ARE PRE-TRAINED LANGUAGE MODELS AWARE OF PHRASES? SIMPLE BUT STRONG BASELINES FOR GRAMMAR INDUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

With the recent success and popularity of pre-trained language models (LMs) in natural language processing, there has been a rise in efforts to understand their inner workings. In line with such interest, we propose a novel method that assists us in investigating the extent to which pre-trained LMs capture the syntactic notion of constituency. Our method provides an effective way of extracting constituency trees from the pre-trained LMs without training. In addition, we report intriguing findings in the induced trees, including the fact that pre-trained LMs outperform other approaches in correctly demarcating adverb phrases in sentences.

1 INTRODUCTION

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

In this paper, inspired by the fact that the training objectives of both the approaches for grammar induction and for training LMs are identical, namely, (masked) language modeling, we investigate whether pre-trained LMs can also be utilized for grammar induction/unsupervised parsing, especially *without* training. Specifically, we focus on extracting constituency trees from pre-trained LMs without fine-tuning or introducing another task-specific module, at least one of which is usually required in other cases where representations from pre-trained LMs are employed. This restriction provides us with some advantages: (1) it enables us to derive strong baselines for grammar induction with reduced time and space complexity, offering a chance to reexamine the current status of existing grammar induction methods, (2) it facilitates an analysis on how much and what kind of syntactic information each pre-trained LM contains in its intermediate representations and attention distributions in terms of phrase-structure grammar, and (3) it allows us to easily inject biases into our framework, for instance, to encourage the right-skewness of the induced trees, resulting in performance gains in English unsupervised parsing.

First, we briefly mention related work (§2). Then, we introduce the intuition behind our proposal in detail (§3), which is motivated by our observation that we can cluster words in a sentence according to the similarity of their attention distributions over words in the sentence. Based on this intuition, we define a straightforward yet effective method (§4) of drawing constituency trees directly from pre-trained LMs with no fine-tuning or addition of task-specific parts, instead resorting to the concept of *Syntactic Distance* (Shen et al., 2018a;b). Then, we conduct experiments (§5) on the induced constituency trees, discovering some intriguing phenomena. Moreover, we analyze the pre-trained LMs and constituency trees from various points of view, including looking into which layer(s) of the LMs is considered to be sensitive to phrase information (§6).

To summarize, our contributions in this work are as follows:

- By investigating the attention distributions from Transformer-based pre-trained LMs, we show that there is evidence to suggest that several attention heads of the LMs exhibit syntactic structure akin to constituency grammar.
- Inspired by the above observation, we propose a method that facilitates the derivation of constituency trees from pre-trained LMs without training. We also demonstrate that the induced trees can serve as a strong baseline for English grammar induction.
- We inspect, in view of our framework, what type of syntactic knowledge the pre-trained LMs capture, discovering interesting facts, e.g. that pre-trained LMs are more aware of adverb phrases than other approaches.

2 Related Work

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

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

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

Concerning pre-trained language models (Peters et al. (2018); Devlin et al. (2019); Radford et al. (2019); Yang et al. (2019); Liu et al. (2019b), *inter alia*)—particularly those employing a Transformer architecture (Vaswani et al., 2017)—these have proven to be helpful for diverse NLP downstream tasks. In spite of this, there is no vivid picture for explaining what particular factors contribute to performance gains, even though some recent work has attempted to shed light on this question. In detail, one group of studies (Raganato & Tiedemann (2018); Clark et al. (2019); Hao et al. (2019); Voita et al. (2019), *inter alia*) has focused on dissecting the intermediate representations and attention distributions of the pre-trained LMs, while the another group of publications (Mareček & Rosa (2018); Goldberg (2019); Hewitt & Manning (2019); Liu et al. (2019a); Rosa & Mareček (2019), to name a few) delve into the question of the existence of syntactic knowledge in Transformer-based models. Particularly, Mareček & Rosa (2019) proposed an algorithm for extracting constituency trees from Transformers trained for machine translation, which is similar to our approach.

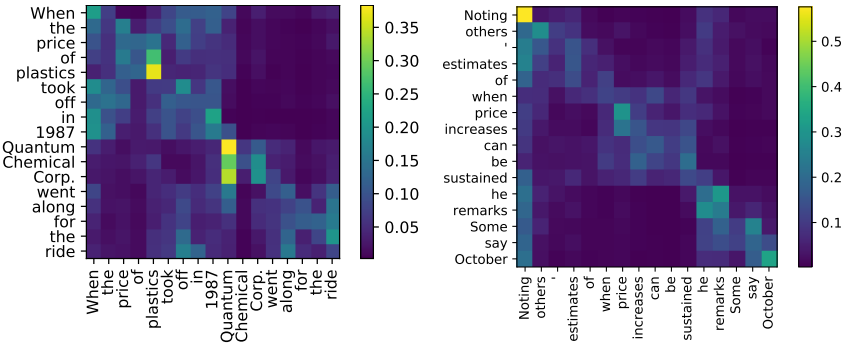

Figure 1: Self-attention heatmaps from two different pre-trained LMs. (Left) A heatmap for the average of attention distributions from the 7th layer of the XLNet-base (Yang et al., 2019) model given the sample sentence. (Right) A heatmap for the average of attention distributions from the 9th layer of the BERT-base (Devlin et al., 2019) model given another sample sentence. We can easily spot the chunks of words on the two heatmaps that are correlated with the constituents of the input sentences, e.g. (Left) 'the price of plastics', 'took off in 1987', 'Quantum Chemical Corp.', (Right) 'when price increases can be sustained', and 'he remarks'.

3 Motivation

As pioneers in the literature have pointed out, the multi-head self-attention mechanism is a key component in Transformer-based language models, and it seems this mechanism empowers the models to capture certain semantic and syntactic information existing in natural language. Among a diverse set of knowledge they may capture, in this work we concentrate on phrase-structure grammar by seeking to extract constituency trees directly from their attention information and intermediate weights.

In preliminary experiments, where we visualize and investigate the intermediate representations and attention distributions of several pre-trained LMs given input, we have found some evidence which suggests that the pre-trained LMs exhibit syntactic structure akin to constituency grammar to some extent. Specifically, we have noticed some patterns which are often displayed in self-attention heatmaps as explicit horizontal lines, or groups of rectangles of various sizes. As an attention distribution of a word in an input sentence corresponds to a row in a heatmap matrix, we can say that the appearance of these patterns indicates the existence of groups of words where the attention distributions of the words in the same group are relatively similar. Interestingly, we have also discovered the fact that the groups of words we observed are fairly correlated with the constituents of the input sentence, as shown in Figure 1 (above) and Figure 3 (in the Appendix A.1).

