Universiteit Leiden

Master Computer Science

Tree Transformer's Disambiguation Ability of Prepositional Phrase Attachment and Garden Path Effects

Name:	Lingling Zhou
Student ID:	s3231844
Date:	08/12/2023
Specialisation:	Data Science
1st supervisor:	Gijs Wijnholds
2nd supervisor:	Suzan Verberne

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Prepositional phrases (PPs) play a crucial role in the knowledge extraction process within methods for constructing knowledge bases. Despite this importance, prepositional phrases remain a significant source of syntactic ambiguity, posing ongoing challenges in the parsing process. While there has been considerable research in this area, existing research has predominantly focused on prepositional phrases with limited analysis at the sentence level. Additionally, investigations into this issue within the context of neural networks are currently quite scarce. On the contrary, another form of structural ambiguity, the garden path effect, has attracted the attention of many scholars within the context of neural language models, like RNNG and LSTMs trained on different corpora.

On the other hand, neural language models that induce constituency tree structures can effectively capture the hierarchical nature of human language. However, many previous studies relied on supervised syntactic parsers, which require costly manual annotations, such as a BiLSTM-based supervised training model that reduces constituent parsing to a sequence labeling task. Consequently, in recent years, research has placed significant emphasis on studying grammar induction which is learning latent tree structures without the need for human-annotated data and has garnered widespread attention from researchers. As fine-tuning pre-trained Transformers has achieved state-of-the-art results on various NLP tasks, one notable approach involves integrating tree structures into the Transformer, known as the Tree Transformer.

Our primary focus is on investigating the disambiguation capability of the unsupervised Tree Transformer at the sentence level regarding prepositional phrase attachment, comparing it with the parsing capabilities of the supervised BiLSTM. We observed that the parsing ability of the Tree Transformer is not as robust as expected when compared to the BiLSTM. Additionally, experimental results indicate that the Tree Transformer tends to attach prepositional phrases to verbs, while for sentences containing “be” or “have” verbs and their variants, the model is more inclined to select nouns for attachment decisions, a phenomenon not observed in the BiLSTM. In addition to parsing ability analysis, we also explored aspects similar to those of LSTMs and RNNG, for instance, whether the Tree Transformer exhibits garden path effects and its sensitivity to subtle lexical cues leading to syntactic state changes will be examined. It was found that the magnitude of the garden path effect and sensitivity to lexical cues displayed by it across multiple datasets are not as pronounced as those observed in large-scale LSTMs. The sensitivity to the presence of an object is higher compared to verb transitivity and verb form.

Contents

1 Introduction **4**

2 Related Work **7**
2.1 Prepositional Phrase Attachment Ambiguity 7
2.2 Garden Path Effect 8
2.3 Grammar Induction 9

3 Fundamentals **12**
3.1 Evaluation Metrics 12
3.2 Tree Transformer 13
3.2.1 Constituent Prior 14
3.2.2 Constituent Attention Module 15
3.3 Constituent Parsing with the Sequence Labeling Model BiLSTM 17
3.3.1 Encoding 18
3.3.2 Prediction 20
3.3.3 Decoding 20

4 Methods **22**
4.1 Data 22
4.1.1 Penn Treebank 22
4.1.2 Prepositional Phrase Attachment Corpus 22
4.1.3 Generating Prepositional Phrase Attachment Sentences 23
4.1.4 Garden Path Effect Datasets 24
4.2 Unsupervised Parsing from Tree Transformer 25
4.3 Experimental set-up 27
4.3.1 Tree Transformer Model 27
4.3.2 BiLSTM Model 29
4.3.3 Parsing PP Attachment 29
4.3.4 Measuring Garden Path Effects 30

5 Results **32**
5.1 Performance on Test data 32
5.2 Parsing Analysis on PP Attachment Sentences 32
5.2.1 Analysis on Parsing Structure 32
5.2.2 Assessment of Parsing Results 33
5.3 Parsing Analysis on the Garden Path Effect 36

6 Discussion **42**

7 Conclusion **43**

1 Introduction

Natural Language Processing (NLP) is a crucial branch of computer science and artificial intelligence aimed to enable computers to understand, interpret, generate and process human language. NLP involves the learning, handling and analysis of textual data, allowing computers to communicate with human language. However, due to the various types of ambiguity inherent in natural language, NLP faces multiple difficulties and challenges. Ambiguity arises from the polysemy, complexity and contextual variations in natural language, with syntactic ambiguity being one of them. Specific form of syntactic ambiguity, where a prepositional phrase (PP) can attach to different positions in a sentence, leading to multiple possible structures and meanings. For instance, sentence "**I saw the man with the telescope**" can have different meanings depending on whether "with the telescope" attaches to "the man" or "saw", leading to two different syntactic structures:

- I saw the man through the telescope.
- I saw the man who had a telescope.

Due to the widespread presence of prepositional phrase attachment ambiguity in natural language, it has garnered significant attention from scholars. Among the works, structure-based methods have been explored, such as the Right Association (Kimball, 1973) and Minimal Attachment methods (Frazier, 1979), which despite their popularity due to simplicity, exhibit notable shortcomings and perform suboptimally in practical applications. Additionally, statistics-based methods have been employed. Hindle and Rooth (1993) introduced the first corpus-based co-occurrence statistical method, known as "lexical association". Then, Ratnaparkhi et al. (1994) developed a maximum entropy model to calculate the probability of attachment decisions. To address sparsity issues in these methods, Collins and Brooks (1995) introduced a back-off model and WordNet (Fellbaum, 1998) classes have been applied by researchers (Stetina and Nagao, 1997; Toutanova et al., 2004). Besides, there are rule-based methods, including an approach proposed by Brill and Resnik (1994), which involves learning a set of transformation rules from a corpus. However, the limitation of this method lies in the specificity and lack of generality of these rules, making it cumbersome for practical applications and maintenance. Hence, it can be observed that the exploration of prepositional phrase attachment ambiguity in neural language models is quite limited.

Another well-studied syntactic ambiguity is garden path effect (Bever, 1970) which means the comprehender is guided toward a locally plausible but ultimately incorrect parse, much like being led down a garden path only to discover errors along the way. It is a valuable area of study in psycholinguistics because it provides insights into how humans process and understand language. It helps researchers explore the complexities of sentence comprehension, the role of syntax and grammar, as well as our brains adapt to unexpected linguistic structures. Therefore, the garden path effect has been widely studied in the field of neural language models. Among them, Van Schijndel and Linzen (2018a,b) demonstrated the garden path effect in LSTM models by simulating human reading times.

Surprisal, in the context of language and psycholinguistics, is a concept used to quantify the unexpectedness or information content of a word or phrase within a sentence. It measures the degree of surprise or uncertainty associated with encountering a specific linguistic element based on the context and prior linguistic information. In addition, surprisal is often used to study how human process language and make predictions regarding forthcoming words in a sentence. The surprisal theory (Hale, 2001; Levy, 2008) has demonstrated that the surprisal derived from grammar-based language models can qualitatively explain specific cases of syntactic ambiguity resolution difficulties. Subsequently, the surprisal introduced by broad-coverage grammar-based language models has been shown to be correlated with reading time by Demberg and Keller (2008). For neural language models, the surprisal in RNNs has been found to be a powerful predictor of human reading times (Frank and Bod, 2011; Goodkind and Bicknell, 2018), and Van Schijndel and Linzen (2018a) demonstrated their ability to make reading time predictions comparable to grammar-based language models. In addition to validating the surprisal theory across different models, Futrell et al. (2019) tested multiple LSTMs and RNNG to determine if they exhibit the garden path effect and observed their levels of surprisal in disambiguating sentences.

As is widely known, human language exhibits rich hierarchical structures. Hence, models capable of automatically deriving tree structures from raw text can simulate this type of hierarchical structure. However, most previous research incorporating such tree structures into neural networks has predominantly focused on supervised syntactic parsers, relying on annotated parsing trees. For instance, Gómez-Rodríguez and Vilares (2018) proposed a BiLSTM-based model treating constituent parsing as a sequence labeling task.

As a result, researchers have explored various techniques, hoping that models could learn latent tree structures from unlabeled data without explicit syntactic annotations. This line of exploration is known as grammar induction (Smith and Eisner, 2005), aiming to induce tree structures without relying on extensive manually annotated training data. Among them, Yogatama et al. (2016) depicted the problem as a reinforcement learning task. In addition, there have been attempts based on recurrent neural networks, such as PRPN (Shen et al., 2018a) and On-LSTM (Shen et al., 2018b). Works based on recursive neural networks, such as URNNG (Kim et al., 2019b) and DIORA (Drozdov et al., 2019), also exist. Due to pre-trained Transformers from large-scale raw text, obtaining high-quality language representations, and achieving state-of-the-art results in various NLP tasks, Wang et al. (2019), inspired by Tree-RNNs (Tai et al., 2015), proposed the Tree Transformer.

In our work, to gain a better understanding of the capabilities of the Tree Transformer, we pose the following research questions:

- How well can the Tree Transformer model handle PP attachment ambiguity?
- How well can the Tree Transformer model predict garden path effects?

The results based on these research questions not only reveals the Tree Transformer's proficiency in parsing and language modeling tasks but also provides a deeper understanding of its intrinsic mechanisms when handling complex grammar structures and ambiguity. These aspects have also been previously unexplored. Our primary contributions are:

- We convert the PP attachment corpus into naturalistic sentences using BERT.

- We train a Tree Transformer from scratch on the Penn Treebank, and evaluate it on our novel data to quantify the Tree Transformer's performance.
- We evaluate pre-trained BiLSTM on our novel data to quantify the model's performance.
- We quantify the garden path effect size of Tree Transformer and its sensitivity to subtle lexical cues, compared with the models in another work.

In section 2, we will delve into some prior research. For instance, studies on prepositional phrase attachment will be systematically introduced, encompassing parsing accuracy. Concerning grammar induction, the focus will be on research involving unsupervised neural language models. Additionally, research related to the garden path effect and its quantification through surprisal will be expounded in this section. The subsequent section 3 will elucidate the evaluation metrics and the underlying principles of the models. The section 4 will detail the methods employed in the experiments, including the datasets used, the unsupervised parsing algorithm for Tree Transformer and the configurations of various experiments. All experimental results will be consolidated in section 5. Finally, comprehensive discussion and conclusion of the entire experiment will ensue in sections 6 and 7, respectively.

2 Related Work

This section reviews some historical progress in prepositional phrase attachment ambiguity, grammar induction and garden path effect.

2.1 Prepositional Phrase Attachment Ambiguity

Natural language processing (NLP) strives to create computer systems for the automatic processing of human language. Unlike programming languages, natural language poses significant challenges because of its ambiguity. One of the most common and widely studied forms of ambiguity is structural ambiguity, also known as syntactic ambiguity, where a sequence of words can be structured in multiple ways, leading to various interpretations. Prepositional phrase (PP) attachment is a subproblem in natural language parsing and a typical type of structural ambiguity. Choosing the correct attachment can significantly impact the semantic interpretation of a sentence. However, resolving PP attachment ambiguity is challenging because, from a natural language grammar perspective, candidate attachment decisions often appear equally reasonable. Determining the optimal attachment for a preposition often requires semantic interpretation of the words within the sentence. There has been a substantial body of research related to prepositional phrase attachment ambiguity.

- **Structure-based Methods:** these methods are heuristic strategies for resolving prepositional phrase (PP) attachment ambiguities, based on high-level observations of human parsing preferences. Several studies have proposed structure-based suggestions for resolving ambiguity, and these strategies are appealing because of their simplicity, as they do not require calculations about semantics or discourse. There are primarily two structural methods, Right Association and Minimal Attachment methods. The Right Association method proposed by Kimball (1973) suggests that a word tends to attach to the adjacent word on their right side. The Minimal Attachment method introduced by Frazier (1979) posits that words prefer to attach with as few additional syntactic nodes as possible to existing non-terminal words. In essence, it aims to construct a syntax tree with the fewest nodes. However, in the classic case of prepositional phrase attachment ambiguity, as illustrated in an example like "I saw the man with the telescope", when attaching prepositional phrases in a verb + object context, these two principles yield opposite attachment predictions. The Right Association method predicts attachment to the noun, while the Minimal Attachment method predicts attachment to the verb. Besides, another study of PP attachment involving written responses to samples from the "Wizard of Oz" travel information has shown that both Right Association and Minimal Attachment fail to account for over 55% of cases (Whittemore et al., 1990). Even, only 36% of cases act according to strict Minimal Attachment method. Moreover, experiments conducted by Taraban and McClelland (1988) also suggest that structural models do not effectively predict human behavior in resolving ambiguities. As a result, although these methods are straightforward, practical applications have revealed that they often perform inadequately.

  Ratnaparkhi et al. (1994) developed a statistical model to compute the probabilities of attachment decisions. They employed a maximum entropy model that considers subsets of the quadruples v, n1, p, n2 from Wall Street Journal, with each subtuple having weights indicating their predictive strength for either noun or verb attachment. These weights were trained to maximize the likelihood of the training data. They conducted experiments with word features and word class features, achieving attachment accuracy of 81.6% (words and classes) and 77.7% (words only).

However, these methods faced the issue of sparsity, where data observed in the test set might not have been included in the training data. To overcome this, Collins and Brooks (1995) introduced a back-off model that achieved an accuracy of 84.5%. This model used subsets of the quadruples and maintained frequency counts of single words, pairs and triples. Another approach to mitigating sparsity involved using WordNet (Fellbaum, 1998) classes, substituting nouns with their corresponding WordNet classes, which resulted in less sparse corpus statistics (Stetina and Nagao, 1997; Toutanova et al., 2004). Additionally, clusters of similar nouns and verbs derived from the corpus were utilized (Pantel and Lin, 2000).

