no independence assumptions (PRPN; Shen et al., 2018b)[1], (ON; Shen et al., 2019)[2], and (URNNG; Kim et al., 2019b)[3], (2) autoencoder-like training objective (DIORA; Drozdov et al., 2019)[4] and (S-DIORA; Drozdov et al., 2020)[5], (3) PCFGs with neural parameterization (Neural PCFG and Compound PCFG; Kim et al., 2019a)[6], (4) based on transformer models (Tree Transformer; Wang et al., 2019)[7] and (Constituency Test; Cao et al., 2020)[8].

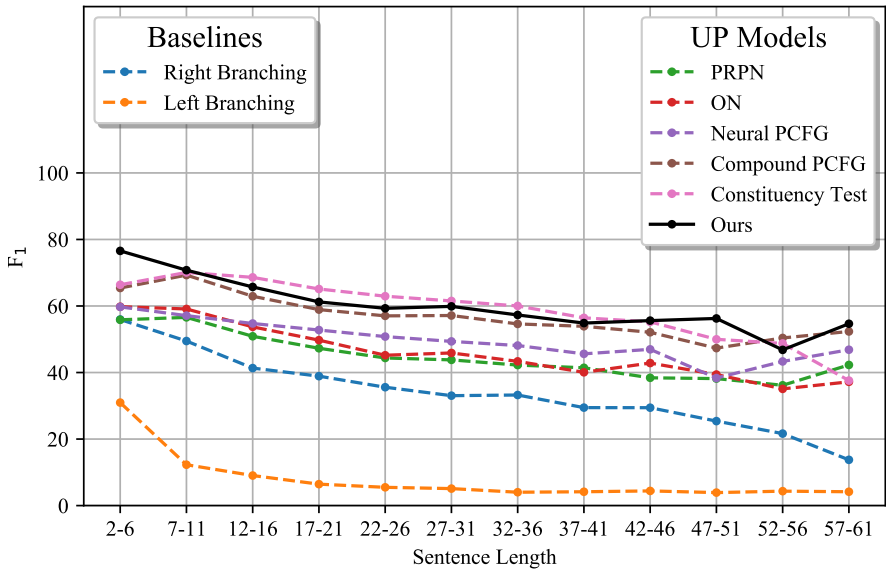

Figure 5.1: F_1 of different models grouped by sentence length on the PTB test set.

Figure 5.1 provides a comparison of the different parsers' performance over varying sentence lengths binned at equal intervals. As we can perceive, right-branching is a strong baseline and is almost comparable to the results of PRPN. Among category (1), we find that ordered neurons (ON) which use gated attention mechanisms to train

[1] https://github.com/yikangshen/PRPN
[2] https://github.com/yikangshen/Ordered-Neurons
[3] https://github.com/harvardnlp/urnng
[4] https://github.com/iesl/diora
[5] https://github.com/iesl/s-diora
[6] https://github.com/harvardnlp/compound-pcfg
[7] https://github.com/yaushian/Tree-Transformer
[8] https://github.com/stevenxcao/constituency-test-parser

Model	**WSJ-Full**		**WSJ-10**	
	Mean	Max	Mean	Max
Trivial Baselines:				
Left Branching (LB)	8.7		17.4	
Balanced	18.5			
Right Branching (RB)	39.5		58.5	
Unsupervised Parsing approaches:				
PRPN† (Shen et al., 2018b)	37.4	38.1	58.4	–
URNNG⋆ (Kim et al., 2019b)	–	45.4	–	–
ON† (Shen et al., 2019)	47.7	49.4	63.9	–
Tree Transformer†⋆ (Wang et al., 2019)	50.5	52.0	66.2	–
Neural PCFG† (Kim et al., 2019a)	50.8	52.6	64.6	–
DIORA⋆ (Drozdov et al., 2019)	–	58.9	60.5	–
Compound PCFG† (Kim et al., 2019a)	55.2	60.1	70.5	–
S-DIORA†⋆ (Drozdov et al., 2020)	57.6	64.0	71.8	–
Constituency Test⋆ (Cao et al., 2020)	62.8	65.9	68.1	–
Ours⋆ (using inside)	55.9	57.2	64.2	–
Ours⋆ (using inside w/ self-training)	61.4	64.2	66.9	–
Ours⋆ (using inside and outside w/ co-training)	**63.1**	66.8	**73.1**	–
Oracle Binary Trees	84.3		82.1	

Table 5.1: Unlabeled sentence-level F_1 on the full as well as sentences of length ≤ 10 of the PTB test set without punctuation or unary chains. We evaluate each model using the evaluation script provided by Kim et al. (2019a) and take the baseline numbers of certain models from (Kim et al., 2019a; Cao et al., 2020). † denotes models trained without punctuation and ⋆ denotes models trained on additional data.

an RNN for inducing trees, achieves a score of 47.7 F_1, whereas, in category (2), we observe that S-DIORA, which is a superior variant of DIORA scores 57.6 F_1 on the PTB test set. When considering category (3), we notice that Compound PCFG which is an extension of Neural PCFG using a continuous latent vector formulated at a sentence-level, attains a score of 55.2 F_1, whereas, in category (4), we see that Constituency Test which trains a RoBERTa model using predefined constituency tests, manages to obtain 62.8 F_1 on the PTB test set. In comparison to the models as mentioned above, our vanilla inside model learns the underlying phenomena of whether a span forms a coherent meaning, can produce competitive results and is already in the range of previous best models like DIORA and Compound PCFG. Our best parser using the inside and outside

models after the co-training procedure, which mainly optimizes for span boundaries of plausible constituents, secures a marginally greater score of 63.1 F_1 on the PTB test set. Although, Shi et al. (2021) obtain strong results by relying on distant supervision using sources that contain naturally-occurring bracketings , we believe the nature and diversity of the datasets (especially the QA-SRL dataset) has a much bigger role to play in boosting scores than on the actual learning process itself. Moreover, a considerable amount of hyperlinks in the data match syntactic constituents, thereby limiting the scope for the actual algorithm to induce meaningful tree structures. Hence, we do not include their results in our analysis. Table 5.1 reports the unlabeled sentence-level F_1 scores on the PTB test set. While a great deal of approaches use additional data, there is a slight difference, however. DIORA and S-DIORA approaches both use context-insensitive vectors to maintain the context-free nature of chart parsing. In contrast, Tree Transformer and Constituency Test approaches rely on the context-sensitive PLMs.

It is worth noting that domain-adaptive pre-training (pre-training RoBERTa model on the PTB training sentences) as well as task-adaptive pre-training (pre-training RoBERTa model on the synthetically created classification data) and further fine-tuning (Gururangan et al., 2020), did not yield substantial gains in the downstream task performance for us.

Next, we take a look at Chinese which exhibits several linguistic differences compared to English such as lesser morphology, more mixed headedness, and lesser proportions of attachment ambiguity, etc. (Levy and Manning, 2003). We note that the scores for all the chosen models are fairly lower than their English counterparts indicating the sheer amount of difficulty in parsing Chinese sentences due to tree-structural differences between the two treebanks. In the case of CTB, our final parser achieves 41.8 F_1 on the CTB test set, which is clearly 5.8 F_1 points greater than the previous best-published result by Compound PCFG. Table 5.2 displays the unlabeled sentence-level F_1 scores on the CTB test set. It is worth noting that we do not fine-tune the hyperparameters while training our parser on any of the languages to prevent overfitting. Furthermore, we did not consider some of the models which was previously chosen for PTB during our analysis as extending those models to include Chinese is non-trivial and not fairly straightforward owing to multiple reasons, e.g., lack of relevant datasets (as DIORA uses SNLI and MultiNLI for training), lack of linguistic knowledge expertise (not easily transferable notion for designing constituency tests), and so on.