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

4 Proposed Method

4.1 Syntactic Distance and Tree Construction

We leverage the concept of *Syntactic Distance* proposed by Shen et al. (2018a;b) to draw constituency trees from raw sentences in an intuitive way. Formally, given a sequence of words in a

sentence, $w_1, w_2, \ldots, w_n$, we compute $\mathbf{d} = [d_1, d_2, \ldots, d_{n-1}]$ where d_i corresponds to the syntactic distance between w_i and w_{i+1}. Each d_i is defined as follows:

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

where $f(\cdot, \cdot)$ and $g(\cdot)$ are a distance measure function and representation extractor function, respectively. The function g converts each word into the corresponding vector representation, while f computes the syntactic distance between the two words given their representations. Once $\mathbf{d}$ is derived, it can be easily converted into the target constituency tree by a simple algorithm following Shen et al. (2018a). For a specification of the algorithm, we refer the reader to Appendix A.2.

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

4.2 Pre-trained Language Models

We consider four types of recently proposed language models. These are: BERT (Devlin et al., 2019), GPT-2 (Radford et al., 2019), RoBERTa (Liu et al., 2019b), and XLNet (Yang et al., 2019). They all have in common that they are based on the Transformer architecture and have been proven to be effective in natural language understanding (Wang et al., 2019) or generation. We handle two variants for each LM, varying in the number of layers, attention heads, and hidden dimensions, resulting in eight different cases in total. In particular, each LM has two variants. (1) base: consists of l=12 layers, a=12 attention heads, and d=768 hidden dimensions, while (2) large: has l=24 layers, a=16 attention heads, and d=1024 hidden dimensions[1]. Note that we deal with a wide range of pre-trained LMs, unlike previous work which has mostly analyzed a specific architecture, particularly BERT. For details about each LM, we refer readers to the respective original papers.

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

One remaining issue is that all the pre-trained LMs we use regard each input sentence as a sequence of *subword* tokens, while our formulation assumes words cannot be further divided into smaller tokens. To resolve this difference, we tested certain heuristics that guide how subword tokens for a complete word should be exploited to represent the word, and we have empirically found that the best result comes when each word is represented by an average of the representations of its subwords[2]. Therefore, we adopt the above heuristic in this work for cases where a word is tokenized into more than two parts.

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

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

4.3 DISTANCE MEASURE FUNCTIONS

For the distance measure function f, we prepare three options (F^v) for G^v and two options (F^d) for G^d. Formally, $f \in F^v \cup F^d$, where $F^v = \{\text{COS}, \text{L1}, \text{L2}\}$, $F^d = \{\text{JSD}, \text{HEL}\}$. COS, L1, L2, JSD, and HEL correspond to Cosine, L1, and L2, Jensen-Shannon, and Hellinger distance respectively. Note that the functions in F^v are only compatible with the elements of G^v, and the same holds for F^d and G^d. The exact definition of each function is listed in the Appendix A.3.

4.4 INJECTING BIAS INTO SYNTACTIC DISTANCES

One of the main advantages we obtain by leveraging syntactic distances to derive parse trees is that we can easily inject inductive bias into our framework by simply modifying the values of the syntactic distances. Hence, we investigate whether the extracted trees from our method can be further refined with the aid of additional biases. To this end, we introduce a well-known bias for English constituency trees—the *right-skewness* bias—in a simple linear form[3]. Namely, our intention is to influence the induced trees such that they are moderately right-skewed following the nature of gold-standard parse trees in English. That we directly adjust the syntactic distance values to inject bias is novel, although certain previous studies (Shen et al., 2018b; 2019) have previously exploited such biases in a different manner. Formally, we compute $\hat{d}_i$ by appending the following linear bias term to every d_i:

$$\hat{d}_i = d_i + \lambda \cdot \text{AVG}(\mathbf{d}) \times (1 - 1/(m-1) \times (i-1)), \tag{2}$$

where $\text{AVG}(\cdot)$ outputs an average of all elements in a vector, λ is a hyperparameter, and i ranges from 1 to $m = n - 1$. We write $\hat{\mathbf{d}} = [\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_m]$ in place of $\mathbf{d}$ to signify biased syntactic distances. Note that the main purpose of explicitly injecting such a bias is examining what changes are made to the resulting tree structures rather than boosting quantitative performances *per se*, though it is of note that it serves this purpose as well.

5 EXPERIMENTS

5.1 GENERAL SETTINGS

In this section, we conduct unsupervised constituency parsing on two datasets. The first dataset is WSJ Penn Treebank (PTB, Marcus et al. (1993)), in which human-annotated gold-standard trees are available. As LMs are not fine-tuned with training sets in our framework, we only use the test set of the PTB (following a trivial split of the dataset). The second one is the MNLI (Williams et al., 2018b), which is originally designed to test natural language inference but often utilized as a means of evaluating parsers. It contains constituency trees produced by an external parser (Klein & Manning, 2003). We leverage the union of two different versions of the MNLI development set as test data following convention (Htut et al., 2018; Drozdov et al., 2019). To preprocess the datasets, we follow the setting of Kim et al. (2019a) with the minor exceptions that words are not lower-cased and number characters are preserved instead of being substituted by a special character.

For implementation, to compare pre-trained LMs in an unified manner, we resort to an integrated PyTorch codebase[4] that supports all the models we consider. Given an LM and input sentence, we compute a set of $\mathbf{d}$ from all possible combinations of f and g, followed by the resulting constituency trees converted from each $\mathbf{d}$ by the tree construction algorithm in Section 4.1. Among the candidate trees, we select one derived from the best choice of f and g in terms of sentence-level F1 (S-F1) w.r.t. gold-standard trees as a representative for the LM. For each LM, in addition to its S-F1 score, we report its label recall scores for six main categories: SBAR, NP, VP, PP, ADJP, and ADVP. We also present the results of utilizing $\hat{\mathbf{d}}$ instead of $\mathbf{d}$, empirically setting the bias hyperparameter λ as 1.5. When we do not employ the right-skewness bias from Section 4.4, we instead apply a heuristic used in previous work (Shen et al., 2018b; 2019) to the tree construction algorithm for fair comparison. We take four naïve baselines into account, random (averaged over 5 trials), balanced, left-branching, and right-branching binary trees. Furthermore, we compare our parse trees against

[3]We emphasize that it will be necessary to carefully design biases for other languages as they have their own properties.