- **Rule-based Methods:** Brill and Resnik (1994) proposed a method for learning a collection of transformation rules obtained from a corpus. They employed a greedy search to learn a series of transformations that could minimize the error rate on the training data. These transformations are rules that make attachment decisions based on up to three elements of the (v, n1, p, n2) quadruples. Typical examples include rules like "Choose noun attachment if P=of" or "Choose verb attachment if V=buy and P=for". However, these rules could be overly specific and lack generalizability, leading to low recall. To address this issue, word-class information from WordNet was integrated into the model by them, enabling transformations to consider both classes and words. For example, if n2 belongs to a time semantic class, then verb attachment is selected. This approach achieved 80.8% accuracy (only words) and 81.8% accuracy (words and semantic classes).

2.2 Garden Path Effect

The garden path effect is a phenomenon in psycholinguistics, a field that studies the cognitive processes involved in language comprehension. It is another syntactic ambiguity, and also refers to the temporary ambiguity (Frazier and Fodor, 1978) that can occur when a listener or reader encounters a phrase or sentence that initially appears to be structured one way but is later revealed to have a different grammatical structure or meaning. This leads to a momentary "detour" in understanding the sentence, similar to walking down a garden path and then realizing you are on the wrong path (Bever, 1970). The garden path effect is caused by the ambiguity in sentence structure and the human brain's natural tendency to interpret sentences based on the most common or expected grammatical patterns. When a sentence deviates from these expectations, it can result in momentary confusion and readers tends to slow down because of the high reading time at the disambiguation word. This effect has been used to investigate which information determines people's beliefs about potential parses in the context of given local ambiguous contexts, such as whether factors like world knowledge play a role (Ferreira and Clifton Jr, 1986; Trueswell et al., 1994). Furthermore, Van Schijndel and Linzen (2018a,b) demonstrated the existence of garden path effects in LSTMs in the context of simulating human reading times.

The surprisal hypothesis (Hale, 2001; Levy, 2008) provides a significant explanation for garden path effects, proposing that these slowdowns are a result of the unpredictability of each word appearing in a sentence. It suggests that readers maintain a probability representation of all possible parses of the input while incrementally processing the sentence. The processing difficulty in garden path sentences is associated with the cost of updating this representation, which scales with the negative logarithm of the probability or surprisal of the newly encountered content. The theory anticipates that the slowdown related to garden path sentences can be fully explained by the surprisal variations between the disambiguating region in ambiguous garden path sentences and the corresponding region in matching unambiguous sentences. Hale (2001) has been demonstrated that surprisal derived from grammar-based language models (LMs) which encode the syntax of the sentences explicitly, such as probabilistic context-free grammar model, can qualitatively account for specific cases of syntactic disambiguation difficulty. Subsequently, Demberg and Keller (2008) proved that surprisal induced by broad-coverage grammar-based LM is related to reading times.

Additionally, numerous studies have established connections between the performance of recurrent neural networks (RNNs) and aspects of human language processing (Elman, 1990; MacDonald and Christiansen, 2002), as well as grammaticality judgments (Lau et al., 2017). RNN LMs have also been shown to make sufficient syntactic predictions (Linzen et al., 2016; Gulordava et al., 2018). Surprisal derived from RNNs has been found to be a robust predictor of human reading times (Frank and Bod, 2011; Goodkind and Bicknell, 2018) and capable of making reading time predictions comparable to grammar-based language models (Van Schijndel and Linzen, 2018a). Although Van Schijndel and Linzen (2021) later assessed surprisal related to garden path sentences using LSTM LMs trained on large natural language corpora and found that the costs of word predictability derived from the language model significantly underestimated the extent of human garden path effects. This suggests that predictability is not the sole factor contributing to processing costs associated with garden path sentences.

In addition to validating the surprisal theory across different models, Futrell et al. (2019) tested multiple LMs on two major local ambiguities causing garden path effects and posed two main questions for each ambiguity. Firstly, does the network exhibit the fundamental garden path effect, indicating it possesses a syntactic state representation that makes disambiguators surprising? Secondly, is the network sensitive to subtle lexical cues indicating syntactic structure? We will explore these similar problems on Tree Transformer.

2.3 Grammar Induction

Human language exhibits rich hierarchical structures. Parse trees with hierarchical structures can effectively capture these human intuitions. However, annotating parse trees can be challenging and expensive, which has led to the reliance on supervised syntactic parsers in most prior works. While supervised parsers can achieve high performance on well-constructed sentences, only a limited number of languages have treebank data available for supervised training parsers. Additionally, in some cases, syntax rules may be violated, such as in the case of tweets, and over time, syntax rules of languages may evolve. These limitations restrict the generalization capabilities of supervised parsers, rendering them inadequate. Therefore, the task of learning latent tree structures from raw text without manual annotation has become a significant concern in natural language processing, referred to as grammar induction (Smith and Eisner, 2005), and has garnered more attention from researchers in recent years. Grammar induction has a long and rich history in the field of NLP. However, early attempts at purely unsupervised grammar induction were mostly discouraging. Clark (2001) and Klein and Manning

(2002) emerged as some of the earliest successful statistical methods for grammar induction. Notably, Klein and Manning (2002) introduced the constituent-context model (CCM), which clearly models distituents and constituents, serving as the foundation for many subsequent works.

Over the years, there has been significant interest in incorporating tree structures into neural networks to induce tree structures for improved natural language sentence representations. One such attempt was to formulate the problem as a reinforcement learning (RL) task (Yogatama et al., 2016). In its setup, an unsupervised parser acts as an actor, with parsing operations treated as its actions. The actor strives to maximize the total reward, which corresponds to the performance on downstream tasks. Other works have primarily concentrated on generating tree structures from recurrent neural networks and recursive neural networks. PRPN, short for Parsing-Reading-Predict Network (Shen et al., 2018a), is a model that can simultaneously infer syntactic structures from unannotated sentences and utilize the inferred structures to improve language modeling. It consists of three components: a differentiable neural parsing network that employs convolutional neural networks to compute syntactic distances, representing syntactic relationships among all consecutive word pairs in a sentence; a reading network that leverages these syntactic structures to engage in relevant memory; and a prediction network that uses directly related memory to predict the next token. On-LSTM (Shen et al., 2018b) induces tree structures by introducing ordered neurons into recurrent neural networks and adding inductive biases. This allows hidden neurons to capture both short-term and long-term information using the novel gating mechanism and activation function. Another work proposed by Kim et al. (2019b) delved into the unsupervised learning of recurrent neural network grammars (RNNG), also referred to as URNNG. RNNG, as proposed by Dyer et al. (2016), represents a novel probabilistic model for sentence generation, explicitly capturing nested, hierarchical relationships among phrases and words. Given the inherent complexity of directly marginalizing the space of latent trees, URNNG utilized amortized variational inference between a tree structure inference network and the RNNG decoder. This strategy encouraged the decoder to generate rational tree structures. One work related to recursive neural network is deep inside-outside recursive autoencoders (DIORA), proposed by Drozdov et al. (2019), aiming to predict each word in the input sentence based on the remaining part of the sentence and employing inner-outer dynamic programming to account for all potential binary trees within the sentence. The algorithm ultimately extracts the highest-scoring parse. Additionally, there is the compound PCFG proposed by Kim et al. (2019a). It first employs a probabilistic context-free grammar (PCFG) to derive sentences from a corpus, and then maximizes the marginal likelihood of these sentences to realize grammar induction. This allows for capturing dependencies over longer ranges in tree-based generation processes.

Later, in an effort to make Transformer's learned attention more interpretable and to enable hierarchical language understanding, Wang et al. (2019) integrated tree structures into a bidirectional Transformer encoder, resulting in the Tree Transformer. The core idea is to constrain words at each layer to attend only to other words within the same constituent. In the previous work by Wu et al. (2018), the effectiveness of such constraints had already been demonstrated. Unlike previous work that required supervised parsers, the tree structures are automatically derived by Tree Transformer from the original text using the proposed "Constituent Attention" module. Additionally, inspired by Tree-RNNs (Tai et al., 2015) but not entirely the same, while Tree-RNNs represent each phrase and sentence with its constituent sub-phrases, the Tree Transformer progressively combines several smaller constituents from lower layers into larger ones as it moves to higher layers. We will explore the parsing capabilities of Tree Transformer.

3 Fundamentals

In this section, the evaluation metrics and models used in the experiments will be introduced in detail, including many formulas.

3.1 Evaluation Metrics

Accuracy: accuracy is a straightforward and easy-to-understand metric of overall correctness in a classification task. It calculates the proportion of correctly predicted instances out of the total number of instances in the dataset. It is a good metric for balanced datasets where the classes are roughly equally distributed. However, it can be misleading in imbalanced datasets. In addition, one drawback of accuracy is that it does not take into account how close the prediction is to the actual value if the prediction is wrong.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{1}$$

Precision: precision is a metric that measures how many of the predicted positive instances were actually positive. It quantifies the model's ability to make precise positive predictions and is particularly relevant in situations where false positives should be minimized. Besides, precision is valuable when addressing imbalanced datasets where one class considerably outnumbers the other. In such cases, it provides insight into how well the model is performing on the minority class, which is often of greater interest. Precision is calculated as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2}$$

Recall: recall, also referred to as True Positive Rate or Sensitivity, measures how many of the actual positive instances were correctly predicted as positive. It is also useful for the imbalanced datasets and ensures that the model does not miss important positive instances, even if they are rare. Recall is calculated as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3}$$

F1-Score: the F1 score is derived by taking the harmonic mean of precision and recall. It strikes a balance between these two metrics, making it useful when both false positives and false negatives need to be minimized, which is beneficial to the imbalanced datasets. F1-Score is calculated as:

$$\text{F1-Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \tag{4}$$

Surprisal: according to the surprisal theory and existing researches, the garden path effects in neural language models can be quantified by computing surprisal, which involves assigning the negative log probabilities to each word in a sentence as:

$$\text{surprisal}(w_i) = -\log_2 p(w_i|w_{1\ldots i-1})$$

where w_i represents the current word or character, and the probability distribution over sequences of input is calculated by applying the softmax function to the output of the Tree Transformer. The logarithm uses base 2, thus measuring surprisal in bits.

3.2 Tree Transformer

The Transformer architecture introduces self-attention mechanism and positional encoding to address challenges in processing long sequences and has achieved significant success in NLP as well as other domains. However, it is still difficult to interpret the information captured by learned attention of pre-trained Transformer. Therefore, there is a prior work (Wang et al., 2019) introduced Tree Transformer which allows the model to better learn human intuition about hierarchical structures by incorporating tree structures into self-attention. Specifically, an additional "Constituent Attention" module is inserted into the bidirectional Transformer encoder, which adds a constraint "Constituent Prior" to the attention heads in order to guide the attention heads to follow the tree structures and decide whether two words belong to the same constituent.

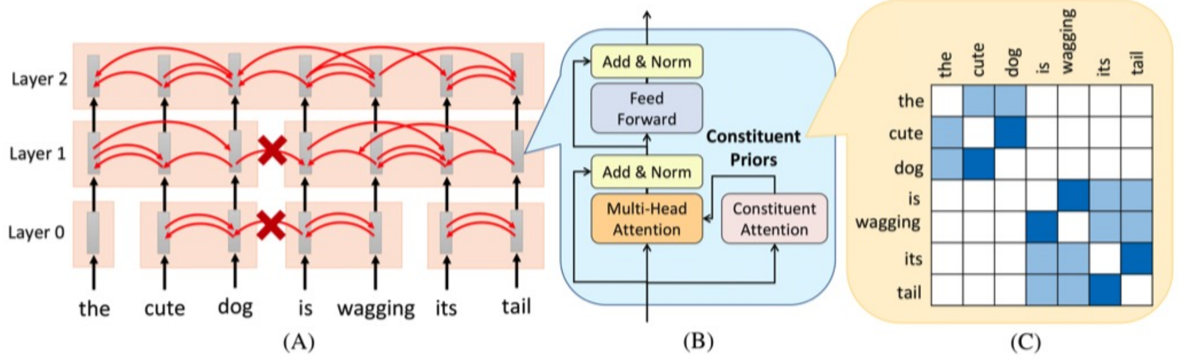


Figure 1: (A) A 3-layer Tree Transformer, where constituents are induced from the input sentence as blocks. Two adjacent constituents might be grouped into one in the subsequent layer, leading to an increasing size of constituents one layer to another. Red arrows represent self-attention between words. (B) Building blocks of Tree Transformer. (C) Constituent prior C in the layer 1. (Wang et al., 2019)

Therefore, when given a sentence as input, Tree Transformer generates a tree structure. Figure 1(A) illustrates a 3-layer Tree Transformer. The building blocks of the Tree Transformer, as shown in Figure 1(B), are similar to the ones used in the bidirectional Transformer encoder, with the addition of the proposed constituent attention module. The blocks in Figure 1(A) are induced constituents from the input sentence. The red arrows represent self-attention. Words belonging to different constituents are restricted from attending to each other, denoted by the red crosses. At layer 0, some adjacent words are combined into constituents; for example, given the sentence "The cute dog is wagging its tail", Tree Transformer automatically deduces that "cute" and "dog" are in one constituent, and "it" and "tail" are in another constitute. Two adjacent constituents may be merged together in the next layer, resulting in a gradual increase in constituent size from one layer to another. Finally, at the highest layer (layer 2), all words merge into a single constituent. Consequently, a parse tree with hierarchical structure is formed. Since all words belong to the same constituent at this layer, attention heads can attend to any other word freely. In the last layer, the behavior of Tree Transformer is similar to a typical Transformer encoder. Figure 1(C) represents the Constituent prior C for the layer 1, and the intensity of color indicates the attention magnitude between words.