Figures A.1 and A.2 in the appendix show step-wise analysis at each stage of the training for a sample sentence taken from the CTB and PTB training set, respectively.

Model	CTB	
	Mean	Max
Trivial Baselines:		
Left Branching (LB)	9.7	
Random Trees	15.7	16.0
Right Branching (RB)	20.0	
Unsupervised Parsing approaches:		
PRPN (Shen et al., 2018b)	30.4	31.5
ON (Shen et al., 2019)	25.4	25.7
Neural PCFG (Kim et al., 2019a)	25.7	29.5
Compound PCFG (Kim et al., 2019a)	36.0	39.8
Ours (using inside)	37.8	38.4
Ours (using inside w/ self-training)	40.6	41.7
Ours (using inside and outside w/ co-training)	**41.8**	43.3
Oracle Binary Trees	81.1	

Table 5.2: Unlabeled sentence-level F_1 on the CTB test set. We evaluate each model using the evaluation script provided by Kim et al. (2019a) and take the baseline numbers also from Kim et al. (2019a).

As we can observe, in contrast to the vanilla inside model, the parser using the inside and outside models after the co-training stage produces fewer crossing brackets.

5.2.2 Left-branching Languages

We next evaluate the efficacy of our approach on languages with left-branching tendencies and consider the Japanese Treebank (KTB) for our analysis. One of the main differences between English and Japanese grammar is that in the latter the object comes before the verb which makes parsing challenging. Consequently, there warrants an exploration to assess the generalization capabilities of our chosen models for a strictly left-branching language. As we can observe, left-branching is a strong baseline and is roughly on par with the results of PRPN, which has an innate right-branching bias. Our final parser obtains 39.2 F_1 on the KTB test set which surpasses the previous benchmarks. Table 5.3 shows the Evalb F_1 scores on the KTB test set. We further note that sentences in KTB do contain null elements and these artifacts usually start with the "*" marker, which we remove to maintain the unsupervised nature of our experiments.

Model	KTB-40		KTB-10	
	Mean	Max	Mean	Max
Trivial Baselines:				
Left Branching (LB)	29.4		51.6	
Right Branching (RB)	9.8		22.9	
Unsupervised Parsing approaches:				
PRPN (Shen et al., 2018b)	27.2	31.8	30.1	33.6
URNNG (Kim et al., 2019b)	10	10.2	22.7	22.7
DIORA (Drozdov et al., 2019)	24.9	26.0	42.3	43.3
DIORA-all (Hong et al., 2020)	36.4	40.0	47.1	48.9
Ours (using inside)	33.7	36.3	53.8	55.9
Ours (using inside w/ self-training)	37.6	39.8	55.5	58.2
Ours (using inside and outside w/ co-training)	**39.2**	41.1	**56.7**	59.1
Upper Bound	76.5		76.6	

Table 5.3: Evalb F_1 on the full (F_1-all) and length ≤ 10 (F_1-10) sentences of the KTB test set discarding punctuation corresponding to KTB-40 and KTB-10, respectively. We take the baseline numbers of models from Li et al. (2020b). See Table 2.1 to view the hyperparameters used for `evalb`.

Based on our findings, we hypothesize that a monolingual model trained on a corpus with an extensive vocabulary size for a given language performs substantially better than multilingual models, such as XLM-RoBERTa (Conneau et al., 2020). We attribute the degradation in performance for multilingual models due to a likely misalignment in the vector space between languages. Figure A.3 in the appendix shows a step-wise analysis at each stage of the training for a sample sentence taken from the KTB training set.

5.3 Analyses

This section describes the effects of our algorithm's three main components: self-training, co-training, and distituent selection. Furthermore, to better understand the types of errors that the system is or is not making, we perform linguistic error analysis and provide suitable explanations and alternatives to rectify such errors.

5.3.1 Effect of Self-Training

PLMs that capture rich contextualized textual representations can help in parsing when the sheer amount of unlabeled data is enormous. For this reason, we expect that self-training in addition to pre-training adds no new information to the fine-tuned parser. However, we find that self-training boosts the parser's performance by about 9.8%, reassuring that self-training complements unsupervised pre-training. Table 5.4 shows a fine-grained analysis corresponding to different stages of self-training for the inside model in the case of PTB. As can be inferred, self-training improves the performance of the inside model by 5.5 F_1 points. There is one caveat, though. For our experiments, we use the PTB test set instead of the PTB validation set to avoid fine-tuning and thereby overfitting on the test set based on the feedback obtained from the validation set. Additionally, it helps to keep the nature of our experiments *purely* unsupervised. We do note that, however, when evaluated individually on the validation set, we find the results comparable to that of the test set.

Model	**#ST-steps**			
	0	1	2	3
Inside	55.9	57.7	59.5	61.4

Table 5.4: Unlabeled sentence-level F_1 on the full PTB test set after applying the iterative self-training algorithm on the inside model.

5.3.2 Effect of Co-training

The process of creating two conditionally independent views to integrate multi-view information is paramount. One of the options would be to concatenate both the inside and outside vectors during training and inference. With this approach, we see negligible improvement, scoring a meager 13.2 F_1 on the PTB test (without self-training). Hence, we verify the effectiveness of co-training compared to simply concatenating: the simple concatenation strategy cannot fully harvest the information from each view and tends to make the learning problem on the downstream task intractable. The idea of separating the two models for co-training is to learn constituent boundaries to identify the splitting points in a sentence through independent data views. After co-training, the performance of the inside-outside joint model increases by 1.7 F_1 points, as shown in Table 5.5 with improvements at each step. In comparison to using self-training, we observe that the

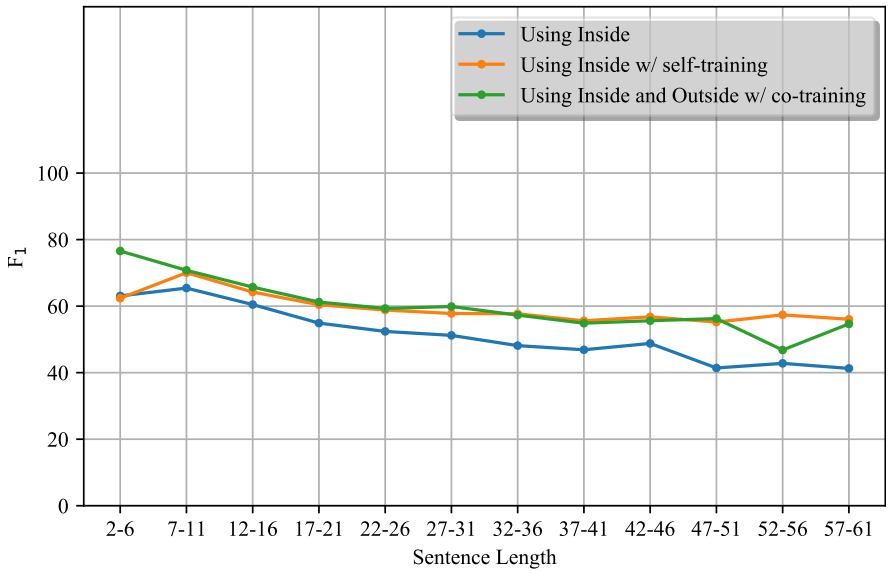

Figure 5.2: F_1 grouped by sentence length on the PTB test set for different strategies.

gain is not significant after co-training. We hypothesize that when the initial seed data is limited, co-training can address the problem of coverage. As the amount of seed data increases, the superiority of co-training gradually reduces. Besides, the inside vectors (built upon transformer architecture) inherently possess contextual knowledge due to being trained on a large corpus and having a high likelihood of adding no new information during co-training. Figure 5.2 shows the plots corresponding to the effect of sentence length on the model performance under the three settings: (1) using only inside, (2) using inside with self-training, and (3) using inside and outside with co-training strategies.