[4]https://github.com/huggingface/pytorch-transformers

Table 1: Results on the PTB test set. Bold numbers correspond to the top 3 results for each column. L: layer number, A: attention head number (AVG: the average of all attentions). †: Results reported by Kim et al. (2019a).

Model	f	L	A	S-F1	SBAR	NP	VP	PP	ADJP	ADVP
Baselines										
Random (5 trials)	-	-	-	19.3	11%	23%	15%	21%	22%	26%
Balanced	-	-	-	18.3	8%	25%	10%	20%	19%	24%
Left Branching	-	-	-	8.7	5%	11%	0%	5%	2%	8%
Right Branching	-	-	-	39.4	**68%**	24%	**71%**	42%	27%	38%
Ours (w/o bias)										
BERT-base	HEL	9	2	36.8	37%	41%	38%	49%	31%	50%
BERT-large	JSD	9	10	40.2	44%	44%	39%	55%	31%	53%
GPT2	JSD	9	1	39.6	31%	52%	33%	44%	29%	38%
GPT2-medium	JSD	10	13	41.9	38%	55%	33%	49%	39%	44%
RoBERTa-base	JSD	9	4	40.9	46%	46%	44%	51%	39%	58%
RoBERTa-large	JSD	15	8	38.0	32%	50%	32%	42%	34%	51%
XLNet-base	HEL	7	AVG	42.9	41%	52%	37%	51%	44%	**65%**
XLNet-large	L2	11	-	41.2	38%	50%	37%	48%	**45%**	61%
Ours (w/ bias λ=1.5)										
BERT-base	HEL	9	AVG	43.0	47%	43%	57%	47%	39%	60%
BERT-large	HEL	17	AVG	46.0	**57%**	48%	57%	54%	43%	60%
GPT2	JSD	9	1	42.3	44%	51%	44%	48%	29%	41%
GPT2-medium	HEL	8	2	44.1	56%	43%	**59%**	53%	36%	48%
RoBERTa-base	JSD	7	AVG	44.7	51%	46%	58%	52%	41%	60%
RoBERTa-large	JSD	12	AVG	43.0	47%	46%	53%	45%	43%	51%
XLNet-base	HEL	7	AVG	**49.4**	**63%**	52%	**60%**	**59%**	**47%**	**63%**
XLNet-large	JSD	11	AVG	47.3	56%	50%	55%	53%	**48%**	**62%**
Other models										
PRPN(tuned)†	-	-	-	47.3	50%	59%	46%	57%	44%	32%
ON(tuned)†	-	-	-	48.1	51%	**64%**	41%	54%	38%	31%
Neural PCFG†	-	-	-	**50.8**	52%	**71%**	33%	**58%**	32%	45%
Compound PCFG†	-	-	-	**55.2**	56%	**74%**	41%	**68%**	40%	52%

ones from existing grammar induction models. All scripts used in our experiments will be publicly available for reproduction and further analysis[5].

5.2 Experimental Results on PTB

In Table 1, we report the results of the various models on the PTB test set. First of all, our method combined with pre-trained LMs shows competitive results in terms of S-F1 even without the right-skewness bias—five among eight instances outperform the strong right-branching baseline. This result implies that the extracted trees from our method can be regarded as a baseline for English grammar induction. When the right-skewness bias is applied to the models, their F1 scores increase by about six percentage points. This improvement indicates that the pre-trained LMs do not properly capture the largely right-branching nature of English syntax. We conjecture this trend would also hold for the existing grammar induction models, even though some of them already utilized the bias, based on the fact that the right-branching baseline beats all other models in recognizing subordinate clauses (SBAR) and verb phrases (VP). Existing models show exceptionally high recall scores on noun phrases (NP). In contrast, the pre-trained LMs record the best recall scores on adjective and adverb phrases (ADJP and ADVP), suggesting that the LMs and existing models capture disparate aspects of English syntax to differing degrees.

In comparison with other LM models, both of the XLNet-based models demonstrate their effectiveness in unsupervised parsing. Particularly, our method combined with the XLNet-base model serves as a robust baseline with the aid of the right-skewness bias, achieving a top 3 result in terms of almost every evaluation metric except the recall score on NP. One plausible explanation for this outcome is that the training objective of XLNet, which employs both autoencoding (AE) and autoregressive (AR) features, might encourage the model to be better aware of phrase structure than other LMs. However, it is hard to conclude what factors contribute to its high performance without further analysis.

On the other hand, there is an obvious trend that the functions in F^d—the distance measure functions for attention distributions—lead most of the LM instances to the best parsing results, indicating

[5] `https://anonymized.for.review`

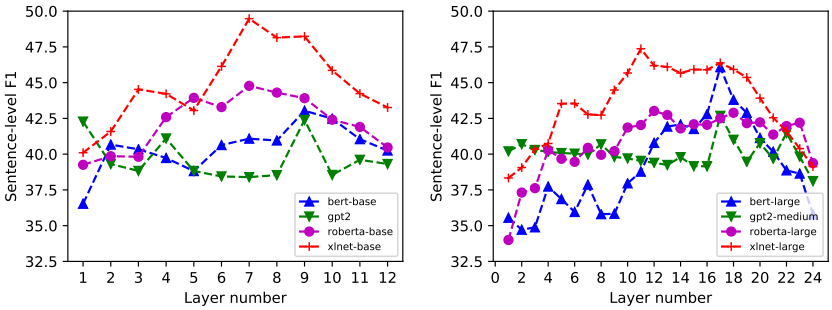

Figure 2: The best layer-wise S-F1 scores of each LM instance on the PTB test set. (Left) The performances of the X-'base' models. (Right) The performances of the X-'large' models.

that deriving parse trees from attention information can be more compact and effective than extracting them from the LMs' intermediate representations, which should contain linguistic knowledge beyond phrase structure. In addition, the results in Table 1 show that large parameterizations of the LMs do not guarantee improved performance in grammar induction, at least following our method. A possible explanation for this phenomenon is that enlarging the number of parameters of an LM may make it difficult for simple methods, including ours, to extract syntactic knowledge from the LM. Meanwhile, as we expected in Section 4.2 and as seen in the 'A' (attention head number) column of Table 1, the average of attention distributions in the same layer often provides better results than individual attention distributions—especially when combined with the skewness bias.