Next, let's delve into the Constituent Attention module and its integration with the self-attention mechanism of the original Transformer.

3.2.1 Constituent Prior

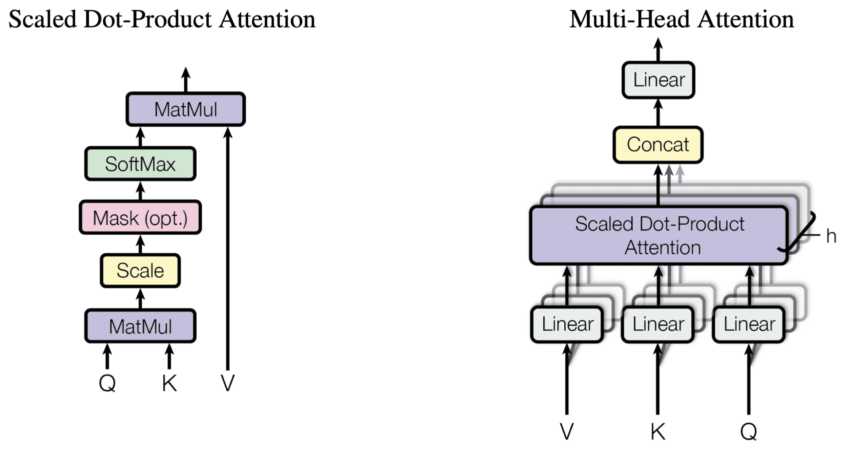


Figure 2: The Transformer uses two attention mechanisms: (left) Scaled Dot-Product Attention and (right) Multi-Head Attention. (Vaswani et al., 2017)

The Transformer uses a scaled dot-product attention in each layer. As shown in the Figure 2 (left), the first step is to calculate the dot product of query matrix Q and key matrix K. Both matrices are composed of vectors with dimensions d_k. This is because, in practice, computing the attention function for an entire set of queries simultaneously is more efficient. In order to do so, these queries are packed into a matrix Q. Similarly, matrices K and V can be obtained. In addition, due to the mismatch in matrix size, what is actually calculated is the dot product of matrix Q and transposed matrix K. The second step is to scale the matrix obtained after the dot product. It can prevent the vanishing gradient problem faced later in softmax. To scale it, the scaling factor of $1/\sqrt{d_k}$ is applied. Finally, passing the processed matrix to the softmax function gives us the weights of the values. This is the so-called attention probability matrix, denoted as E, with dimensions $N \times N$, where N corresponds to the number of words in the input sentence. $E_{i,j}$ is the probability that the word at position i attends the word at position j. The equation is as follows:

$$E = \text{softmax}(\frac{QK^T}{\sqrt{d_k}}) \tag{5}$$

However, for E in the Tree Transformer, it not only depends on the matrices Q and K, but also on the additional Constituent Prior C mentioned. Similar to E, Constituent Prior C is also a $N \times N$ matrix. The difference is that the C is symmetric and $C_{i,j}$ has the same value as $C_{j,i}$, where $C_{i,j}$ represents the probability that word w_i and word w_j are members of a common constituent. In order to force attention within the constituents and prevent each position from attending to positions in other constituents, Tree Transformer restricts the attention probability matrix E according to Constituent Prior C, as shown below:

$$E = C \odot \text{softmax}(\frac{QK^T}{\sqrt{d_k}}) \tag{6}$$

where $\odot$ is element-wise multiplication. When the value of $C_{i,j}$ is small, it indicates that positions i and j are likely to belong to different constituents, where the attention weight $E_{i,j}$ will be smaller. On the contrary, when $C_{i,j}$ has a larger value, then positions i and j have a high likelihood of being part of the same constituent, in which the attention weight $E_{i,j}$ will be larger.

As shown in Figure 2 (right), since Transformer employs multi-head attention with h different heads, which allows the attention mechanism to be run h times in parallel. Queries and keys are linearly projected into d_k dimension h times by using different learned linear projections. Thus, h different query matrices Q and key matrices K are obtained at each layer. In a given layer, all attention heads in multi-head attention commonly use the same C, but each layer possesses its individual Component Prior C which is distinct and not shared across layers. Then, the ultimate output of the multi-head attention module with dimension $d_{model} = h \times d_k$ can be obtained by concatenating the outputs of all attention heads.

3.2.2 Constituent Attention Module

How does the Constituent Attention module generate Constituent Prior C? As mentioned before, the Constituent Attention module can automatically convert the raw text into a tree that captures the syntactic relationship between words by continuously merging different constituents from low layers to high layers. Then the generation of Constituent Prior C is actually to estimate the breakpoints between constituents, or calculate the probability that two adjacent words belong to the same constituent. Therefore, the Constituent Attention module owns a sequence $a = \{a_1, ..., a_i, ..., a_N\}$ in each layer, where a_i is the probability of the word w_i and its neighbor word w_{i+1} located in the same constituent. Smaller values of a_i mean that there is a lower probability that w_i and w_{i+1} exist in the same constituent, that is, there is a breakpoint between them. With the sequence a, Constituent Prior C can be obtained. The probability $C_{i,j}$ that word w_i and word w_j belong to the same constituent is calculated by the multiplication of all $a_{i \leq k < j}$ between these two words, as follows:

$$C_{i,j} = \prod_{k=i}^{j-1} a_k \tag{7}$$

Since if one of $a_{i \leq k < j}$ between two words w_i and word w_j is smaller, the value of $C_{i,j}$ obtained after multiplying all of a will also become smaller. Therefore, choosing multiplication over summation makes the probability gap larger, resulting in a more pronounced component distribution. Then, in practice, multiplication may lead to a probability vanishing problem due to the existence of probability 0. To avoid this, applying log-sum on the basis of multiplication can effectively achieve:

$$C_{i,j} = e^{\sum_{k=i}^{j-1} \log(a_k)} \tag{8}$$

As a result, the process of deriving Constituent Prior C from sequence a becomes clear. Regarding the acquisition of sequence a, it is imperative to introduce the following two mechanisms: $Neighboring\ Attention$ and $Hierarchical\ Constraint$.

- **Neighboring Attention:**

  In line with its name, Neighboring Attention involves computing attention between adjacent words. Similar to equation (5), it utilizes scaled dot-product attention to calculate the score $s_{i,i+1}$ which indicates w_i links to w_{i+1}. However, the difference lies in the fact that the query and key vectors in this context are computed by the same network architecture but with distinct sets of network parameters. Additionally, the dimensionality of both vectors is d_{model} rather than d_k. The specific calculation of $s_{i,i+1}$ is as follows:

$$s_{i,i+1} = \frac{q_i \cdot k_{i+1}}{d} \tag{9}$$

  where q_i represents the link query vector of w_i, and k_{i+1} represents the link key vector of w_{i+1}. Their dot product $q_i \cdot k_{i+1}$ signifies the likelihood that w_i and w_{i+1} share as part of the same constituent. Here, the scaling factor d is set to be $\frac{d_{model}}{2}$.

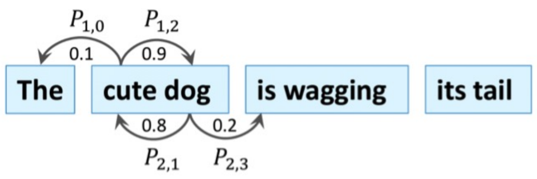


Figure 3: Neighboring Attention: the probabilities of the word only linking to its left and right neighboring words.(Wang et al., 2019)

  However, in the absence of other restrictions, the model tends to link all words together and allocate all words to a single constituent. In other words, if both $s_{i,i+1}$ and $s_{i,i-1}$ have large values, the attention heads are free to link to any position without the constraint of constituent prior, similar to the original Transformer model. Therefore, in comparison to scores s between neighboring words, transforming these scores into a probability distribution where one receives high probabilities (indicating strong attention) while another receives lower probabilities (indicating weak attention or no attention at all) is an effective sparsity constraint to make the attention mechanism more interpretable. As shown in the Figure 3, each word in the raw text is constrained to link only to its right neighbor and left neighbor. For instance, the scores of w_i linking to w_{i+1} and to w_{i-1} are normalized to obtain the probabilities of the word w_i attending to its left and right neighboring words, $p_{i,i+1}$ and $p_{i,i-1}$, respectively. Here, the softmax function is applied to the two attention links of w_i to enforce the attention to be sparse:

$$p_{i,i+1}, p_{i,i-1} = \text{softmax}(s_{i,i+1}, s_{i,i-1}) \tag{10}$$

where $p_{i,i+1} + p_{i,i-1} = 1$. Since $p_{i,i+1}$ and $p_{i,i-1}$ may have different values, averaging the probabilities of the two attention links results in $\hat{a}_i$, which means that adjacent words are linked only when the two words mutually attend to each other. The calculation process is illustrated as follows:

$$\hat{a}_i = \sqrt{p_{i,i+1} \times p_{i+1,i}} \tag{11}$$

which will be used in the next part to generate a_i and further obtain the sequence a.

- **Hierarchical Constraint:**

  If $Neighboring\ Attention$ was described as a horizontal constraint on word-to-word links within a layer, then $Hierarchical\ Constraint$ can be seen as a vertical constraint between layers. This is because, to obtain the tree structure of the raw text, it is essential to ensure that the words which are merged into the same constituent at lower layers continue to belong to the same constituent at higher layers. To implement this, a constraint is imposed on the link probability a for each word at each layer. Specifically, it is required that a_k^l (the link probability for a word at index k in layer l) should always be greater than a_k^{l-1} (the link probability for the same word in the previous layer, $l-1$). As a result, the link probability a_k^l is computed as follows:

$$a_k^l = a_k^{l-1} + (1 - a_k^{l-1})\hat{a}_k^l \tag{12}$$

  where $\hat{a}_k^l$ is derived from $Neighboring\ Attention$. This constraint ensures that as the model moves up through layers, it respects the hierarchical structure of constituents that were established in the lower layers. It maintains consistency in how words are grouped into constituents, helping the model capture hierarchical dependencies and structures within the input data.

  Finally, at layer l, the link probabilities a^l are used to compute the constituent representation C^l using equation (8). Initially, at the lowest layer, different words are treated as distinct constituents, and the link probabilities a_k^{-1} are initialized to zero.

3.3 Constituent Parsing with the Sequence Labeling Model BiLSTM

Constituent parsing plays a pivotal role in the field of NLP. It is a syntactic analysis technique that focuses on understanding the tree structure of sentences, including phrases and their relationships. Through constituent parsing, we gain a deeper insight into the grammatical components of a sentence, such as noun phrases, verb phrases, and their hierarchical organization.

Different from Tree Transformer, Gómez-Rodríguez and Vilares (2018) proposed a supervised learning method to transform constituent parsing into a sequence labeling task. It can be divided into three parts: encoding, prediction and decoding.

3.3.1 Encoding

The encoding function linearizes a tree into a sequence of labels equal to the sentence length subtracted by one. For each input word w_t, it generates a label encoding the quantity of ancestors shared by the words w_t and w_{t+1}, as well as the non-terminal symbol at the lowest common ancestor. This encoding function has been proven to be injective for every tree that does not contain unary branches. Specifically, injectivity here means that each unique sequence of labels should have a unique corresponding tree, and two different sequences of labels should not map to the same tree. Moreover, after applying collapsing techniques, this method is made extensible to parse unary chains. The encoding function involved and the collapsing technique adopted in this experiment will be detailed below.

- **Encoding function:**

  In the following, bold font is used to represent vectors and matrices (e.g. $\mathbf{x}$). The input sequence of words is initially defined as $\mathbf{w} = [w_1, w_2, ..., w_{|w|}]$, where $w_i \in V$. $T_{|w|}$ represents the set of constituent trees with $|w|$ leaf nodes and no unary branches. The constituent parsing problem is then assumed to involve mapping each sentence $\mathbf{w}$ to a tree in $T|w|$. In other words, the correct parsing is assumed to have no unary branches. This mapping can be achieved through the encoding function $\Phi_{|w|} : T_{|w|} \rightarrow L^{|w|-1}$. By defining a set of labels L, each tree in T_w is encoded into a unique label sequence in $L^{|w|-1}$.

  Using w_i to represent the word at position i in the sentence, where $1 \leq i \leq |w| - 1$, its label is set to a 2-tuple $l_i = (n_i, c_i)$, where n_i is an integer encoding the number of common ancestors between w_i and w_{i+1}, c_i is the non-terminal symbol at the lowest common ancestor. As illustrated in Figure 4, the number of common ancestors can be encoded using both the absolute scale and the relative scale. Absolute scale encoding is relatively straightforward, where n_i is identical to the number of common ancestors between w_i and w_{i+1}. On the other hand, relative scale encoding involves representing n_i as the difference in the number of ancestors relative to what is encoded in n_{i-1}. The latter significantly reduces the size of the label set. Therefore, in the experiment, relative scale encoding will be adopted.