Model	**#CT-steps**		
	0	1	2
Inside and Outside	61.4	62.9	63.1

Table 5.5: Unlabeled sentence-level F_1 on the full PTB test set after applying the iterative co-training algorithm on the joint inside and outside models.

5.3.3 Effect of Distituent Selection

It is quite evident from Section 4.2 that our algorithm injects language-specific biases during the seed bootstrapping step. We further explore two settings on the PTB —

random and left-branching bias, to assess the extent to which the selection strategy of the disitituent affects the parser's performance: (1) In the random setting, we select distituents from the slice `(start:r)`, where *r* is a random number generated between `start+1` and `end-1`, both inclusive, which produces 19.3 F_1 for the inside model; (2) In the left-branching bias setting, we prepare the seed bootstrapping similar to KTB (a left-branching treebank; as explained in Section 4.2.2), which results in 11.2 F_1 for the inside model. Therefore, the method in which we carry out the initial seed classification strongly influences the predicted parse trees.

5.3.4 Linguistic Error Analysis

Conventionally, we evaluate the performance of a parser using a single metric, F_1 score. Although a robust metric, it does not provide any additional information about the source of remaining errors that are linguistically meaningful. In the case of PTB, as depicted in Table 5.6, we notice our model achieves high accuracy scores while predicting all the phrase types except for the adjective phrase (ADJP). To classify errors, we rank the set of node errors according to their frequencies and place them into buckets. We find prepositional phrase (PP) attachment to be the most significant contributor to the overall error. We further classify each bucket as one of the error types and provide possible explanations for some of them below:

Bracketing inner NP of a definite Noun Phrase. When a definite article links with a singular noun, there is a need to shelve the inner spans for accommodating the larger span having the definite article, e.g., *the* [*stock market*]

Grouping NP too early overlooking broader context. Before the training step, the parser collapses infrequent tokens to a single form. As a result, it aggressively groups rare words in the corpus. Either or both of building an improved outside model and tuning the vocabulary size can alleviate these type of errors to a considerable extent, e.g., *Shearson* [*Lehman Hutton*] *Inc.*

Omitting conjunction joining two phrases. The parser demonstrates poor signs of understanding of co-ordination cases in which conjunction is a direct sibling of the nodes being shifted, or is the leftmost or rightmost node being shifted, e.g., *Notable* [*&* *Quotable*]

Confusing contractions with Possessives. Since many contraction phrases like {*they're*, *it's*} exist in PTB, the parser often mistakes them with Possessive NPs, causing undesired splitting. Expanding the contractions can be a good first step to correct these

	PRPN	ON	URNNG	Compound PCFG	S-DIORA	Constituency Test	Our Best Parser
SBAR	50.0	51.2	74.8	56.1	59.2	66.1	**81.7**
NP	59.2	64.5	39.5	74.7	78.0	**79.4**	73.5
VP	46.7	41.0	76.6	41.7	**78.9**	68.2	70.4
PP	57.2	54.4	55.8	68.8	67.1	**86.2**	77.8
ADJP	44.3	38.1	33.9	40.4	49.1	**62.6**	40.9
ADVP	32.8	31.6	50.4	52.5	59.9	63.9	**70.4**

Table 5.6: Average recall per constituent category (i.e. label recall) in (%). We take the results of PRPN, ON, URNNG, and Compound PCFG from Kim et al. (2019a), S-DIORA from Drozdov et al. (2020), and Constituency Test from Cao et al. (2020).

systematic errors, e.g., *the company* [*'s $ 488 million in 1988*]

In the future, we would like to build comprehensive error analysis protocols for both CTB and KTB, supplemented by feedback obtained from the respective language experts. In addition, we can make use of a supervised constituency parser to synthetically label the predicted parse trees and then run the Berkeley parser analyzer (Kummerfeld et al., 2012) to quantitatively understand the various linguistic error types as shown by Drozdov et al. (2020).

5.3.5 Unsupervised Labeled Parsing

We further investigate unsupervised labeled constituency parsing to see if the parser can identify linguistically pertinent constituent spans and extract labels such as Noun Phrases (NPs) and Verb Phrases (VPs). We typically evaluate labeled parsing by checking if a span associates with the correct label. We can effectively induce span labels using our methodology after extracting the learned representations from the inside and outside strings and later clustering. When labeling a gold bracket, our method achieves 61.2 F_1 on the full PTB test set and is comparable with the current best model, DIORA (Drozdov et al., 2019). Figure B.1 visualizes the alignment of induced clusters and linguistic gold-standard labels using a heatmap. We note that since RoBERTa outputs subword vectors which do not conform strictly to word-level vectors, we further aggregate these vectors with mean-pooling to achieve a word-level representation using Sentence Transformers (Reimers and Gurevych, 2019).[9] We select 600 codes while

[9] https://github.com/UKPLab/sentence-transformers

doing the clustering initially, such that we are left with about 25–30 clusters after the most common label assignment process, i.e., the number of unique phrase types. The phrase clusters are assigned to: $NP_{(7)}$, $PP_{(5)}$, $WHPP_{(3)}$, $ADVP_{(3)}$, $ADJP_{(2)}$, $S_{(2)}$, $WHADVP_{(1)}$, $UCP_{(1)}$, $VP_{(1)}$, $PRN_{(1)}$, $QP_{(1)}$, $SBAR_{(1)}$, $WHNP_{(1)}$, $CONJP_{(1)}$ according to the majority gold labels present in the chosen cluster. In other words, these 14 assigned phrase types correspond with the 14 most frequent labels. Table B.1 lists the induced non-terminal grouped across different clusters and also its correctness in identifying the gold labels. A future work can be in the direction of creating a single model capable of achieving both bracketing and labeling at one pass and the possibility to explore multilingually. Further, these induced labels can function as additional features for the inside and outside models to achieve even better predictive power.

5.4 Limitations of this Work

This study contains several drawbacks, some of which are intentional and help to define the focus of the investigation, while others are more troublesome.

Assumptions about Language Branching. Language has a tendency to be consistently right-branching or consistently left-branching or branch about halfway in between. In the case of a right-branching language (such as English), the head of the sentence is typically at the beginning, and a sequence of modifiers usually follows it (head-initial). As a result, the parse trees grow down and to the right (e.g. *the cat who was waiting at the doorstep*). In contrast, for a left-branching language (like Japanese), the modifier usually precedes the head (head-final) and generates parse trees that grow down and to the left (e.g. *who was waiting at the doorstep, the cat*). The prior knowledge about the linguistic concept of branching has an impact on our seed bootstrapping procedure and a tendency to create a branching bias (see Section 4.2 for specific details). Furthermore, there exists a lack of support for languages with mixed branching. In principle, we can fix a single branching strategy by reversing the word order (e.g. right-to-left instead of left-to-right) and aligning the words in the final parse to match that of the original input sentence. We argue that such right-skewness biases are prevalent in most of the recent works which rely on the notion of syntactic depth to recover phrase-structure trees (Dyer et al., 2019).

Additional Heuristics for PTB. While our approach is standalone in nature and fully capable of producing strong performance on all the evaluated treebanks which is

comparable to the recent systems (Kim et al., 2019b,a; Drozdov et al., 2019), we assert that in order to achieve the state-of-the-art performance for PTB, there is a need to inject a small amount of heuristics making it not purely-supervised (see Section 4.3.1.1 for exact details). However, the contributions from these heuristics are much smaller in comparison to the semi-supervised learning approaches combined. We merely treat these heuristics as a post-processing step for refinement purposes after identifying the most common false constituents in the PTB development set.