5.3 Experimental Results on MNLI

We present the results of various models on the MNLI dataset in Table 3 of Appendix A.5. We observe trends in the results which mainly coincide with those of the PTB dataset. Particularly, (1) right-branching trees are strong baselines for the task, especially showing their strengths in capturing SBAR and VP clauses/phrases, (2) our method resorting to the LM instances is also comparable to the right-branching trees, demonstrating better performance consistently in recognizing adverb phrases (ADVP), and (3) attention distributions seem more suitable for distilling the phrase structures of sentences than intermediate representations.

However, there are some exceptions worth mentioning. First, the right-branching baseline seems to be even stronger in the case of MNLI, recording a score of over 50 in sentence-level F1. We conjecture that this result comes principally from two reasons: (1) the average length of sentences in MNLI is much shorter than in PTB, giving a disproportionate advantage to naïve baselines, and (2) our data preprocessing, which follows Kim et al. (2019a), removes all punctuation marks, unlike previous work (Htut et al., 2018; Drozdov et al., 2019), leading to an unexpected advantage for the right-branching scheme. Moreover, we need to note that the gold-standard parse trees in MNLI are not human-annotated, rather automatically generated. Second, GPT-2 instances exhibit their improved performances on the MNLI compared against those measured in the PTB. We also speculate it is caused by the same factor as mentioned above—the exceptionally right-skewed nature of the gold-standard trees in MNLI. In detail, as the GPT-2 is based on uni-directional language modeling instead of bi-directional, its self-attention matrices usually have a triangular form, meaning that a word in a sentence mainly attends over only previous words, and thus the resulting trees from the GPT-2 are more easily skewed, resulting in more right-branching trees.

6 Further Analysis

6.1 Performance comparison by layer

To take a closer look at how different the layers of the pre-trained LMs are in terms of parsing performance, we retrieve the best sentence-level F1 scores from the lth layer of an LM from all combinations of f and g_l, with regard to the PTB and MNLI respectively. Then we plot the scores

Table 2: Results of training a pseudo-optimum f_{ideal} for the PTB dataset with the XLNet-base model.

Model	f	L	A	S-F1	SBAR	NP	VP	PP	ADJP	ADVP
XLNet-base (λ=0)	HEL	7	AVG	42.9	41%	52%	37%	51%	44%	65%
XLNet-base (λ=1.5)	HEL	7	AVG	49.4	**63%**	52%	60%	59%	47%	63%
XLNet-base + random init.	f_{ideal}	-	-	38.6	29%	53%	31%	42%	34%	47%
XLNet-base (worst)	f_{ideal}	1	-	56.1	45%	70%	60%	67%	44%	59%
XLNet-base (best)	f_{ideal}	7	-	**62.5**	60%	**74%**	**72%**	**78%**	**48%**	**69%**

as graphs in Figure 2 for the PTB and Figure 4 in Appendix A.4 for the MNLI. Note that each score is from the models to which the right-skewness bias is applied. From the graphs, we observe several interesting patterns. First, XLNet-based models outperform other competitors across most of the layers. Second, the best outcomes are largely shown in the middle layers of the LMs, akin to the observation from Shen et al. (2019). Moreover, we discover from raw statistics that regardless of the choice of f and g_l, the parsing performances reported as S-F1 are moderately correlated with the layer number l. In other words, it seems that there are some particular layers in the LMs which are more sensitive to syntactic information.

6.2 Estimating the upper limit of distance measure functions

Although we introduced effective candidates for f, we explore the potential of extracting more sophisticated trees from pre-trained LMs, supposing we are equipped with a pseudo-optimum f, call it f_{ideal}. To obtain f_{ideal}, we train a simple linear layer on each layer of the pre-trained LMs with supervision from the gold-standard trees of the PTB training set, while g remains unchanged—the pre-trained LMs are frozen during training. We choose the XLNet-base model as a representative for the pre-trained LMs. For more details about experimental settings, refer to Appendix A.6.

In Table 2, we present three new results using f_{ideal}. As a baseline, we report the performance of f_{ideal} with a randomly initialized XLNet-base. Then, we list the worst and best result of f_{ideal} according to g, when it is combined with the pre-trained LM. We here mention some findings from the experiment. First, comparing the results with the pre-trained LM against one with the random LM, we reconfirm that pre-training an LM apparently enables the model to capture some aspects of grammar. Second, we find that there is a fluctuation in the performances of f_{ideal}, which is surprisingly similar to the one observed in Section 6.1—the best outcome comes from the 7th layer of the LM while the worst from the first layer. Third, we identify that the LM has a potential to show improved performance on grammar induction by adopting a more sophisticated f. However, note that our method equipped with even a simple f is remarkably good at catching ADJP and ADVP.

6.3 Constituency Tree Examples

We visualize several gold-standard trees from the PTB and the corresponding tree predictions for comparison. For more details, we refer readers to Appendix A.7.

7 Conclusion and Future Work

In this paper, we propose a simple but effective method of inducing constituency trees from pre-trained language models with no training. Furthermore, we report a set of intuitive findings observed from the extracted trees, demonstrating that the pre-trained LMs exhibit some properties similar to constituency grammar. In addition, we show that our method can serve as a strong baseline for English grammar induction when combined with (or even without) appropriate linguistic biases. On the other hand, there are still remaining issues that can be good starting points for future work. First, although we analyzed our method based on two popular datasets, we focused only on English grammar induction. As each language has its own properties (and correspondingly would need individualized biases), it is desirable to expand this work to other languages. Second, it would also be desirable to investigate whether further improvements can be achieved by directly grafting the pre-trained LMs onto existing grammar induction models. Lastly, by verifying the usefulness of the knowledge from the pre-trained LMs and linguistic biases for grammar induction, we want to point out that there is still much room for improvement in the existing grammar induction models.

REFERENCES

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pp. 89–96. ACM, 2005.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. Learning to compose task-specific tree structures. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

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

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 276–286, Florence, Italy, August 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019.

Andrew Drozdov, Patrick Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 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)*, pp. 1129–1141, Minneapolis, Minnesota, June 2019.

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

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

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

Serhii Havrylov, Germán Kruszewski, and Armand Joulin. Cooperative learning of disjoint syntax and semantics. 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)*, pp. 1118–1128, Minneapolis, Minnesota, June 2019.

John Hewitt and Christopher D. Manning. 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)*, pp. 4129–4138, Minneapolis, Minnesota, June 2019.

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

Phu Mon Htut, Kyunghyun Cho, and Samuel Bowman. Grammar induction with neural language models: An unusual replication. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 371–373, Brussels, Belgium, November 2018.

Yoon Kim, Chris Dyer, and Alexander Rush. Compound probabilistic context-free grammars for grammar induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2369–2385, Florence, Italy, July 2019a.