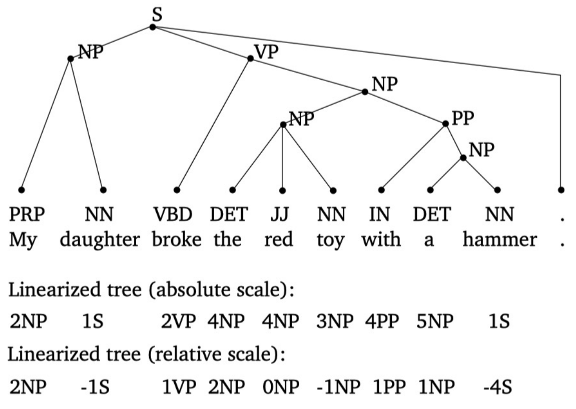

S
NP
VP
NP
NP
PP
NP
PRP My
NN daughter
VBD broke
DET the
JJ red
NN toy
IN with
DET a
NN hammer
. .
Linearized tree (absolute scale):
2NP 1S 2VP 4NP 4NP 3NP 4PP 5NP 1S
Linearized tree (relative scale):
2NP -1S 1VP 2NP 0NP -1NP 1PP 1NP -4S

Figure 4: An example of a constituent tree linearized using absolute (up) and relative (down) scales.

It is important to note that this encoding method does not necessarily linearize the tree into a binary tree. Additionally, empirical observations suggest that certain tokens, typically connected directly to the root node (e.g., the final punctuation in the Figure 4), make it challenging for the model to learn effectively. To address these situations successfully in practice, a simplified labeling scheme is applied. When a node is directly connected to the root of the tree, the node is assigned a special label (ROOT, c_i).

- **Collapsing technique:**

  Due to the practical limitation that the encoding function cannot handle unary branches, an extension is required. As illustrated in the Figure 5, the encoding function $\Phi_{|w|}$ cannot directly assign non-terminal symbols to unary branches because there are no word pairs (w_i, w_{i+1}) sharing common ancestors. These unary branches can be categorized into two types: intermediate unary chains, representing unary chains that eventually become non-terminal symbols (e.g., $X \to Y$ in Figure 5), and leaf unary chains that name chains generating POS tags (e.g., $Z \to T_5$).

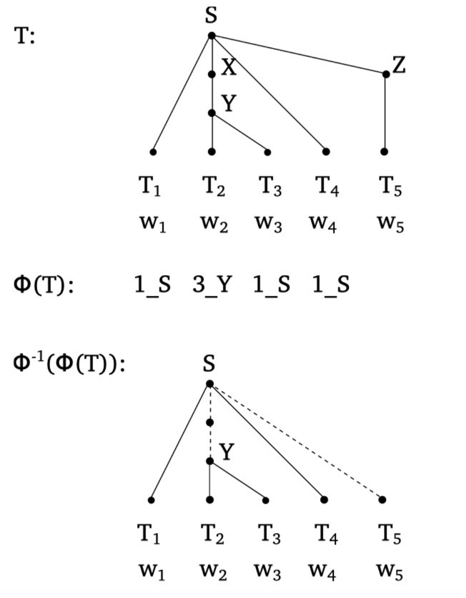


Figure 5: An example of a tree that cannot be directly linearized by the encoding function $\Phi_{|w|}$. w_i and T_i represent a word and its POS tag, respectively. Dashed lines indicate branches that, when decoded after directly applying $\Phi_{|w|}$, result in incorrect decoding. The non-terminal symbol for the second ancestor of $w_2(X)$ cannot be decoded because there is no word pair with X as their lowest common ancestor. A similar situation can be observed at the nearest ancestor for $w_5(Z)$.

Intermediate unary chains are collapsed into a chained single symbol, but they can be encoded using $\Phi_{|w|}$ like any other non-terminal symbols. However, leaf unary chains are collapsed together with POS tags, and they cannot rely on $\Phi_{|w|}$ for encoding and decoding because the encoding function assumes a fixed sequence of leaf nodes and does not explicitly encode them. To address this issue, the encoding function can be extended to transform the label l_i into a triple (n_i, c_i, u_i), where u_i encodes the collapsed label of the leaf unary chain for w_i (if any), or is none otherwise. This extended encoding function is referred to as $\Phi'_{|w|}$.

3.3.2 Prediction

The simplification of the constituent parsing problem into a sequence labeling task aims to predict the function $F_{|w|,\theta} : V^{|w|} \rightarrow L^{|w|-1}$, where θ represents the parameters to be learned. This function generates output labels for each token in the input sequence. State-of-the-art neural models for sequence labeling, such as bidirectional long short-term memory networks (Bi-LSTM), have found widespread applications. Figure 6 shows the network architecture we used.

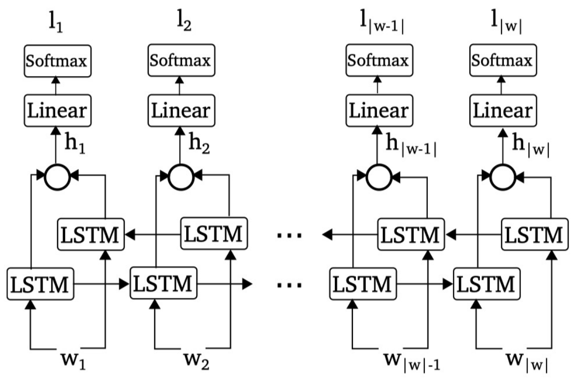


Figure 6: Architecture of bidirectional long short-term memory network.

Defining LSTM$(\mathbf{x})$ as a long short-term memory network for processing a sequence $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{|\mathbf{x}|}]$, the encoding of its i-th element, BiLSTM$(\mathbf{x}, i)$, is defined as:

$$\text{BiLSTM}(\mathbf{x}, i) = \mathbf{h}_i = \mathbf{h}_i^l \circ \mathbf{h}_i^r = \text{LSTM}^l(\mathbf{x}_{[1:i]}) \circ \text{LSTM}^r(\mathbf{x}_{[|\mathbf{x}|:i]})$$

In the case of multiple layers of BiLSTM, the output of BiLSTM$_m$ serves as input fed into BiLSTM$_{m+1}$. The output label for each w_i is ultimately predicted as softmax$(W \cdot \mathbf{h}_i + b)$, where W and b are weights and biases.

When processing a sentence $[w_1, w_2, ..., w_{|w|}]$, the model takes as input a sequence of embeddings $[\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w}_{|\mathbf{w}|}]$. Each $\mathbf{w_i}$ is formulated as $\mathbf{w}_i \circ \mathbf{p}_i \circ \mathbf{ch}_i$, where $\mathbf{w}_i$ and $\mathbf{p}_i$ represent the word and Part-of-Speech (POS) tag embeddings, and $\mathbf{ch}_i$ is a word embedding derived from the initial character embedding layer which relies on a BiLSTM as well.

3.3.3 Decoding

To parse a sentence, $F_{|w|,\theta} \circ \Phi^{-1}_{|w|}$ decodes the obtained label sequence into a constituent tree. However, due to the non-surjective nature of the encoding function, in other words, the encoding function does not cover the entire range of possible output labels, not every sequence of $|w| - 1$ label pairs in the form (n_i, c_i) has a corresponding tree in $T_{|w|}$. Specifically, there are two cases where label sequences do not formally represent a tree. Therefore, $\Phi^{-1}_{|w|}$ is not a complete decoding function, and additional handling is required for the following situations:

- **Sequences with conflicting non-terminals:** For branches greater than two, a non-terminal can be the lowest common ancestor of multiple consecutive word pairs. For example, in the tree in Figure 4, "the" and "red," as well as "red" and "toy," share the same NP node as the lowest common ancestor, corresponding to the label sequence $c_4 = NP, c_5 = NP$. However, if this sequence is adopted and the setting is changed to $c_5 = VP$, the label sequence cannot strictly correspond to any tree encoding. This is because two elements referencing the same node represent different non-terminal labels, creating a contradiction. Therefore, during the decoding process, when there are multiple conflicting non-terminals at a given position in the label sequence in the tree, $\Phi_{|w|}^{-1}$ is calculated simply using the first such non-terminal, ignoring the rest.

- **Sequences generating unary structures:** some sequences of values n_i do not correspond to a tree in $T_{|w|}$, because the only tree structures that satisfy these values include unary branches, leading to an unspecified non-terminal for each. This is also illustrated in the sequence labels $(1, S), (3, Y), (1, S), (1, S)$ mentioned in Figure 4. This sequence, when decoded, cannot correspond to the original tree structure. Therefore, similar to the collapse of unary chains, any generated unary nodes are considered invalid and removed.

4 Methods

In this section, firstly, an introduction to all the datasets used in the experiments will be provided. Following that, the algorithm employed by the Tree Transformer to convert input sentences into parse trees will be presented. Finally, the entire experimental setup will be elaborated in great detail.

4.1 Data

The entire experiment utilized five datasets. The Penn Treebank was used to train and test models. The Prepositional phrase attachment corpus was employed to generate sentences with prepositional phrase attachment ambiguity and further used to analyze the models' ability to attachment disambiguation. Finally, concerning the investigation of the models' capability to eliminate garden path effect, we took these experiments from three datasets, which are the Futrell et al. (2019) ones: Main-verb/Reduced-relative, NP/Z (Overt Object) and NP/Z (Verb Transitivity).

4.1.1 Penn Treebank

The Penn Treebank (PTB) (Marcus et al., 1993) was created to facilitate research in natural language parsing and syntax analysis. It contains a large corpus of English text, primarily from the Wall Street Journal (WSJ), which has been parsed and annotated according to a specific syntactic tree structure. It is widely used to train and evaluate syntactic parsers and other NLP models. In the syntactic trees of the Penn Treebank, each word is assigned a POS tag, and the trees capture hierarchical relationships between words, such as noun phrases, verb phrases and clauses.

To be consistent with other studies, the official splits of PTB are used in the experiments: sections 2 to 21 for training (WSJ-train) and 23 for testing (WSJ-test). WSJ-train consists of 39832 sentences, while WSJ-test comprises 2416 sentences, as shown in Table 1.

Dataset	section	Number
WSJ-train	2-21	39832
WSJ-test	23	2416

Table 1: The official splits of Penn Treebank.

4.1.2 Prepositional Phrase Attachment Corpus

The prepositional phrase attachment corpus (Ratnaparkhi et al., 1994) extracted from WSJ treebank refers to a dataset used in NLP and computational linguistics for studying prepositional phrase attachment ambiguity. PP attachment ambiguity arises when a prepositional phrase can be attached to different parts of a sentence, leading to different interpretations. In this corpus, examples are annotated to indicate the attachment of prepositional phrases. It includes examples, $\{v, n1, p, n2\}$ quadruples, where a prepositional phrase$(p, n2)$ can be attached to a noun $n1$ or verb v, resulting in different syntactic structures and meanings. In addition, an attachment label l is included, indicating the actual attachment of this prepositional phrase. Therefore, the format of the data is $\{v, n1, p, n2, l\}$ and the example is shown

Category	Word
Verb v	provide
Noun n_1	services
Preposition p	for
Noun n_2	customers
Label l	V

Table 2: An original example from the prepositional phrase attachment corpus Ratnaparkhi et al. (1994).

Category	Word
Subject $n0$	they
Verb v	provide
Modifier $m1$	various
Noun $n1$	services
Preposition p	for
Modifier $m2$	their
Noun $n2$	customers
Label l	V

Table 3: An example of a converted sentence, preserving the prepositional phrase attachment ambiguity.

in the Table 2.

In the experiments, to distinguish it from the WSJ-train, we used prepositional phrases extracted from the WSJ-test. That is to say, 3097 samples of the prepositional phrase attachment corpus were used to generate sentences which were obtained for subsequent analysis of sentence-level prepositional phrase attachment ambiguity.

4.1.3 Generating Prepositional Phrase Attachment Sentences

With the prepositional phrase attachment corpus, the quadruples are expanded to 7-tuples $\{n0, v, m1, n1, p, m2, n2\}$ to form a complete sentence. Meanwhile, the original attachment labels remain unchanged. This is done by introducing the pre-trained model "bert-base-uncased" (Devlin et al., 2018a) and directly using this model to perform masked language modeling with the pipeline. Prior to this, data cleaning is necessary. Gerunds or present participle, past participles, and the verbs like "be" that do not conform to the required forms are filtered out. Symbols and numbers that fail to meet the requirements are removed. Subsequently, the process of generating the sentence begins.

The first step is to mask the subject position like "[MASK] v $n1$ p $n2$", and take the personal pronoun with the highest score as the subject $n0$. The following step is to mask the modifier of noun $n1$ like "$n0$ v [MASK] $n1$ p $n2$", and the noun, adjective, determiner and possessive pronoun with the highest score can be used as $m1$. Then, the modifier of the second noun $m2$ can be obtained in the same way. The NLTK POS tagger is used to identify words here. As the result, we got 1424 sentences with the format of $\{n0, v, m1, n1, p, m2, n2, l\}$. An example of these sentences is shown in Table 3.

4.1.4 Garden Path Effect Datasets

To explore the impact of Tree Transformer on garden path effect, two different data from Futrell et al. (2019), Main-verb/Reduced-relative and NP/Z Ambiguity data sets, will be used in the experiment.

- **Main-verb/Reduced-relative Ambiguity**:

  First, the experiment will investigate the garden path effect caused by Main Verb/Reduced Relative (MV/RR) ambiguities. In the dataset, there are 28 examples and each of them has 4 sentences. The first verb of the sentences is ambiguous. It can be considered both as the main verb of the sentence and as a word introducing a reduced relative clause. The term "reduced relative clause" refers to a relative clause that lacks an explicit complementizer and starts with a past participle verb. This ambiguity can persist for an extensive stretch of the following context until the disambiguator appears.