Unable to Induce Grammar. Another shortcoming of our approach is that the parser is grammarless. In other words, we do not rely on a grammar model to generate rule probabilities for inducing the bracketing structure of a sentence. However, in the recent past, there has been rapid progress in studying the induced grammar's ability to parse sentences by employing neural networks to learn grammar rules (e.g. context-free rewrite rules) implicitly (Jin et al., 2019; Kim et al., 2019a; Zhu et al., 2020; Yang et al., 2021).

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we presented the readers with a comprehensive overview of unsupervised parsing as well as described our contributions to the development of this field.

In Chapter 2, we walked through the history of unsupervised parsing, which dates back to the 1990s. Since 2015, the field has been revolutionized by the general trend in modeling linguistic structures using neural models. We formally stated the task definition and explained the evaluation protocols. In addition, we pointed out unfair comparisons prevalent among today's systems and posited a need to enforce a standard protocol both during training and inference.

In Chapter 3, we proposed a neural unsupervised parsing framework to learn constituency trees from raw sentences. Unlike the previously proposed models such as the Constituent-Context Model (CCM; Klein and Manning 2002), the Dependency Model with Valence (DMV; Klein and Manning 2005), and Unsupervised Maximum Likelihood estimator for Data-Oriented Parsing (UML-DOP; Bod 2006), which explicitly required the parts of speech (POS) of the words as input both during training and inference, we argue ours is a more realistic setup as we have not made use of any POS tags. The training pipeline consisted of mainly three stages: adoption of a seed bootstrapping technique, relying on inside and outside strings drawing close connections from spectral learning, and further refinement of the proxy-labels learnt through the employment of self-training and co-training strategies. In addition, we applied the CKY dynamic programming parsing algorithm to calculate the Viterbi tree with the help of a chart table and obtain the best parse.

In Chapter 4, we reviewed basic terminologies around PLMs, specifically, the

transformer architecture, including various components like self-attention, multi-head attention, positional embeddings, and layer normalization. Next, we explained salient information about BERT whose invention could very well be regarded as the "ImageNet moment" in NLP, and further described the variant which served as the backbone model in our approach, the RoBERTa. We concisely provided details about the kind of syntactic knowledge present in these PLMs, especially BERT and its variants. Finally, we focused on the inner workings and practical implementation of our algorithm.

In Chapter 5, we performed a comprehensive empirical study to portray the effectiveness of our approach. We showed that our parser could generalize multilingually to treebanks of both right-branching and left-branching languages with a minimal change. We also analyzed the effect of distituent selection and performed an exhaustive linguistic error analysis to study the common parsing errors. In addition, we examined the impact of self-training and co-training strategies on the overall performance for different modeling phases of training. Furthermore, we explored unsupervised labeled parsing to induce span labels from the inside and outside vectors. Lastly, we highlighted few areas of concern which become potential limitations and shall be addressed in our future work.

6.2 Future Work

Despite the recent success in unsupervised parsing, we still have no acceptable system that can shed some light on the children's discovery of structure in language. We argue that in order to achieve this goal, the focus needs to shift from incentivizing the creation of more accurate parsers to actually learning the hierarchical structures in a meaningful manner.

Humans and NLP systems process different amounts and nature of the training data. For instance, PLMs are typically trained on Wikipedia, books, articles, and other corpora; the syntactic properties acquired by these models are significantly different compared to the child-directed speech picked up by humans. Therefore, there is often a limit associated with learning from these datasets alone. Moreover, PLMs generally rely only on the textual component to learn language-related phenomena, which is in stark contrast to the visual perception and social interaction through which humans learn language. Consequently, it becomes vital to integrate and jointly learn from other data modalities while designing unsupervised parsing systems. Visually-grounded grammar induction and the establishment of large-scale datasets for its evaluation purposes is an

important direction in this regard (Shi et al., 2019; Zhao and Titov, 2020; Hong et al., 2021). Recently, there has been considerable progress in creating language models from raw audio as input, e.g., Textless NLP[1], that can learn aspects of natural language such as accent, tone, prosody, expression, pitch, timbre, etc., beyond text. In the modern era of PLMs, the ability to incorporate parse tree structures on downstream tasks has been severely limited in applications. However, recent studies (Bai et al., 2021; Sachan et al., 2021) have shown the inclusion of syntactical information from dependency or syntax trees has been proven to be beneficial for information retrieval tasks of semantic role labeling (SRL) and named entity recognition (NER). Furthermore, there is a desire to achieve out-of-distribution generalization as well as domain adaptation. The PTB dataset consists of multiple genres and has universally been referred to as the prototypical news domain. It would be interesting to know where the current unsupervised parsing models stand when evaluated on non-canonical data arising from social media and dialogue systems. One other possible direction for future research is to explore few-shot parsing techniques to emphasize the diversity of the few labeled examples in the training dataset to yield maximum performance. Besides, from an engineering standpoint, Few-shot parsing (refer Section 2.3) produces substantially better results and is workable compared to fully unsupervised parsing. Along the same lines, it has been shown that combining few-shot parsing and self-training approaches can further boost performance for all models (Shi et al., 2020). Further, understanding the kind of supervision one requires to solve an intermediate auxiliary task not dependent on theory explicitly and developing models grounded in basic linguistic theory can be a rising direction for future work. Lately, many top-scoring models seem to adopt this strategy (Cao et al., 2020; Shi et al., 2021). Lastly, the research area of multilingual and code-switched unsupervised parsing needs additional spotlight, with greater reason, since creating an unsupervised parser for low-resource languages is one of the field's main motivations (see Section 1.1).

We hope to see newer research directions emerge in the field of unsupervised parsing in the upcoming years and much focus to shift to some of the lesser spoken languages out of about 7,000 languages in the world. Besides, from the viewpoint of internet of tiny things, multi-lingual unsupervised parsing using a single model has the added advantage of removing the additional overhead due to network latency and also consuming lesser memory as they have fewer trainable parameters.

[1] https://ai.facebook.com/blog/textless-nlp-generating-expressive-speech-from-raw-audio