Yoon Kim, Alexander Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 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)*, pp. 1105–1117, Minneapolis, Minnesota, June 2019b.

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

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

Dan Klein and Christopher Manning. 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)*, pp. 478–485, Barcelona, Spain, July 2004.

Dan Klein and Christopher D. Manning. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 128–135, Philadelphia, Pennsylvania, USA, July 2002.

Dan Klein and Christopher D Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pp. 423–430. Association for Computational Linguistics, 2003.

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

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

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. Linguistic knowledge and transferability of contextual 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)*, pp. 1073–1094, Minneapolis, Minnesota, June 2019a.

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

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

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

David Mareček and Rudolf Rosa. Extracting syntactic trees from transformer encoder self-attentions. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 347–349, Brussels, Belgium, November 2018.

David Mareček and Rudolf Rosa. From balustrades to pierre vinken: Looking for syntax in transformer self-attentions. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 263–275, Florence, Italy, August 2019.

Nikita Nangia and Samuel Bowman. Listops: A diagnostic dataset for latent tree learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pp. 92–99, 2018.

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

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

Alessandro Raganato and Jörg Tiedemann. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 287–297, Brussels, Belgium, November 2018.

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

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron Courville, and Yoshua Bengio. 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)*, pp. 1171–1180, Melbourne, Australia, July 2018a.

Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations*, 2018b.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations*, 2019.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5797–5808, July 2019.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics*, 6:253–267, 2018a.

Adina Williams, Nikita Nangia, and Samuel Bowman. 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)*, pp. 1112–1122, New Orleans, Louisiana, June 2018b.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. Learning to compose words into sentences with reinforcement learning. In *International Conference on Learning Representations*, 2017.

A APPENDIX

A.1 ATTENTION HEATMAP EXAMPLES

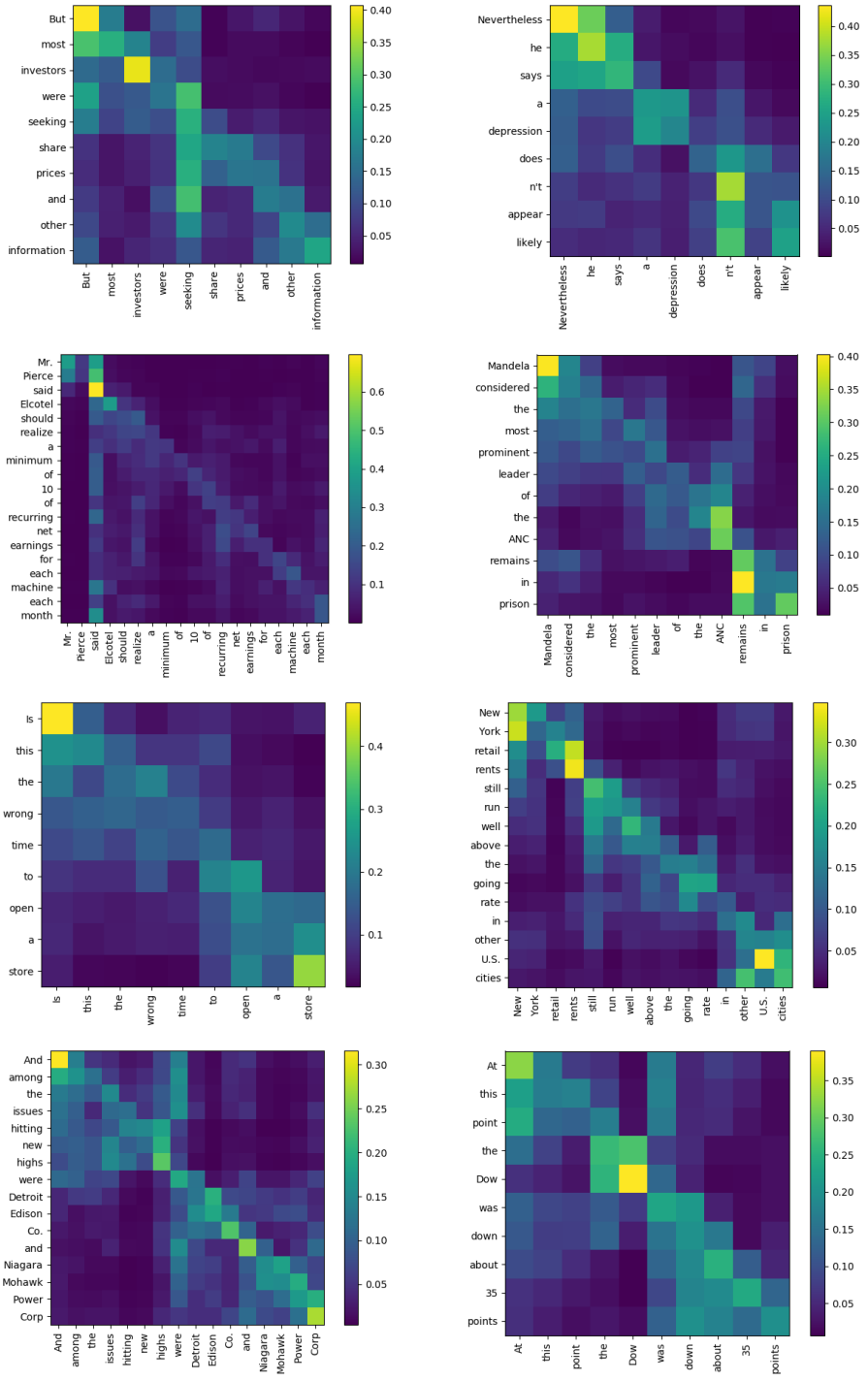

Figure 3: Self-attention heatmaps for the average of all attention distributions from the 7th layer of the XLNet-base model, given a set of input sentences.

A.2 TREE CONSTRUCTION ALGORITHM WITH SYNTACTIC DISTANCES

Algorithm 1 Syntactic Distances to Binary Constituency Tree (originally from Shen et al. (2018a))

$S = [w_1, w_2, \ldots, w_n]$: a sequence of words in a sentence of length n.
$\mathbf{d} = [d_1, d_2, \ldots, d_{n-1}]$: a vector whose elements are the distances between every two adjacent words.
function TREE(S, $\mathbf{d}$)
    if $\mathbf{d} = []$ **then**
        node $\leftarrow$ Leaf($S[0]$)
    else
        $i \leftarrow \arg\max_i(\mathbf{d})$
        $\text{child}_l \leftarrow \text{TREE}(S_{\leq i}, \mathbf{d}_{<i})$
        $\text{child}_r \leftarrow \text{TREE}(S_{>i}, \mathbf{d}_{>i})$
        node $\leftarrow$ Node(child_l, child_r)
    end if
    return node
end function

A.3 DISTANCE MEASURE FUNCTIONS

Table 3: The definitions of distance measure functions for computing syntactic distances between two adjacent words in a sentence. Note that $\mathbf{r} = g^v(w_i)$, $\mathbf{s} = g^v(w_{i+1})$, $P = g^d(w_i)$, and $Q = g^d(w_{i+1})$, respectively. d: hidden embedding size, n: the number of words (w) in a sentence (S).