  (1) a. The women brought the sandwich from the kitchen **fell** in the dining room
  b. The women given the sandwich from the kitchen **fell** in the dining room
  c. The women who was brought the sandwich from the kitchen **fell** in the dining room
  d. The women who was given the sandwich from the kitchen **fell** in the dining room

  In sentence 1a, the verb "brought" is initially analyzed as part of the main verb phrase, but upon the appearance of the disambiguator "fell", readers realize that "fell" is the actual main verb and the verb "brought" had to be reanalyzed as part of a relative clause. Consequently, in such cases, due to strong prior context bias towards interpreting the verb as the main verb, human comprehension is severely disrupted, making the syntactic interpretation more challenging. This is akin to what is observed in the famous sentence "the horse raced past the barn fell", resulting in a high garden path effect.

  In contrast, the garden path effect theoretically should be reduced or eliminated in sentences like 1b. The verb "given", as the head word of the relative clause, is a past participle form different from the simple past tense verb. Compared to "gave", "given" is more explicit, indicating that it should not be the main verb of the sentence. Therefore, the possibility of misinterpreting the verb as a main verb is lower or excluded before the appearance of the disambiguator "fell". If a language model is sensitive to morphological cues in syntactic structure, it should exhibit a reduced garden path effect or even no garden path effect under such explicit conditions.

  Furthermore, the garden path effect should be eliminated in sentences 1c, where the presence of the words "who was" makes it clear to the reader that the verb "brought" is part of the relative clause and not the main verb of the sentence. Therefore, we can quantify the garden path effect size by model and verb-form ambiguity by calculating the mean surprisal at the disambiguators for sentences 1a minus sentences 1c and sentences 1b minus sentences 1d.

- **NP/Z Ambiguity:**

  This is another highly classical syntactic garden path effect configuration consisting of 24 NP/Z examples and each of them has 4 sentences. In the datasets, noun phrases exhibit local syntactic ambiguity, because they can act as either the direct object of the main verb in the subordinate clause or as the subject of the main clause. The ambiguity persists until the true subject of the main clause emerges, at which point the ambiguity is resolved and the noun phrase is reinterpreted as the latter. This is referred to as "NP/Z" because the verb of the subordinate clause can take either an NP object or Z(ero), indicating a null object. There are two datasets related to this kind of ambiguity, one is to use overt objects to disambiguate, the other is to use intransitive verbs to disambiguate. Additionally, using a comma to mark the end of a clause makes the sentence easier at the disambiguator. Similar to MV/RR, the quantification of the garden path effect size by model and presence of object/embedded verb transitivity can be obtained by calculating the mean surprisal difference at the disambiguators between sentences a and sentences c or between sentences b and sentences d.

– **Overt Object:** As shown in sentence 2a, prior to encountering "burst", "shot" is naturally interpreted as a transitive verb with "the woman" as its direct object. However, upon the appearance of "burst", readers realize that "shot" should be considered as an intransitive verb with "the woman" as the subject of the main clause. In sentence 2b, introducing the explicit object "the gun" to the transitive verb effectively reduces or eliminates the ambiguity before the disambiguator appears.

(2) a. As the gangster shot the woman **burst** into hysterics
b. As the gangster shot his gun the woman **burst** into hysterics
c. As the gangster shot, the woman **burst** into hysterics
d. As the gangster shot his gun, the woman **burst** into hysterics

– **Verb Transitivity:** As shown in sentence 3b, unlike the transitive verb that introduces the object "his gun", it replaces the transitive verb "shot" with the intransitive verb "laughed". Then this verb cannot accept an object in English, so readers should not be misled into believing that "the women" is its object. In theory, replacing transitive verbs with intransitive verbs can reduce the garden path effect. However, this lexical information about syntactic structure is so subtle that it is not known whether human are as sensitive to it as theory.

(3) a. As the gangster shot the woman **burst** into hysterics
b. As the gangster laughed the woman **burst** into hysterics
c. As the gangster shot, the woman **burst** into hysterics
d. As the gangster laughed, the woman **burst** into hysterics

4.2 Unsupervised Parsing from Tree Transformer

For each sentence in our experiments, we use the Tree Transformer model to extract parse trees. After training the model, the neighbor link probability a is obtained. A small value of a suggests that it could be the point where two constituents are separated. The idea is to use a top-down greedy parsing algorithm (Shen et al., 2018a) which is a parsing strategy that starts from a high-level view of sentence structure, makes locally optimal decisions at each step, and recursively dissects the sentence into two constituents based on the minimum a values, forming a parse tree. Top-down greedy parsing performs parsing of sentences without the need for explicit supervision or labeled data. The a values are used as clues to identify potential boundaries between constituents during parsing.

However, the model has multiple layers, and each layer has its set of a^l. As mentioned in section 3.2, a values capture hierarchical information as they move from one layer to another, and using all of them can provide a more comprehensive view of the sentence's structure since a values strictly increase from layer to layer. Therefore, instead of picking a values from a single layer for parsing, a values from all layers of the model are used. Algorithm 1 (Wang et al., 2019) outlines how the hierarchical information from a is used for unsupervised parsing. when looking at the values of a in the top layers of the model, these values are very close to 1 indicate that there is a strong likelihood or probability that neighboring words in the sentence are linked or belong to the same constituent. In other words, the top layers tend to group words together rather than identifying breakpoints between constituents.

Algorithm 1 Unsupervised Parsing from Tree Transformer with Multiple Layers

```
 1: $a \leftarrow$ link probabilities
 2: $m \leftarrow$ minimum layer id        ▷ Discard the $a$ from layers below the minimum layer
 3: $threshold \leftarrow 0.8$            ▷ Breakpoint threshold
 4: procedure BUILD TREE($l, s, e$)       ▷ $l$: layer index, $s$: start index, $e$: end index
 5:     if $e - s < 2$ then               ▷ The constituent is unable to be split
 6:         return $(s, e)$
 7:     end if
 8:     $span \leftarrow a^l_{s \leq i < e}$
 9:     $b \leftarrow$ argmin$(span)$     ▷ Obtain the breakpoint
10:     $last \leftarrow$ max$(l - 1, m)$ ▷ Obtain the index of the last layer
11:     if $a^l_b > threshold$ then
12:         if $l = m$ then
13:             return $(s, e)$
14:         end if
15:         return BuildTree$(last, s, e)$
16:     end if
17:     $tree1 \leftarrow$ BuildTree$(last, s, b)$
18:     $tree2 \leftarrow$ BuildTree$(last, b + 1, e)$
19:     return$(tree1, tree2)$            ▷ Return trees
20: end procedure
```

The unsupervised parsing for Tree Transformer starts parsing from the top layer of the model and progressively moves down through the layers. The process continues until it reaches a designated bottom layer m. The choice of m is an adjustable parameter, typically set to 2 or 3. Layers below m are excluded because they tend to provide less useful information for parsing, which has been observed in previous researches (Liu et al., 2019; Shen et al., 2018b). To determine valid breakpoints for parsing, a $threshold$ value is set. If a minimum a value falls below this $threshold$, it is considered a valid breakpoint, suggesting the separation of constituents. Wang et al. (2019) found that the model's performance is not significantly affected by the specific threshold value chosen, 0.8 is considered low enough to be a valid breakpoint.

4.3 Experimental set-up

4.3.1 Tree Transformer Model

In order to get the Tree Transformer for subsequent performance analysis, the architecture, training and testing process will be elaborated upon below. Except for differences in the processing of the training set, we follow the work of Wang et al. (2019) for other settings.

- **Architecture:**

  In the experiments conducted, the Tree Transformer is constructed based on a bidirectional Transformer encoder. The implementation of the Tree Transformer encoder is identical to that of the original Transformer encoder. For all experiments, the number of self-attention heads, denoted as h, is set to 8. The hidden size d_{model} of Transformer and Constituent Attention are both set to 512, with a feed-forward size of 2048. The dropout rate is set at 0.1.

  Additionally, it is worth noting that prior research has indicated that increasing the number of layers can lead to improved performance, as it empowers the Tree Transformer to represent more complex tree hierarchies. However, Wang et al. (2019) found that performance ceases to improve when the depth exceeds 10. Therefore, the number of layers is fixed at 10 for all experiments.

- **Training Process:**

  The Tree Transformer model undergoes unsupervised training using BERT Masked LM (Devlin et al., 2018b), and employs the WordPiece tokenizer from BERT, as proposed by Wu et al. (2016), to tokenize words resulting in a vocabulary of 13375 tokens. However, instead of using the original WSJ-train to train the model, we first simplify the sentences in WSJ-train into sentences composed of words with certain POS tags:

  [‘CC’, ‘CD’, ‘DT’, ‘EX’, ‘FW’, ‘IN’, ‘JJ’, ‘JJR’, ‘JJS’, ‘LS’, ‘MD’, ‘NN’, ‘NNS’, ‘NNP’, ‘NNPS’, ‘PDT’, ‘POS’, ‘PRP’, ‘PRP$’, ‘RB’, ‘RBR’, ‘RBS’, ‘RP’, ‘SYM’, ‘TO’, ‘UH’, ‘VB’, ‘VBD’, ‘VBG’, ‘VBN’, ‘VBP’, ‘VBZ’, ‘WDT’, ‘WP’, ‘WP$’, ‘WRB’, ‘,’].

  Also, if the word is a number, then it will replace it with a capital “N”. Then, in the simplified sentences, after each word is tokenized by the tokenizer, only the first token is retained. Based on this preprocessed WSJ-train, we use mini-batch training of size 64, with a total of 60000 batches. 60 dimensions are used to represent word embeddings and each word is represented by a vector of 13375 components as the same as the vocabulary size.

  In addition, it is well-documented that optimal results are achieved with the $Adam$ optimizer with a learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.98$. Therefore, during the training process, parameters are configured accordingly.

- **Testing Process:**

  Following the evaluation setting established in prior research, the performance of the Tree Transformer in unsupervised constituency parsing is assessed by calculating F1 scores on the WSJ-test, which is processed in the same way as WSJ-train. The annotated trees in the Penn Treebank are also preprocessed in the same way, only the parts with specific POS tags are retained. Precisely, for each sentence, Tree Transformer generates a parse tree using the previously mentioned unsupervised parsing approach. Meanwhile, the corresponding processed annotated parse tree is considered as the ground-truth tree. In order to facilitate assessment, we convert both trees into span representations. For the example 3, its corresponding parse tree and span representation are illustrated in Figure 7.

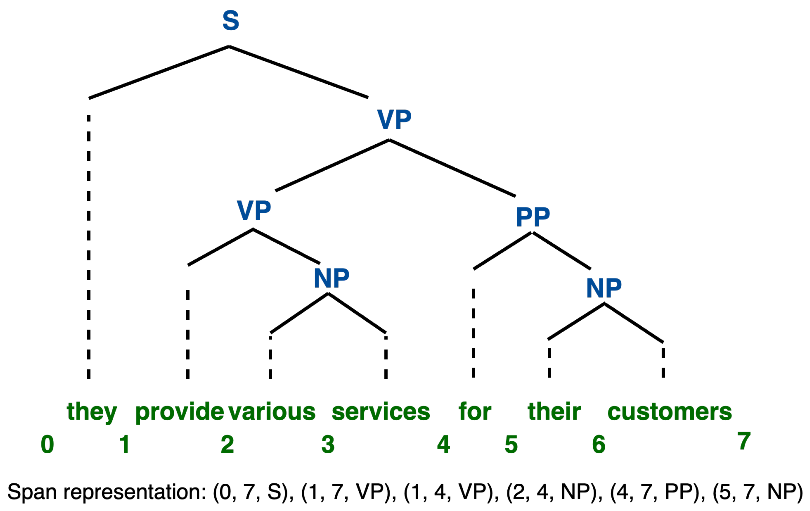

S
VP
VP
PP
NP
NP
they
provide
various
services
for
their
customers
0
1
2
3
4
5
6
7
Span representation: (0, 7, S), (1, 7, VP), (1, 4, VP), (2, 4, NP), (4, 7, PP), (5, 7, NP)

Figure 7: A constituency parse tree with its span representations.

Initially, start from the root node of the tree and traverse each node in the parse tree using depth-first traversal. Then, for each phrase structure, record its starting and ending positions. This can be achieved by maintaining a pointer or index during the traversal process. The starting position is the position of the leftmost child node of the phrase structure, while the ending position is the position of the rightmost child node. Next, for each tagged phrase structure, generate a corresponding span. We use bracket notation to represent it as "(start, end)". For example, the corresponding span for "NP: various services" is (3, 4). Finally, if the parse tree has subtrees, repeating the above steps for each subtree can transform the entire parse tree recursively into spans. Additionally, since parse trees corresponding to pp attachment sentences always include the span (0, 8), we do not consider it when calculating the F1 score. In other words, for the example sentence, the final span representation is (0, 7), (1, 7), (1, 4), (2, 4), (4, 7), (5, 7).

The F1 score is then computed based on the spans of these generated and ground-truth trees. Averaging the F1 scores of all sentences obtains the final F1 score. This evaluation process seeks to measure the proximity of the Tree Transformer's parsing to the ground-truth parsing provided in the Penn Treebank.

Furthermore, as the lower layers of the Transformer do not provide informative representations, utilizing syntactic structures from these lower layers would lead to a deterioration in parsing quality. Conversely, when the a value for the top few layers approaches 1, most of the syntactic information becomes lost. Consequently, excessively large values of m in parsing algorithm would also negatively impact performance. Hence, based on findings from existing research, it has been determined that setting the designated bottom layer m to 3 yields the optimal performance.

4.3.2 BiLSTM Model

In order to study the performance gap between Tree Transformer and supervised parser. We use the pre-trained BiLSTM model, i.e. $BILSTM^{\Phi'}_{m=2,e,ch}$, from Gómez-Rodríguez and Vilares (2018) to make additional comparison. Likewise, its architecture, training procedure from prior work and testing procedure applied by us will be introduced.