Appendix A

Parse Trees

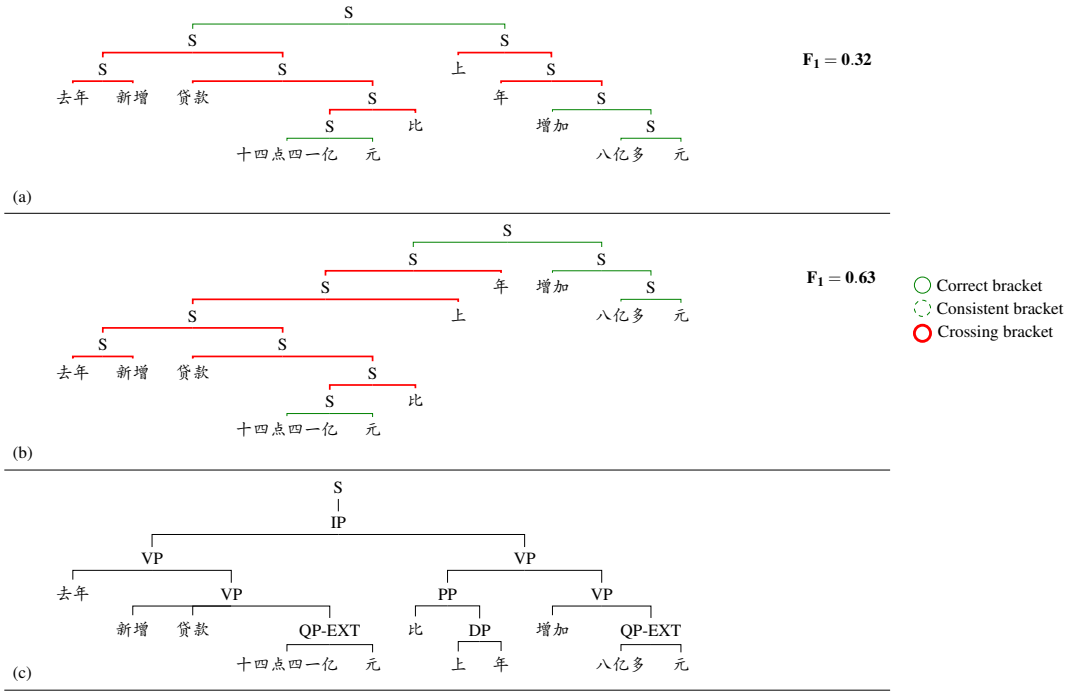

Figure A.1: Example tree taken from the CTB training set. After the co-training procedure (b), the parser correctly identifies constituents "十四点四一亿元", "新增贷款十四点四一亿元", and "去年新增贷款十四点四一亿元" compared to the previous step using the inside model (a). It only makes 3 errors due to crossing brackets at "贷款十四点四一亿元", "年增加八亿多元", and "上年增加八亿多元".

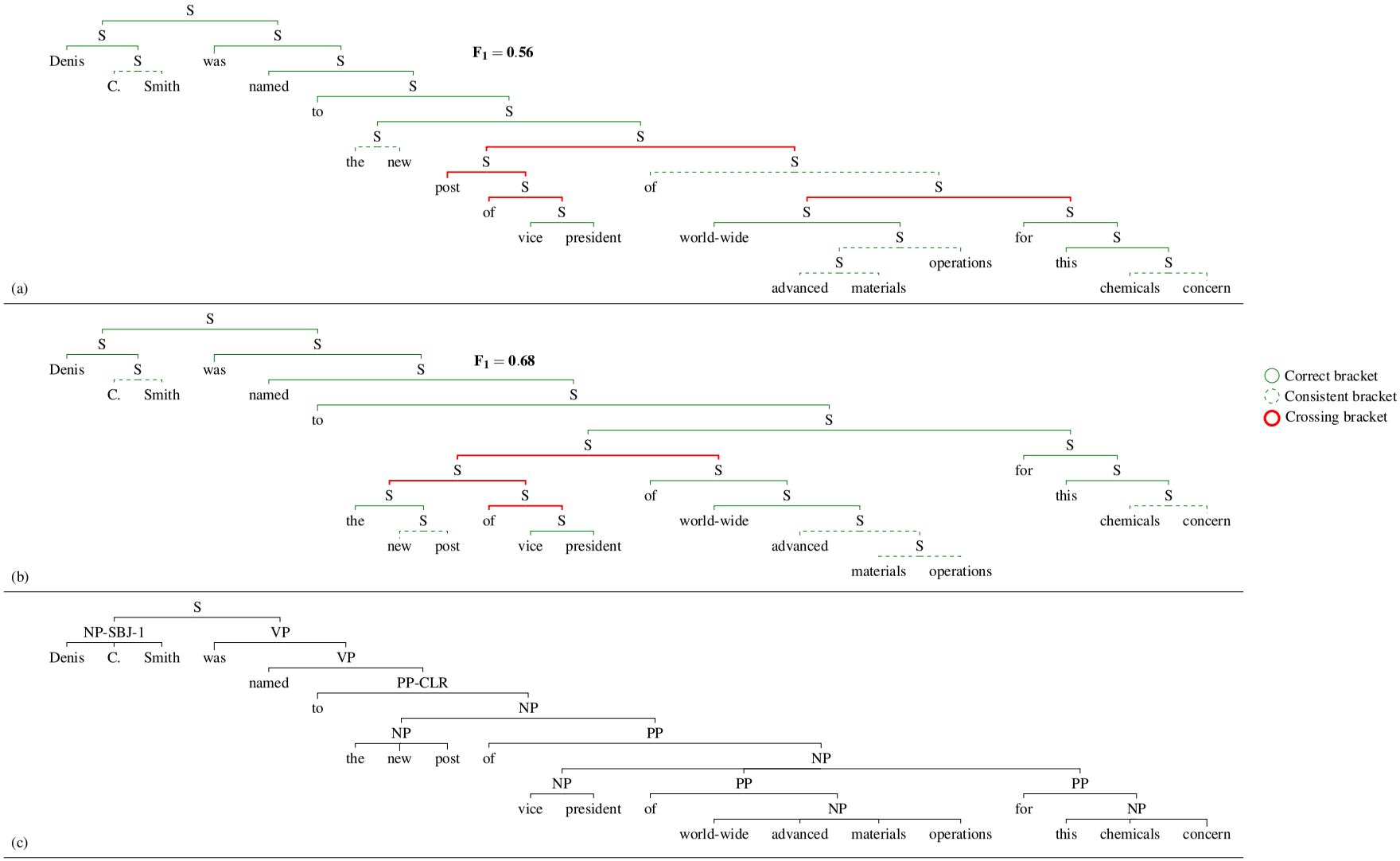

Figure A.2: Displays the parse tree output for a sample sentence: (a) Using Inside (b) Using Inside and Outside (c) Gold Tree. After the co-training procedure (b), the parser correctly identifies constituents *"the new post"* and *"of world-wide advanced materials operations"* which were earlier identified as distituents by the inside model (a). It makes two errors due to crossing brackets - namely *"of vice president"*, *"the new post of vice president"*, and *"the new post of vice president of world-wide advanced materials operations"*.