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

A.4 PERFORMANCE COMPARISON BY LAYER ON MNLI

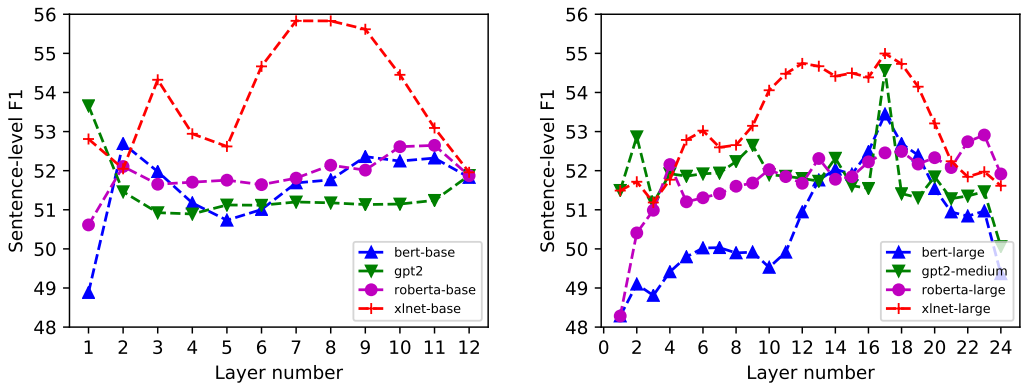

Figure 4: The best layer-wise S-F1 scores of each LM instance on the MNLI validation set. (Left) The performances of the X-'base' models. (Right) The performances of the X-'large' models.

A.5 EXPERIMENTAL RESULTS ON MNLI

Table 4: Results on the MNLI dev set. Bold numbers correspond to the top 3 results for each column. L: layer number, A: attention head number (AVG: the average of all attentions). †: Results reported by Htut et al. (2018) and Drozdov et al. (2019). ∗: These results are not strictly comparable to ours, due to the difference in data preprocessing.

Model	f	**L**	**A**	**S-F1**	**SBAR**	**NP**	**VP**	**PP**	**ADJP**	**ADVP**
Baselines										
Random (5 trials)	-	-	-	22.9	14%	25%	18%	24%	23%	27%
Balanced	-	-	-	23.4	12%	29%	16%	27%	23%	33%
Left Branching	-	-	-	8.4	6%	13%	1%	4%	1%	8%
Right Branching	-	-	-	51.9	**65%**	28%	**75%**	47%	45%	30%
Ours (w/o bias)										
BERT-base	JSD	9	9	43.9	42%	43%	44%	47%	36%	46%
BERT-large	JSD	17	10	46.3	44%	43%	46%	53%	37%	40%
GPT2	HEL	1	10	48.6	44%	**53%**	46%	45%	36%	36%
GPT2-medium	JSD	3	12	50.5	58%	31%	67%	43%	42%	35%
RoBERTa-base	JSD	9	4	44.4	45%	42%	48%	48%	42%	**49%**
RoBERTa-large	JSD	11	11	41.9	35%	43%	39%	50%	40%	46%
XLNet-base	HEL	7	AVG	45.3	42%	**52%**	38%	49%	38%	**55%**
XLNet-large	HEL	1	15	45.6	32%	**55%**	33%	51%	36%	42%
Ours (w/ bias λ=1.5)										
BERT-base	HEL	2	12	52.6	**64%**	33%	**72%**	49%	**46%**	30%
BERT-large	HEL	17	AVG	53.4	55%	45%	63%	**54%**	**47%**	47%
GPT2	HEL	1	10	53.6	**59%**	50%	60%	49%	38%	36%
GPT2-medium	HEL	2	1	**54.5**	55%	52%	61%	49%	38%	38%
RoBERTa-base	L1	11	-	52.6	55%	38%	**69%**	51%	45%	37%
RoBERTa-large	HEL	3	AVG	47.8	52%	30%	61%	44%	39%	34%
XLNet-base	HEL	7	AVG	**55.8**	58%	50%	63%	**59%**	45%	**51%**
XLNet-large	L2	17	-	**55.0**	58%	45%	65%	**57%**	**46%**	47%
Other models										
PRPN-UP†	-	-	-	48.6∗	-	-	-	-	-	-
PRPN-LM†	-	-	-	50.4∗	-	-	-	-	-	-
DIORA†	-	-	-	51.2∗	-	-	-	-	-	-
DIORA(+PP)†	-	-	-	59.0∗	-	-	-	-	-	-

A.6 EXPERIMENTAL DETAILS FOR TRAINING IDEAL DISTANCE MEASURE FUNCTION

In this part, we present the detailed specifications of the experiments introduced in Section 6.2. We assume f_{ideal} is only compatible with the functions in G^v, as the functions in G^d are not suitable for training as the sizes of the representations provided by G^d are variable according to the length of an input sentence. To train the pseudo-optimal function f_{ideal}, we minimize a pair-wise learning-to-rank loss following previous work (Burges et al., 2005; Shen et al., 2018a):

$$L_{\text{dist}}^{\text{rank}} = \sum_{i,j>i} [1 - \text{sign}(d_i^{\text{gold}} - d_j^{\text{gold}})(d_i^{\text{pred}} - d_j^{\text{pred}})]^+, \tag{3}$$

where d^{gold} and d^{pred} are computed from the gold tree and our predicted one, respectively. $[x]^+$ is defined as $max(0, x)$. We train the f_{ideal} with the PTB training set for 5 epochs. Each batch of the training set contains 16 sentences. We use an ADAM optimizer (Kingma & Ba, 2014) with the learning rate 5e-4. We train the variations of f_{ideal} differentiated by the choice of g in G^v and report the best result in the Table 2. Each f_{ideal} is chosen based on its performance on the PTB validation set. Considering the randomness of training, every result for f_{ideal} is averaged over 3 different trials.