- **Architecture:**

  We use a pre-trained BiLSTM model, which utilizes NCRFpp, a sequence labeling framework grounded on bidirectional short-term memory networks (Yang and Zhang, 2018). The number of layers m is configured to 2. The dropout rate is set to 0.5. Both the left-to-right and right-to-left LSTMs independently generate hidden vectors of size 400.

- **Training Process:**

  The model is trained over the original WSJ-train for 100 epochs using mini-batches of size 8. Word, POS tag and character embeddings are represented in dimensions of 100, 30 and 20, respectively. Word embeddings are adapted from pre-trained GloVe embeddings (Pennington et al., 2014). Character embeddings are additionally processed through a BiLSTM, with the output hidden layer of the character embedding layer set to 50.

  Furthermore, the optimizer employed is SGD, with an initial learning rate of 0.2, momentum set at 0.9, and linear decay set at 0.05.

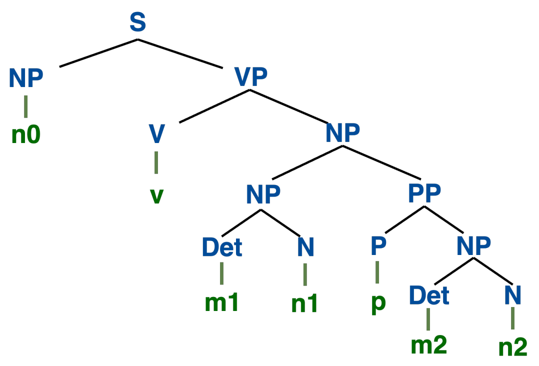
- **Testing Process:**

  For comparison with Tree Transformer, the preprocessed WSJ-test is used in the test process. The predicted sequence labels will be decoded into a parse tree, which will then be converted into span representation. And the calculation way of F1 score remains consistent.

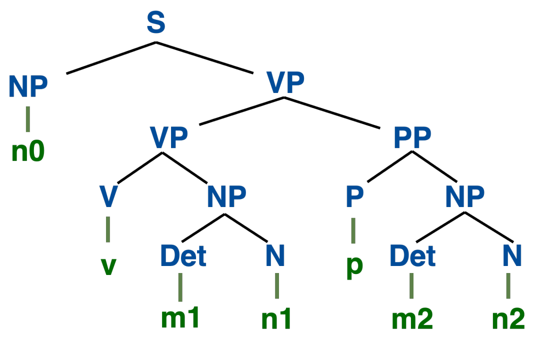
4.3.3 Parsing PP Attachment

- **Ground-truth Parse Trees:** according to the attachment being noun or verb, the corresponding binary ground-truth parse trees are obtained. Figure 8(a) shows the prepositional phrase attached to the noun, whereas Figure 8(b) presents the prepositional phrase attached to the verb. However, for BiLSTM, the generated parsing tree is not necessarily a binary tree, thus allowing for the possibility of ternary ground-truth parsing trees which are shown in Figure 8(c) and 8(d) indicating prepositional phrase attachment decision are noun or verb, respectively.

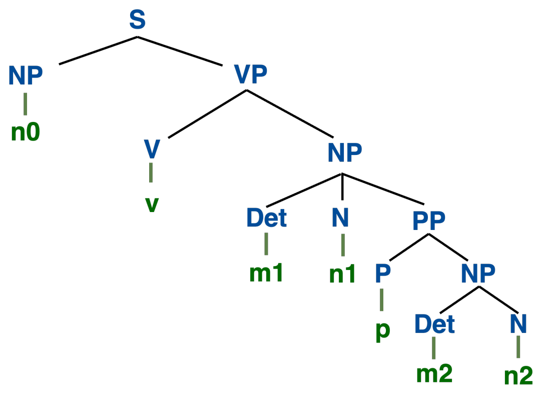
- **Parsing Structure Analysis:** since the data used in the experiment are whole sentences rather than quadruples, the complexity of the parse tree increases. Therefore, it is necessary to conduct in-depth exploration of various situations of parse trees. In addition to counting the proportions of various parsing, the tree structures that can judge the attachment besides the ground-truth parse trees will also be explored.



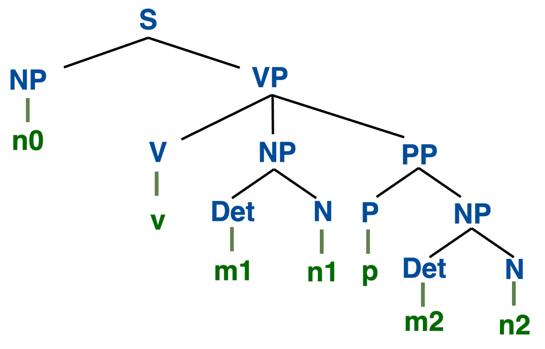
(a) Noun Attachment (binary)



(b) Verb Attachment (binary)



(c) Noun Attachment (ternary)



(d) Verb Attachment (ternary)

Figure 8: Two binary ground-truth parse trees where the prepositional phrase attaches to the noun (a) or to the verb (b), and two ternary ground-truth parse trees where the prepositional phrase attaches to the noun (c) or to the verb (d).

- **Accuracy:** Tree Transformer and pre-trained BiLSTM models are used to parse the processed PP attachment sentences and obtain the corresponding parse trees. Together with the ground-truth parse trees, the parsing accuracy of both models are calculated.
- **Assessment of Parsing Results:** in order to explore the factors that affect Tree Transformer's attachment decision of prepositional phrases, an in-depth observation was made on verbs and prepositions compared with the BiLSTM.

4.3.4 Measuring Garden Path Effects

Similar to the previous work by Futrell et al. (2019), we investigate the behavior of the Tree Transformer and to what extent it reflects incremental representations of syntactic states. For each garden path effect dataset, we compute the magnitude of its garden path effect. As mentioned earlier, this can be quantified by subtracting either sentences c from sentences a or sentences d from sentences b in terms of average surprisal at the disambiguators. Both differences were calculated and on datasets MV/RR as well as NP/Z (verb transitivity) they are compared with the garden path effects of LSTMs and RNNG in the study by Futrell et al. (2019).

To achieve this, we took the words before the disambiguator, along with the masked disambiguator, as input. In the case of the Tree Transformer, in addition to its self-attention output, we can also obtain a vector at the position of the disambiguator. By applying softmax to this vector, the probabilities of different words in the vocabulary at the masked position became known. Finding the corresponding probability of the disambiguating word allowed us to calculate the surprisal.

5 Results

In this section, all results of the experiments are presented and explained, including the test outcomes of the Tree Transformer and pre-trained BiLSTM. Additionally, their abilities in handling sentence-level prepositional phrase attachment ambiguity will be discussed, as well as the performance of the Tree Transformer in the context of garden path effects.

5.1 Performance on Test data

After we trained the Tree Transformer on the preprocessed WSJ-train, the F1 score calculated on the preprocessed WSJ-test is 49.7 as shown in the Table 4. As for the pre-trained BiLSTM, on the same test set, its F1 score is 75.8, which is higher than that of Tree Transformer. However, for the pretrained BiLSTM model, the F1 score deviates from the score reported by Gómez-Rodríguez and Vilares (2018) (90.6). This is due to few factors: first, they calculate the F1 score directly from the predicted labels, rather than the reconstructed span representation. Second, they use the micro- instead of macro-average F1 score. Third, the preprocessing of the data is different.

Model	F1-score
Tree Transformer, L=10	49.7
BiLSTM	75.8

Table 4: The F1 scores of Tree Transformer with 10 layers and pre-trained BiLSTM tested on WSJ-test.

5.2 Parsing Analysis on PP Attachment Sentences

5.2.1 Analysis on Parsing Structure

The parsing of a 7-tuple is more difficult than that of a quadruple, because the parse tree of a 7-tuple has more possibilities. Therefore, when analyzing the parsing capabilities of the model, it is necessary to consider the problem of parsing structure errors. We conduct separate analyses of the parsing results for the Tree Transformer and BiLSTM. While theoretically, the diversity of parsing results should be lower than the former as it generates only binary trees, the reality proves otherwise.

Parsing Structure	Number	Proportion
wrong parsing because of sentences	3	0.21%
prepositional phrases not found	278	19.52%
subjects not found	3	0.21%
$n1$ not in verb phrase	20	1.40%
fail to fully break down	14	0.98%
totally correct parsing	1120	78.65%

Table 5: Different parsing structures with their proportions of data for Tree Transformer

In terms of Tree Transformer, as shown in the Table 5, parsing errors resulting from sentence errors themselves amount to 3, which accounts for only 0.21% of the total. Parsing issues stemming from the model can manifest in various ways. One is structural errors and the other is the incomplete breakdown. The former consists of three kinds of structural errors. In 278 instances, parsing errors occur due to the failure to correctly group the prepositional phrase as a constituent, making up 19.52% of the total. Similarly, the model fails to correctly identify the subject of the sentence, such as merging the subject and the verb into the same constituent in some cases. This kind of error occurs in 3 instances, accounting for 0.21% of the total. Furthermore, there are 20 cases (1.40%) in which the noun $n1$ is not included as part of the verb phrase during parsing. It is worth noting that a sentence may exhibit multiple structural errors.

Regarding parsing results that are not fully broken down, there are four scenarios with a total of 14 examples (0.98%), as illustrated in the Figure 9. 13 cases of them are from Figures 9(a) and 9(b) depicting prepositional phrases that are not fully decomposed but still allow the attachment of the prepositional phrase to be determined. Figure 9(a) represents prepositional phrase attachment to the noun, while Figure 9(b) shows prepositional phrase attachment to the verb. Figures 9(c) and 9(d) represent cases where the verb phrase fails to fully break down, making it impossible to determine the attachment of the prepositional phrase. The remaining 1120 examples are structurally correct and completely parsed, constituting 78.65% of the total.

Hence, from the parsing results, it can be concluded that, in addition to the two ground-truth parse trees provided earlier, both Figures 9(a) and 9(b) can serve as true parse trees for the model to determine prepositional phrase attachment. Consequently, when calculating accuracy in subsequent experiments, all these four parse trees will all be taken into account for models. For BiLSTM, there are fewer structural errors in the parsing results, especially errors caused by the inability to find pp account for only 2 instances in total, significantly lower than Tree Transformer. Among the partially decomposed parsing trees, two types are observed, both involving cases where prepositional phrases are not fully broken down. Figure 10(a) illustrates a tree structure where the attachment is confidently determined to be a verb but remains partially decomposed. And Figure 10(b) depicts a tree structure where the attachment is confidently determined to be a noun, yet the decomposition remains incomplete. Both of these two tree structures do not affect the model's attachment decisions. Therefore, they are considered as the ground-truth trees for BiLSTM.

5.2.2 Assessment of Parsing Results

The Tree Transformer model achieved a parsing accuracy of 47.2% for sentences with prepositional phrase attachments. In comparison, the pre-trained BiLSTM model showed a much better performance with an accuracy of 79.4%, as presented in Table 6. This can be attributed to its supervised learning approach, where the high cost of training leads to substantial returns.

Model	**Accuracy**
Tree Transformer	47.2%
BiLSTM	79.4%

Table 6: Accuracy on pp attachment sentences for Tree Transformer and BiLSTM models.

Then, when deeply exploring the attachment decision tendency of Tree Transformer, the fol-

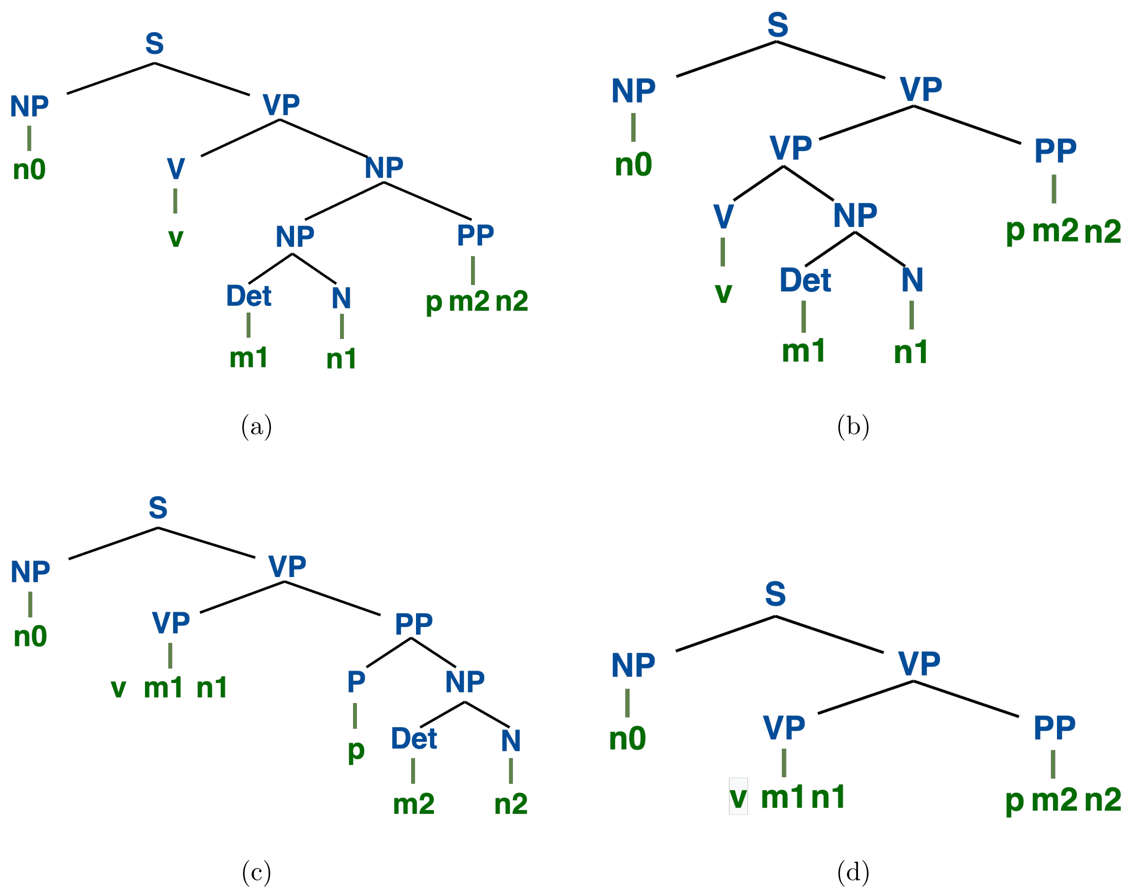


Figure 9: The parse trees failing to fully break down by Tree Transformer.