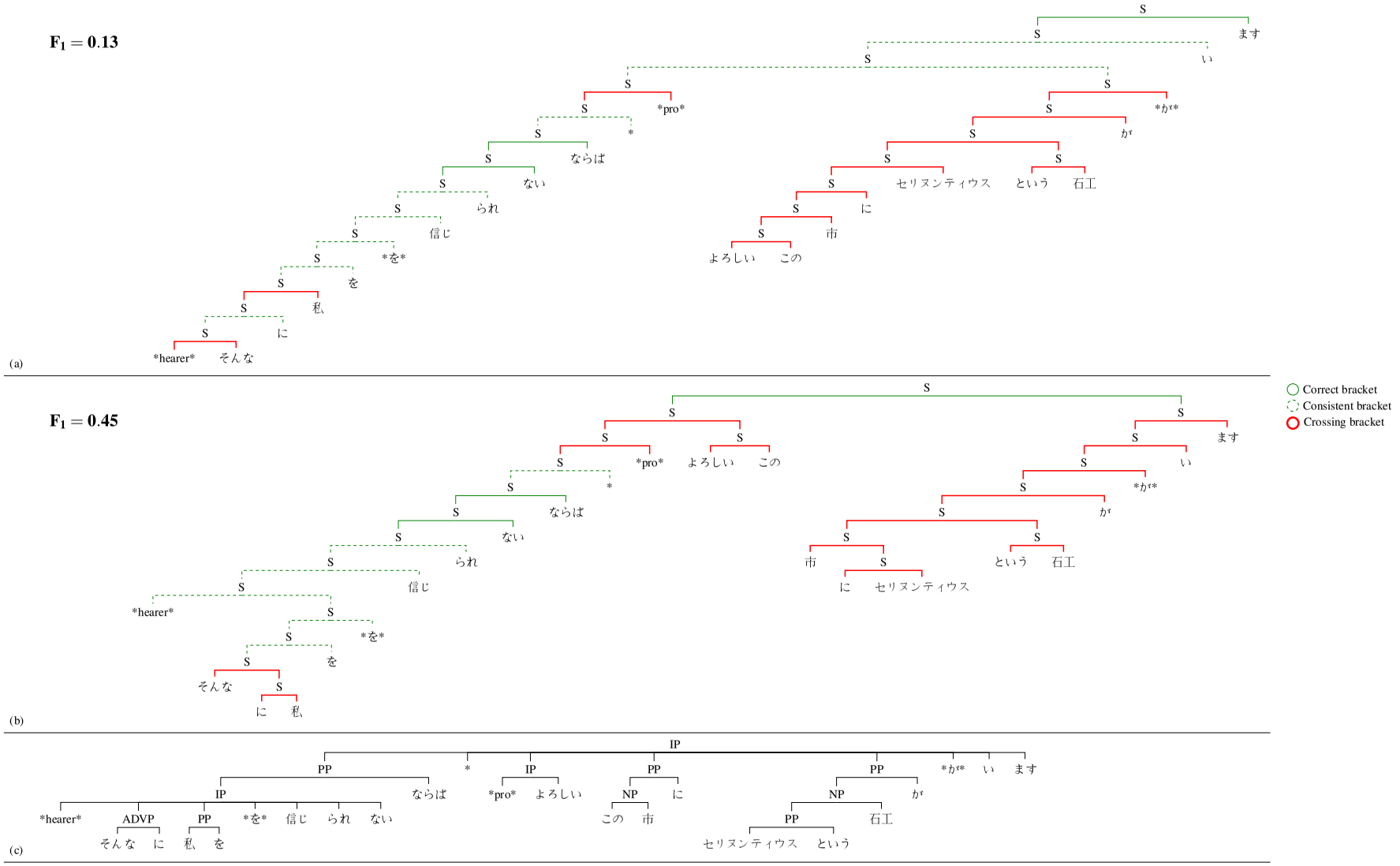

Figure A.3: Example tree taken from the KTB training set. After the co-training procedure (b), the parser correctly identifies constituents “そんなに”, “私を”, “*hearer* そんなに私を*を* 信じられないならば”, “*pro* よろしい”, “この市”, and “この市に”, while incorrectly tagging “セリヌンティウスという石工が” as a distituent compared to the previous step using the inside model.

Appendix B

Constituent Labels & Cluster Analysis

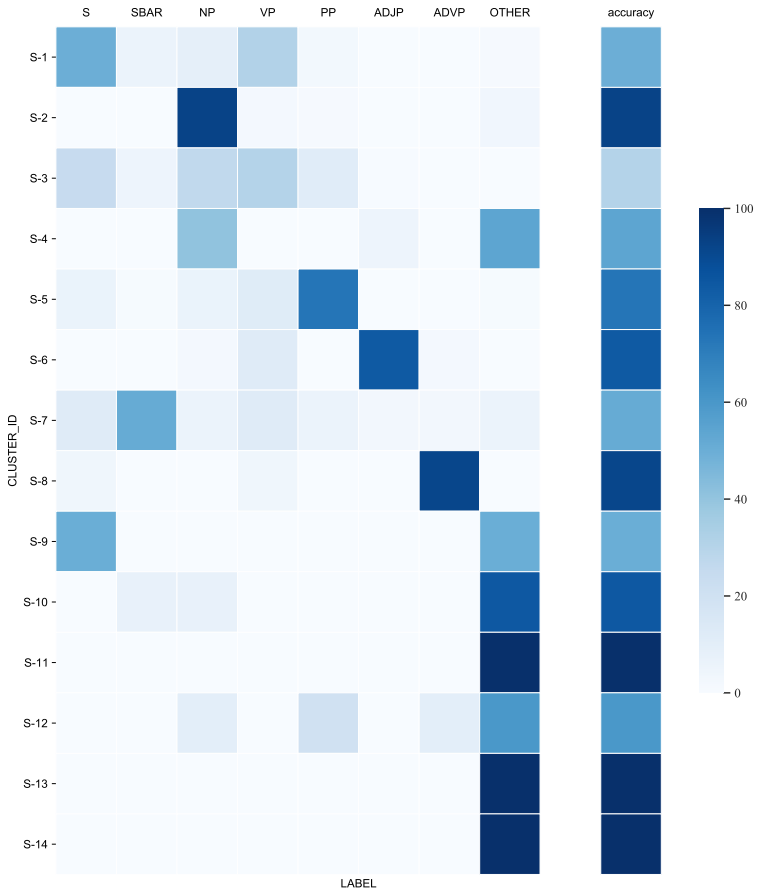

Figure B.1: Alignment between induced and gold labels of the top-performing clusters. We cluster the constituent inside vectors from the ground truth parse (without labels) using the K-Means algorithm and assign each constituent with the most common label inside its cluster. We define accuracy as the probability of correctly predicting the most common label.

Cluster ID	Label	Constituent	Predicted	Status
0	NP	the space shuttle Atlantis	NP	✓
	NP	Once the chief beneficiaries	NP	✓
	PP	in the offing	NP	✗
	PP	in the thrift	NP	✗
	S	the dollar was weak	NP	✗
	SBAR	If the new Cheer sells well	NP	✗
1	ADJP	higher than most anticipated	NP	✗
	NP	more than one billion Canadian dollars 851 mil...	NP	✓
	QP	at least 600 to 700	NP	✗
12	NP	A. Boyd Simpson	NP	✓
	NP	Justice John Harlan	NP	✓
	NP	Robert D. Cardillo	NP	✓
	NP	James D. Awad	NP	✓
	NP	Clark S. Spalsbury Jr	NP	✓
	NP	L.J. Hooker	NP	✓
30	NP	one 's testimony	NP	✓
	NP	the stock market 's plunge Friday	NP	✓
	PP	in the market 's decline	NP	✗
75	ADVP	two years ago	ADVP	✓
	ADVP	two weeks ago	ADVP	✓
	PP	just like two years ago	ADVP	✗
	PP	between now and two years ago	ADVP	✗
310	NP	action on capital gains	VP	✗
	NP	the three airlines being dropped	VP	✗
	NP	news footage of the devastated South Bronx	VP	✗
	NP	the prospect of a fight with GEC for Ferranti	VP	✗
	PP	before declining again trapping more investors	VP	✗
	S	This small Dallas suburb 's got trouble	VP	✗
	S	the earnings picture confuses	VP	✗
	SBAR	it acquired 5 % of the shares in Jaguar PLC	VP	✗
	SBAR	the market is going through another October '87	VP	✗
	VP	may be dubbed Eurodynamics	VP	✓
	VP	resuscitate the protagonist of his 1972 work A...	VP	✓
	VP	said after the 1987 crash	VP	✓
	VP	has a base of 100 set in 1983	VP	✓
514	NP	its two classes of preferred stock	PP	✗
	NP	Oil company refineries	PP	✗
	PP	to depository institutions	PP	✓
	PP	of Remic mortgage securities	PP	✓
	PP	of the preferred-share issue	PP	✓
	PP	in the patent-infringement proceedings	PP	✓
	PP	of mainframe computers	PP	✓
	PP	from mature conventional fields in western Canada	PP	✓
	PP	of its North American vehicle capacity	PP	✓
	VP	have big commodity-chemical operations	PP	✗
533	NP	Bateman Eichler Hill Richards	NP	✓
	NP	KLM Royal Dutch Airlines	NP	✓
	NP	owners Anna and Morris Snezak	NP	✓
	NP	Mehta & Isaly	NP	✓
	PP	at Hambrecht & Quist in San Francisco	NP	✗

Table B.1: Investigation of phrase clusters that shows several syntactic properties. Clearly, there are patterns surrounding identification of people/organization names, time-related signals, quantities, etc.