A.7 CONSTITUENCY TREE EXAMPLES

We here present 6 pairs of trees, which are randomly chosen from the PTB, as an example, where each pair consists of a gold constituency tree and the corresponding predicted one from our best

model—XLNet-base with the right-skewness bias as in Section 5. The upper one corresponds to the gold tree, while the bottom one is the induced tree. The 'T' in the induced trees indicates a dummy tag.

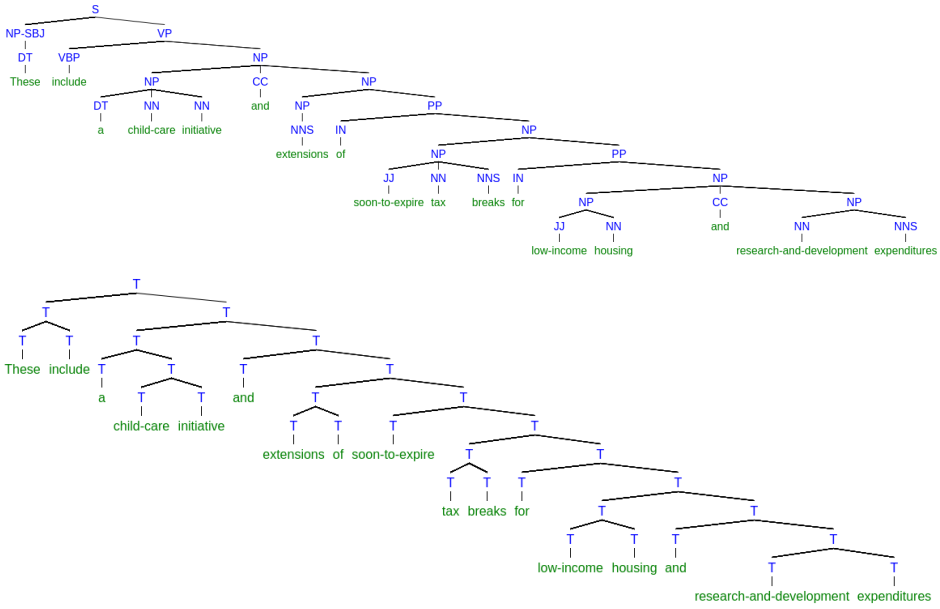

Figure 5: Gold (top) and predicted (bottom) trees for the sentence 'These include a child-care initiative and extensions of soon-to-expire tax breaks for low-income housing and research-and-development expenditures'.

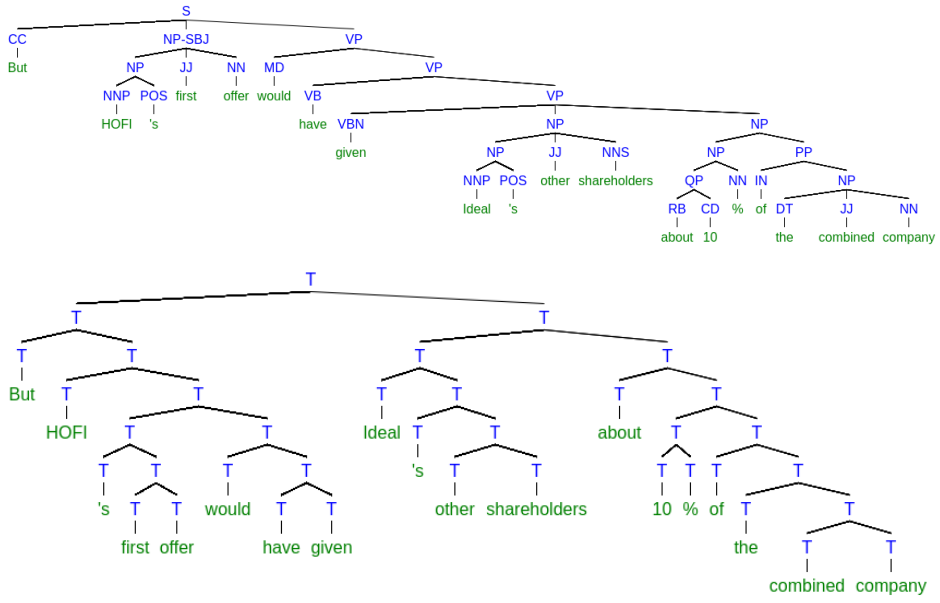

Figure 6: Gold (top) and predicted (bottom) trees for the sentence 'But HOFI 's first offer would have given Ideal 's other shareholders about 10 % of the combined company'.

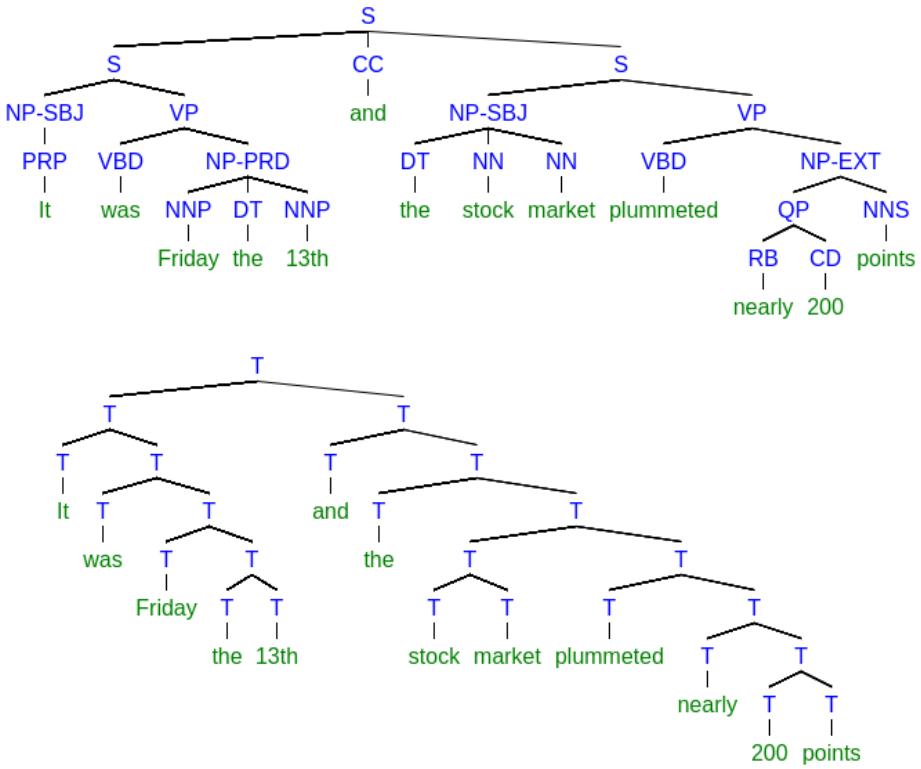

Figure 7: Gold (top) and predicted (bottom) trees for the sentence 'It was Friday the 13th and the stock market plummeted nearly 200 points'.