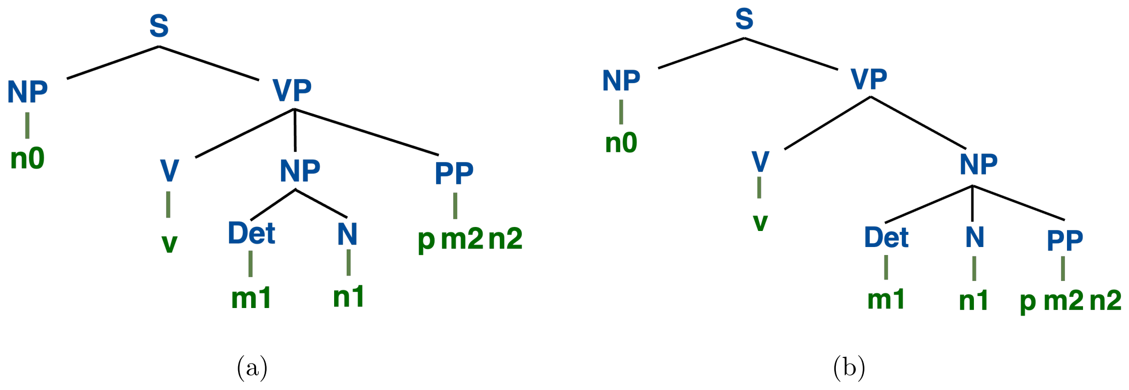


Figure 10: The parse trees failing to fully break down by BiLSTM.

lowing results were obtained. As shown in Table 7, while the number of prepositions attached to nouns (868) is higher than that attached to verbs (556) in the data, the Tree Transformer tends to attach pp to verbs (656) rather than nouns (477). Its ability to determine the attachment of pp to nouns is particularly weak, and there is a significant gap compared to the actual attachments. Additionally, the number of attachments where the attachment decision is neither a noun nor a verb is not negligible, contributing significantly to the low accuracy. This also indicates that the Tree Transformer is not highly sensitive to the attachment of prepositional phrases.

	Predicted_V	Predicted_N	Neither	Sum
Label_V	299	104	153	556
Label_N	357	373	138	868
Sum	656	477	291	1424

Table 7: The summary of attachment decision results on pp attachment sentences for Tree Transformer in the form of a confusion matrix.

However, while BiLSTM also tends to attach pp to verbs, this tendency is much less pronounced compared to the Tree Transformer. As shown in Table 8, the number of instances where BiLSTM attaches pp to verbs is 613, which is higher than the actual attachment count of 556 in the dataset. Conversely, BiLSTM attaches pp to nouns 798 times, which is lower than the actual attachment count of 868. Furthermore, the parsing results from BiLSTM provide a clearer reflection of the attachment situations for pp. The number of attachments that are neither nouns nor verbs is only 13, significantly lower than 291 that the Tree Transformer possesses.

	Predicted_V	Predicted_N	Neither	Sum
Label_V	442	110	4	556
Label_N	171	688	9	868
Sum	613	798	13	1424

Table 8: The summary of attachment decision results on pp attachment sentences for BiLSTM.

To investigate this imbalance in attachment decisions, we conducted an analysis of the parsing results combined with the distribution of verbs and prepositions in the sentences. This analysis aims to explore whether this imbalance is associated with specific verbs or prepositions.

Figure 11 illustrates the distribution of attachment decisions based on prepositions in both the data and the models. More specifically, Figure 11(a) shows the proportions of noun and verb attachments for top 20 frequent prepositions in data, among them, the proportion of the prepositions "of", "about", "between" and "than" attached to nouns is obviously higher than that of verbs. Especially the preposition "of", almost all. Other prepositions appear with a high proportion of verb attachment to varying degrees. Figure 11(b) presents the proportions of noun and verb attachments, as well as neither of them for the top 20 frequent verbs predicted by Tree Transformer. It can be clearly seen from the figure that Tree Transformer shows a proportion trend that is roughly consistent with the data, but the overall proportion shifts towards verb attachment. Only the preposition "than" is left, the proportion of noun attachment reaches 50%. As for the preposition "over", prepositional phrases almost never attach to nouns.

As for Figure 11(c), it shows that the parsing results of BiLSTM are closer to the distribution of the original dataset in terms of different prepositions. However, for certain prepositions, the distribution trend becomes more polarized. For phrases containing the prepositions "of", "for", "about", "between" and "than", BiLSTM is more likely than the dataset to classify them as attached to a noun. On the other hand, for other phrases, BiLSTM always has a stronger tendency of verb attachment than the data set. Especially phrases with "during" and "through", BiLSTM almost consistently attaches them to verbs. Therefore, BiLSTM exhibits a more pronounced attachment tendency difference in the prepositional dimension compared to Tree Transformer.

In addition, Figure 12 shows, for sentences containing different verbs, the proportion of prepositions attached to nouns and verbs in the data or the models. In detail, Figure 12(a) presents the proportions of noun and verb attachments for the top 20 frequent verbs in data. As can be seen from the figure, the proportion of prepositions attached to nouns is generally higher than to the forms of the verb "be" itself. In sentences containing other verbs, preposition attachments are evenly distributed. Figure 12(b) is the proportions of noun and verb attachments as well as neither of them for top 20 frequent verbs predicted by Tree Transformer. It can be found that the results are very different compared to that in the data. For sentences where the verbs are the "be" or "have" and their variations, Tree Transformer tends to attach PPs to the noun. For other notional verbs, however, Tree Transformer tends to attach the PPs to the verb.

This trend is less pronounced for BiLSTM. As shown in Figure 12(c), although sentences containing verbs like "be" or "have" and their variations tend to attach PP to nouns, many other verbs such as "provide" and "get" show a similar tendency but differ from the distribution in the data.

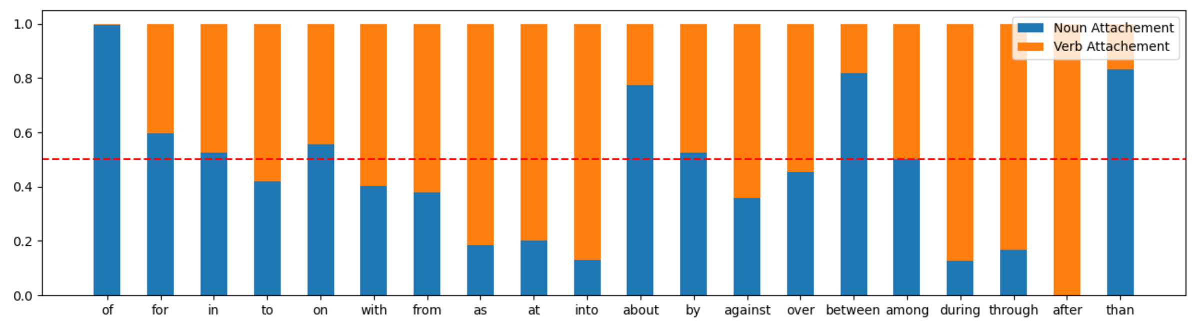
Therefore, in addition to calculating evaluation matrices for the whole data regarding predictions of verb or noun, the data of be/have verbs with their variations, and notional verbs are also aggregated to calculate their corresponding evaluation matrices. The results are given in Figure 13. To be more specific, it can be seen from Figure 13(a) that for the attachment prediction result of noun, Tree Transformer's parsing ability for sentences containing be/have series verbs is higher than the overall level, especially the recall. And its parsing ability for sentences containing notional verbs is slightly lower than the average level. On the contrary, for the prediction result as verb, Tree Transformer's performance on sentences containing be/have series verbs is much lower than the overall level, while its performance on sentences containing notional verbs is slightly above average. Therefore, Tree Transformer not only tends to attach PP with be/have verbs to nouns but also exhibits higher parsing ability.

Regarding BiLSTM, as shown in Figure 13(b), when predicted as nouns, the model's parsing performance on pp containing be/have is only slightly higher than the overall level. However, when predicted as verbs, it is significantly lower than the general level.

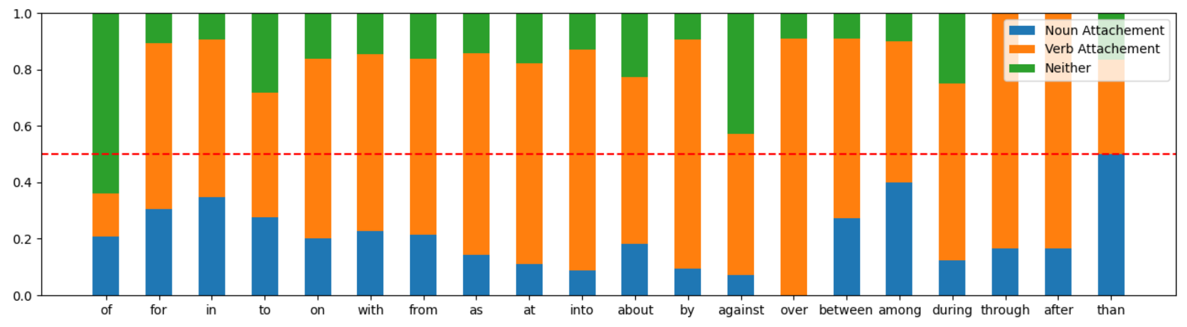
5.3 Parsing Analysis on the Garden Path Effect

In this part, we will combine Figure 14 and 15 to give a comparative analysis of the performance of each model in the garden path effect.

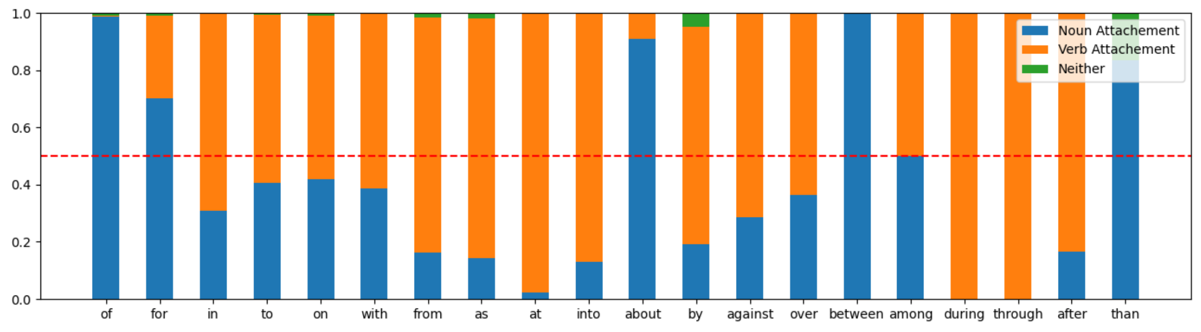
Figure 14(a) depicts the magnitude of the garden path effect generated by Tree Transformer and verb form ambiguity. Figure 15(a) illustrates the garden path effect size by four models and verb form ambiguity directly taken from Futrell et al. (2019). Upon comparison, it is evident that Tree Transformer, like the other models, exhibits a fundamental garden path effect.