Bibliography

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450. (page 30).

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 the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3011–3020, Online. Association for Computational Linguistics. (page 55).

J. K. Baker. 1979. Trainable grammars for speech recognition. In *Speech communication papers presented at th 97th Meeting of the Acoustical Society of America*, pages 547–550, Boston, MA. (page 11).

E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*. (page 14).

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, page 92–100, New York, NY, USA. Association for Computing Machinery. (page 20).

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 865–872, Sydney, Australia. Association for Computational Linguistics. (pages 8, 16, 17, 53).

Antoine Bosselut. 2020. *Understanding Natural Language with Commonsense Knowledge Representation, Reasoning, and Simulation*. University of Washington. (page 30).

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics. (page 16).

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc. (page 32).

Alastair Butler, Tomoko Hotta, Ruriko Otomo, Kei Yoshimoto, Zhen Zhou, and Hong Zhu. 2012. Keyaki treebank: phrase structure with functional information for japanese. In *Proceedings of Text Annotation Workshop*. (page 41).

Steven Cao, Nikita Kitaev, and Dan Klein. 2020. Unsupervised parsing via constituency tests. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4798–4808, Online. Association for Computational Linguistics. (pages 11, 16, 17, 28, 40, 42, 43, 50, 55).

Glenn Carroll and Eugene Charniak. 1992. *Two experiments on learning probabilistic dependency grammars from corpora*. Department of Computer Science, Univ. (page 8).

Ciprian Chelba and Frederick Jelinek. 1998. Exploiting syntactic structure for language modeling. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*. (page 2).

Ciprian Chelba, Tomás Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 2635–2639. ISCA. (page 16).

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics. (page 41).

Jihun Choi, Kang Min Yoo, and Sang goo Lee. 2018. Learning to compose task-specific tree structures. In *AAAI*. (page 11).

Noam Chomsky et al. 2006. *Language and mind*. Cambridge University Press. (page 4).

Alexander Clark and Nathanaël Fijalkow. 2020. Consistent unsupervised estimators for anchored PCFGs. *Transactions of the Association for Computational Linguistics*, 8:409–422. (page 21).

John Cocke. 1969. *Programming languages and their compilers: Preliminary notes*. New York University. (page 27).

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 50–61, Edinburgh, Scotland, UK. Association for Computational Linguistics. (page 8).

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2012. Spectral learning of latent-variable PCFGs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 223–231, Jeju Island, Korea. Association for Computational Linguistics. (pages 10, 21).

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2013. Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 148–157, Atlanta, Georgia. Association for Computational Linguistics. (page 21).

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2014. Spectral learning of latent-variable pcfgs: Algorithms and sample complexity. *Journal of Machine Learning Research*, 15(69):2399–2449. (page 21).

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. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics. (page 46).

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 600–609, Portland, Oregon, USA. Association for Computational Linguistics. (page 8).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics. (pages 10, 32).

Laurent Dinh, David Krueger, and Yoshua Bengio. 2015. NICE: non-linear independent components estimation. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*. (page 9).

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc. (page 10).

Andrew Drozdov, Subendhu Rongali, Yi-Pei Chen, Tim O'Gorman, Mohit Iyyer, and Andrew McCallum. 2020. Unsupervised parsing with S-DIORA: Single tree encoding for deep inside-outside recursive autoencoders. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4832–4845, Online. Association for Computational Linguistics. (pages 11, 15, 16, 17, 28, 42, 43, 50).

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 the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1129–1141, Minneapolis, Minnesota. Association for Computational Linguistics. (pages 11, 15, 16, 17, 28, 42, 43, 46, 50, 52).

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics. (page 9).

Chris Dyer, Gábor Melis, and Phil Blunsom. 2019. A critical analysis of biased parsers in unsupervised parsing. *CoRR*, abs/1909.09428. (pages 15, 51).

Allyson Ettinger. 2020. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48. (page 34).

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334, Boulder, Colorado. Association for Computational Linguistics. (page 1).

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *ArXiv*, abs/1901.05287. (pages 16, 34).

Dave Golland, John DeNero, and Jakob Uszkoreit. 2012. A feature-rich constituent context model for grammar induction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 17–22, Jeju Island, Korea. Association for Computational Linguistics. (page 8).

Joshua Goodman. 1996. Parsing algorithms and metrics. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 177–183, Santa Cruz, California, USA. Association for Computational Linguistics. (page 28).

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for*

Computational Linguistics, pages 8342–8360, Online. Association for Computational Linguistics. (page 44).

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society. (page 30).

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics. (pages 16, 33).

Ruyue Hong, Jiong Cai, and Kewei Tu. 2020. Deep inside-outside recursive autoencoder with all-span objective. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3610–3615, Barcelona, Spain (Online). International Committee on Computational Linguistics. (pages 11, 46).

Yining Hong, Qing Li, Song-Chun Zhu, and Siyuan Huang. 2021. Vlgrammar: Grounded grammar induction of vision and language. *ArXiv*, abs/2103.12975. (page 55).

Phu Mon Htut, Kyunghyun Cho, and Samuel Bowman. 2018. Grammar induction with neural language models: An unusual replication. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4998–5003, Brussels, Belgium. Association for Computational Linguistics. (page 15).

Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R. Bowman. 2019. Do attention heads in BERT track syntactic dependencies? *CoRR*, abs/1911.12246. (page 33).

Yun Huang, Min Zhang, and Chew Lim Tan. 2012. Improved constituent context model with features. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 564–573, Bali, Indonesia. Faculty of Computer Science, Universitas Indonesia. (page 8).

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics. (page 16).

Lifeng Jin, Finale Doshi-Velez, Timothy Miller, William Schuler, and Lane Schwartz. 2018a. Depth-bounding is effective: Improvements and evaluation of unsupervised PCFG induction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2721–2731, Brussels, Belgium. Association for Computational Linguistics. (page 9).

Lifeng Jin, Finale Doshi-Velez, Timothy Miller, William Schuler, and Lane Schwartz. 2018b. Unsupervised grammar induction with depth-bounded PCFG. *Transactions of the Association for Computational Linguistics*, 6:211–224. (page 9).

Lifeng Jin, Finale Doshi-Velez, Timothy Miller, Lane Schwartz, and William Schuler. 2019. Unsupervised learning of PCFGs with normalizing flow. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2442–2452, Florence, Italy. Association for Computational Linguistics. (pages 9, 17, 52).

Lifeng Jin and William Schuler. 2020. Grounded PCFG induction with images. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 396–408, Suzhou, China. Association for Computational Linguistics. (page 12).

Tadao Kasami. 1966. An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report no. R-257*. (page 27).

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 *International Conference on Learning Representations*. (page 10).

Yoon Kim, Chris Dyer, and Alexander Rush. 2019a. Compound probabilistic context-free grammars for grammar induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2369–2385, Florence, Italy. Association for Computational Linguistics. (pages 9, 10, 12, 15, 16, 17, 28, 40, 42, 43, 45, 50, 52).

Yoon Kim, Alexander Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019b. Unsupervised recurrent neural network grammars. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1105–1117, Minneapolis, Minnesota. Association for Computational Linguistics. (pages 9, 16, 28, 42, 43, 46, 52).

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics. (page 15).

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 478–485, Barcelona, Spain. (page 8).

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of*

the Association for Computational Linguistics, pages 128–135, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics. (pages 8, 9, 16, 17, 53).