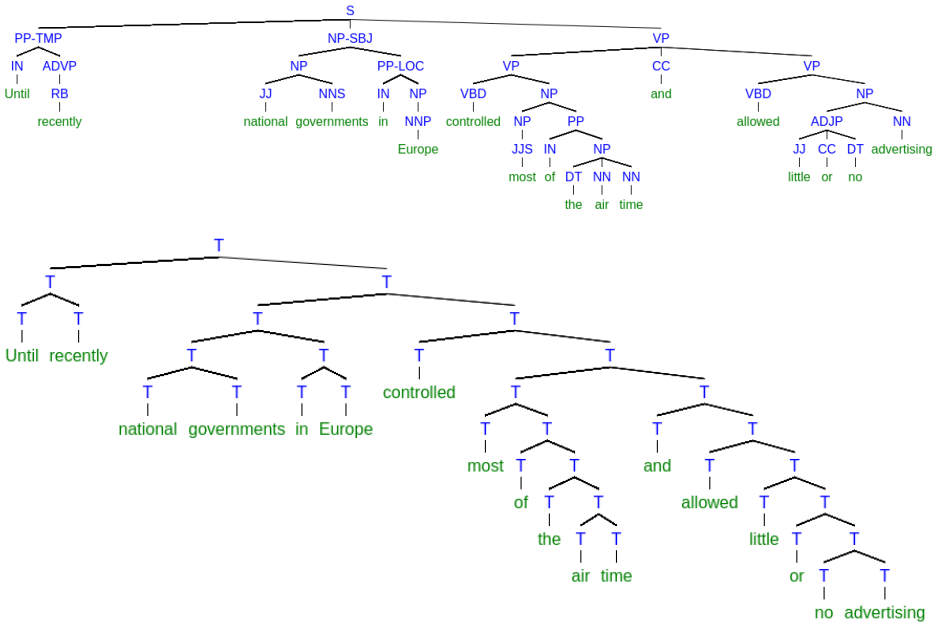

Figure 8: Gold (top) and predicted (bottom) trees for the sentence 'Until recently national governments in Europe controlled most of the air time and allowed little or no advertising'.

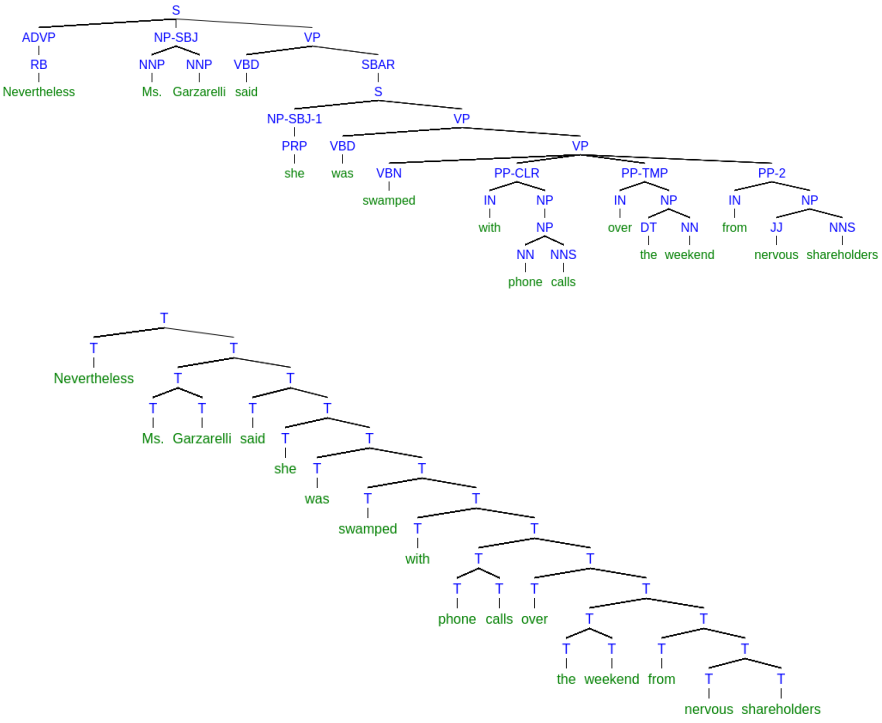

Figure 9: Gold (top) and predicted (bottom) trees for the sentence 'Nevertheless Ms. Garzarelli said she was swamped with phone calls over the weekend from nervous shareholders'.

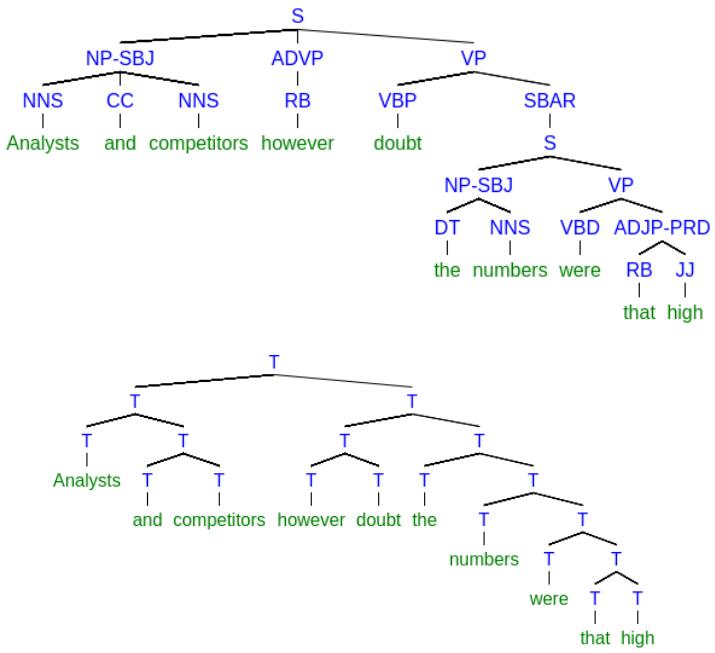

Figure 10: Gold (top) and predicted (bottom) trees for the sentence 'Analysts and competitors however doubt the numbers were that high'.