Dan Klein and Christopher D. Manning. 2005. Natural language grammar induction with a generative constituent-context model. *Pattern Recognit.*, 38:1407–1419. (pages 16, 53).

Noriyuki Kojima, Hadar Averbuch-Elor, Alexander Rush, and Yoav Artzi. 2020. What is learned in visually grounded neural syntax acquisition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2615–2635, Online. Association for Computational Linguistics. (page 12).

Solomon Kullback and Richard A. Leibler. 1951. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86. (page 25).

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the Wall Street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1048–1059, Jeju Island, Korea. Association for Computational Linguistics. (page 50).

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. (page 7).

Roger Levy and Christopher D. Manning. 2003. Is it harder to parse Chinese, or the Chinese treebank? In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 439–446, Sapporo, Japan. Association for Computational Linguistics. (page 44).

Bowen Li, Taeuk Kim, Reinald Kim Amplayo, and Frank Keller. 2020a. Heads-up! unsupervised constituency parsing via self-attention heads. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 409–424, Suzhou, China. Association for Computational Linguistics. (pages 10, 28).

Bowen Li, Lili Mou, and Frank Keller. 2019. An imitation learning approach to unsupervised parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3485–3492, Florence, Italy. Association for Computational Linguistics. (pages 11, 15).

Jun Li, Yifan Cao, Jiong Cai, Yong Jiang, and Kewei Tu. 2020b. An empirical comparison of unsupervised constituency parsing methods. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3278–3283, Online. Association for Computational Linguistics. (pages 15, 40, 46).

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open sesame: Getting inside BERT's linguistic knowledge. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 241–253, Florence, Italy. Association for Computational Linguistics. (page 33).

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692. (pages 23, 32).

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. (page 41).

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics. (page 14).

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 75–82, Ann Arbor, Michigan. Association for Computational Linguistics. (page 10).

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA. Association for Computational Linguistics. (page 19).

David McClosky, Eugene Charniak, and Mark Johnson. 2008. When is self-training effective for parsing? In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 561–568, Manchester, UK. Coling 2008 Organizing Committee. (page 19).

Anhad Mohananey, Katharina Kann, and Samuel R. Bowman. 2020. Self-training for unsupervised parsing with PRPN. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 105–110, Online. Association for Computational Linguistics. (page 20).

Ankur P. Parikh, Shay B. Cohen, and Eric P. Xing. 2014. Spectral unsupervised parsing with additive tree metrics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1062–1072, Baltimore, Maryland. Association for Computational Linguistics. (pages 8, 17, 22).

Hao Peng, Roy Schwartz, and Noah A. Smith. 2019. PaLM: A hybrid parser and language model. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural*

Language Processing (EMNLP-IJCNLP), pages 3644–3651, Hong Kong, China. Association for Computational Linguistics. (page 9).

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia. Association for Computational Linguistics. (page 10).

Elias Ponvert, Jason Baldridge, and Katrin Erk. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086, Portland, Oregon, USA. Association for Computational Linguistics. (page 8).

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training (2018). (page 32).

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9. (pages 10, 32).

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. (page 50).

Devendra Sachan, Yuhao Zhang, Peng Qi, and William L. Hamilton. 2021. Do syntax trees help pre-trained transformers extract information? In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2647–2661, Online. Association for Computational Linguistics. (page 55).

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*, pages 146–182, Seattle, Washington, USA. Association for Computational Linguistics. (page 41).

Yoav Seginer. 2007. Fast unsupervised incremental parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 384–391, Prague, Czech Republic. Association for Computational Linguistics. (pages 8, 17).

Dan Shen, Geert-Jan M. Kruijff, and Dietrich Klakow. 2005. Exploring syntactic relation patterns for question answering. In *Second International Joint Conference on Natural Language Processing: Full Papers*. (page 2).

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron Courville, and Yoshua Bengio. 2018a. Straight to the tree: Constituency parsing with neural syntactic distance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1180, Melbourne, Australia. Association for Computational Linguistics. (page 9).

Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. 2018b. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations*. (pages 9, 11, 15, 16, 17, 42, 43, 45, 46).

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations*. (pages 9, 15, 17, 42, 43, 45).

Yikang Shen, Yi Tay, Che Zheng, Dara Bahri, Donald Metzler, and Aaron Courville. 2021. Structformer: Joint unsupervised induction of dependency and constituency structure from masked language modeling. (page 10).

Haoyue Shi, Karen Livescu, and Kevin Gimpel. 2020. On the role of supervision in unsupervised constituency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7611–7621, Online. Association for Computational Linguistics. (pages 15, 16, 20, 55).

Haoyue Shi, Jiayuan Mao, Kevin Gimpel, and Karen Livescu. 2019. Visually grounded neural syntax acquisition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1842–1861, Florence, Italy. Association for Computational Linguistics. (pages 12, 15, 55).

Tianze Shi, Ozan İrsoy, Igor Malioutov, and Lillian Lee. 2021. Learning syntax from naturally-occurring bracketings. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2941–2949, Online. Association for Computational Linguistics. (pages 10, 16, 44, 55).

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 354–362, Ann Arbor, Michigan. Association for Computational Linguistics. (page 17).

Valentin I. Spitkovsky, Daniel Jurafsky, and Hiyan Alshawi. 2010. Profiting from mark-up: Hyper-text annotations for guided parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1278–1287, Uppsala, Sweden. Association for Computational Linguistics. (page 10).

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *10th Conference of the European Chapter of the Association for Computational Linguistics*, Budapest, Hungary. Association for Computational Linguistics. (page 20).

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics. (page 11).

Marten van Schijndel, Aaron Mueller, and Tal Linzen. 2019. Quantity doesn't buy quality syntax with neural language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5831–5837, Hong Kong, China. Association for Computational Linguistics. (page 34).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc. (pages 10, 30).

Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019. Tree transformer: Integrating tree structures into self-attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1061–1070, Hong Kong, China. Association for Computational Linguistics. (pages 10, 16, 42, 43).

Lydia White. 1990. Second language acquisition and universal grammar. *Studies in Second Language Acquisition*, 12(2):121–133. (page 4).

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics. (page 16).

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics. (page 37).

Dekai Wu and Hongsing Wong. 1998. Machine translation with a stochastic grammatical channel. In *COLING 1998 Volume 2: The 17th International Conference on Computational Linguistics*. (page 1).

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proceedings of the 58th Annual*

Meeting of the Association for Computational Linguistics, pages 4166–4176, Online. Association for Computational Linguistics. (page 33).

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Nat. Lang. Eng.*, 11(2):207–238. (page 41).

Songlin Yang, Yanpeng Zhao, and Kewei Tu. 2021. PCFGs can do better: Inducing probabilistic context-free grammars with many symbols. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1487–1498, Online. Association for Computational Linguistics. (pages 10, 52).

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc. (page 30).

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics. (page 19).

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. (page 11).

Daniel H Younger. 1967. Recognition and parsing of context-free languages in time n3. *Information and control*, 10(2):189–208. (page 27).

Songyang Zhang, Linfeng Song, Lifeng Jin, Kun Xu, Dong Yu, and Jiebo Luo. 2021. Video-aided unsupervised grammar induction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1513–1524, Online. Association for Computational Linguistics. (page 12).

Yanpeng Zhao and Ivan Titov. 2020. Visually grounded compound PCFGs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4369–4379, Online. Association for Computational Linguistics. (pages 12, 55).

Hao Zhu, Yonatan Bisk, and Graham Neubig. 2020. The return of lexical dependencies: Neural lexicalized PCFGs. *Transactions of the Association for Computational Linguistics*, 8:647–661. (pages 9, 10, 17, 52).