(a) Data

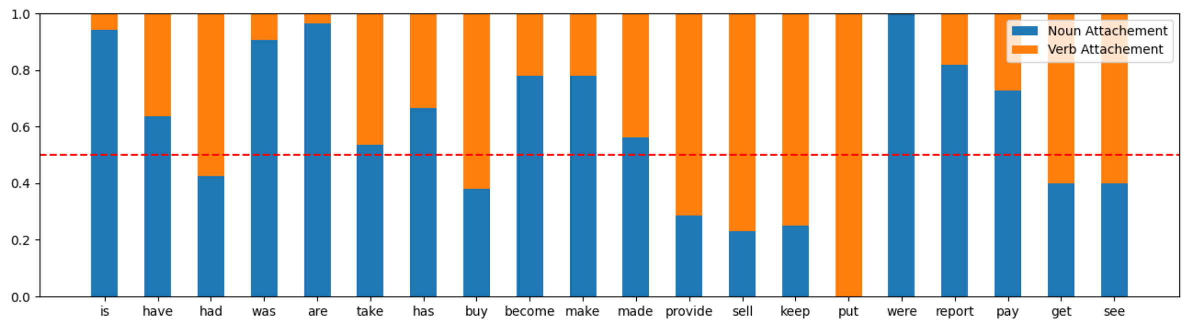


(b) Tree Transformer

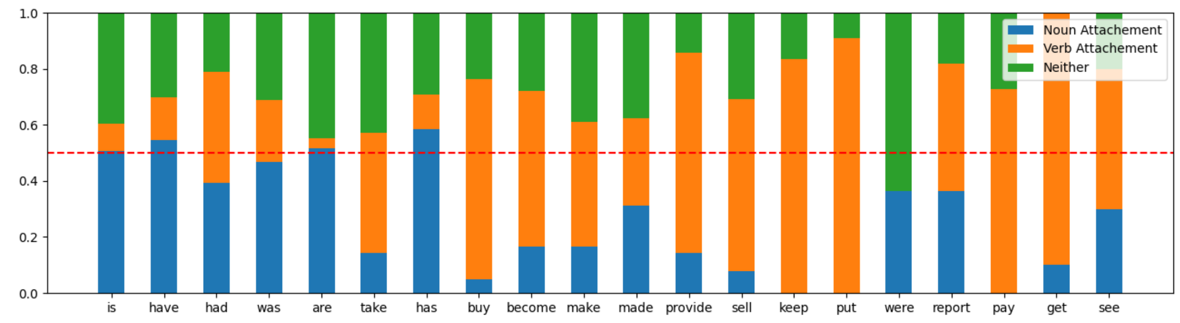


(c) BiLSTM

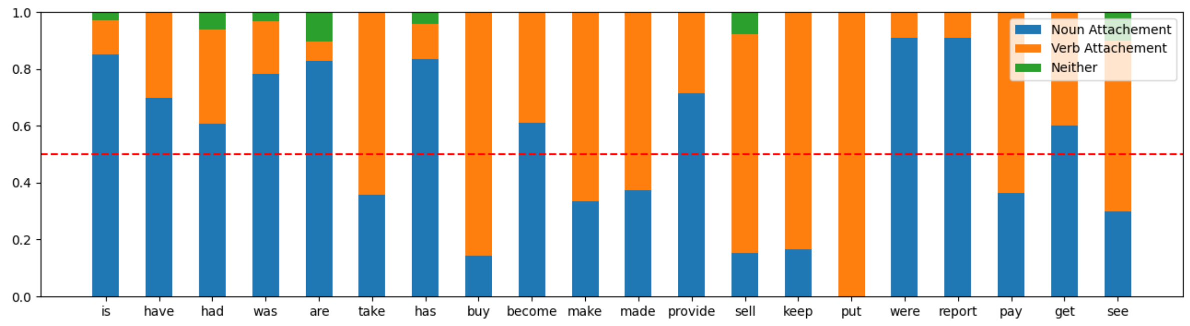
Figure 11: The proportions of noun attachment, verb attachments or neither/incorrect for the 20 most frequent *prepositions* in our dataset (a), by Tree Transformer (b) and BiLSTM (c), ordered by overall frequency.



(a) Data

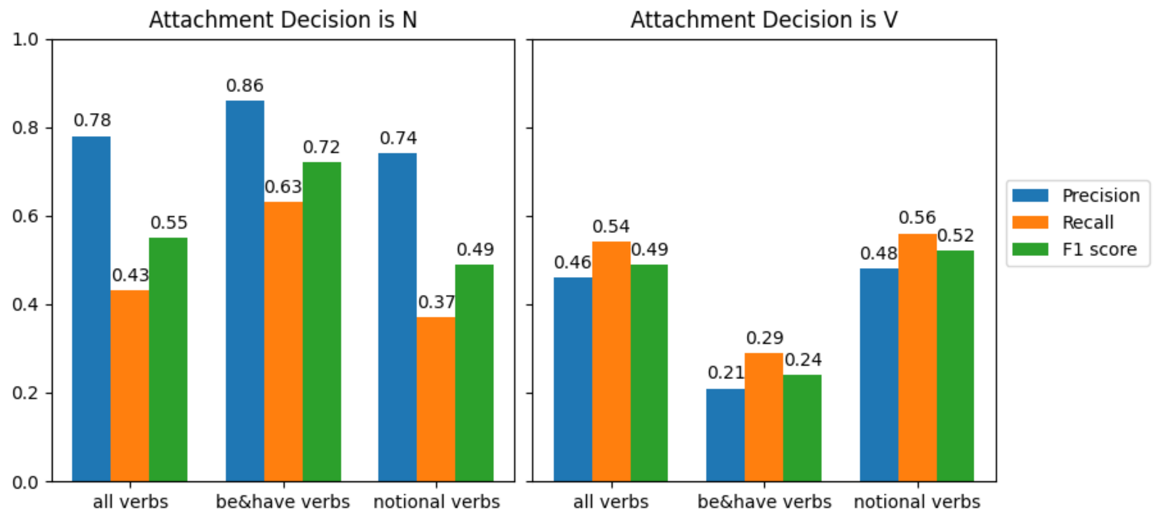


(b) Tree Transformer

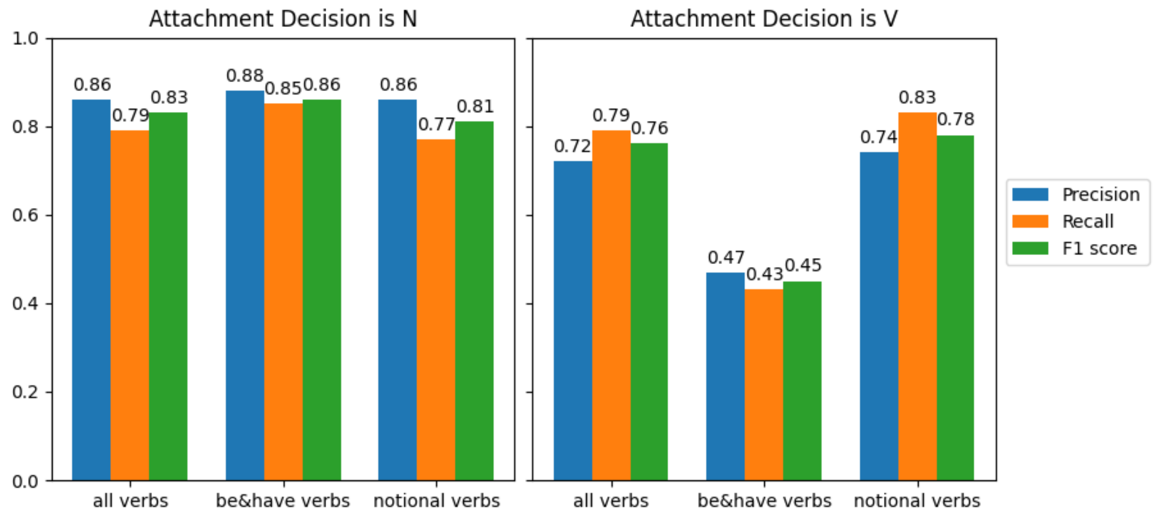


(c) BiLSTM

Figure 12: The proportions of noun attachment, verb attachments or neither/incorrect for the 20 most frequent *verbs* in our dataset (a), by Tree Transformer (b) and BiLSTM (c), ordered by overall frequency.



(a) Tree Transformer

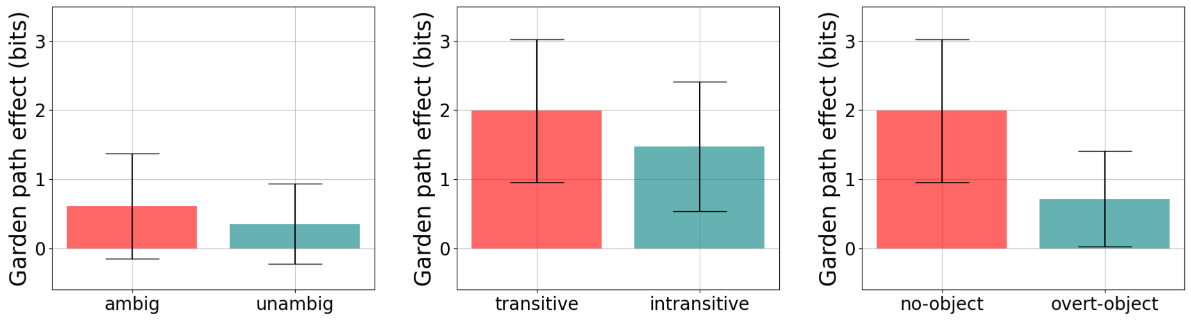


(b) BiLSTM

Figure 13: Evaluation metrics (precision, recall and F1 score) for different types of verbs when attachment decision is noun (N) or verb (V).

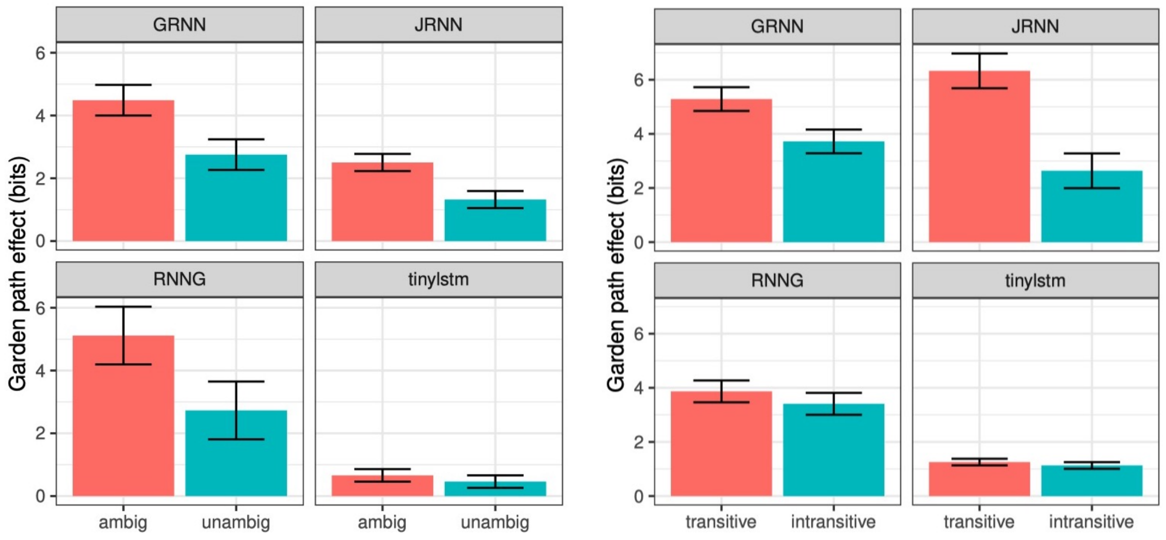
The garden path effect in Tree Transformer is similar to that of TinyLSTM, but considerably smaller than in the other models.

Additionally, if the model uses the morphological form of the verb as a cue for syntactic structure, instances where the verb has not changed to a passive participle form should exhibit a stronger garden path effect compared to situations where the change has already occurred. This is evident in the figures, where the red bars are higher than the green ones. Tree Transformer, along with two large LSTMs and RNNG, displays this pattern. Despite demonstrating crucial human-like garden path effect disambiguation due to the verb form ambiguity, it is noteworthy that significant garden path effect still persist in these models, even when the verb form is unambiguous like passive-participial verb. Regarding TinyLSTM, it does not exhibit sensitivity to ambiguous verb forms and reduced relative clauses.



(a) MV/RR (b) NP/Z(verb transitivity) (c) NP/Z(overt object)

Figure 14: Average garden path effect size by Tree Transformer and disambiguation lexical clues on (a) MV/RR dataset; (b) NP/Z (verb transitivity) dataset; (c) NP/Z (overt object) dataset. Error bars depict 95% confidence intervals computed based on the standard error of the surprisals after subtracting out the mean surprisal (Masson and Loftus, 2003).



(a) MV/RR (b) NP/Z(verb transitivity)

Figure 15: Figures are the original ones taken from the paper of Futrell et al. (2019). Average garden path effect by 4 models and disambiguation lexical clues on (a) MV/RR dataset; (b) NP/Z (verb transitivity) dataset.

Figures 14(b) and 15(b) respectively illustrate the garden path effect sizes generated by Tree Transformer and four other models along with those of verb transitivity. Similarly, we observe the presence of the garden path effect in all models. Although smaller in Tree Transformer, even markedly smaller than in the two large LSTMs, it is higher than in TinyLSTM. As for the sensitivity, only the large LSTMs seem to be sensitive to the transitivity of embedded verbs, showing smaller garden path effects for intransitive verbs. Tree Transformer demonstrates sensitivity just below them, but noticeably higher than RNNG and TinyLSTM.

Figure 14(c) shows the garden path effect sizes generated by Tree Transformer and presence of an object. In comparison with Figure 14(b), it can be observed that Tree Transformer is much more sensitive to the presence or absence of an object than lexical cues such as verb transitivity.

6 Discussion

Tree Transformer, as an unsupervised parser, has been confirmed to be effective in deriving tree structures from raw text that capture human intuitions. The inherent ambiguity in human natural language is a persistent challenge that language models have been striving to overcome. Therefore, to further explore the performance of this low-cost unsupervised parser on syntactic ambiguities, our work primarily investigate two main questions: How well does Tree Transformer handle PP attachment ambiguities? What is its ability to predict the garden path effects? Around these questions, we obtained the following respective answers and made additional contributions.

Our exploration of Tree Transformer in prepositional phrase ambiguity is on the sentence level, which distinguishes it from other works. To achieve this, we use BERT to generate natural sentences from the prepositional phrase attachment corpus, making this research endeavor possible, which is also a significant contribution from our work. In addition, we trained the Tree Transformer from scratch and conducted tests on it with our novel data, comparing it with a pre-trained BiLSTM. Then, new findings have emerged. The results indicate that it is not a robust model for handling PP attachment ambiguity. Compared with supervised learning BiLSTM, there is a significant accuracy gap on pp attachment sentences, largely attributed to the inability to recognize prepositional phrases, making it challenging to determine attachment decisions. Our analysis of structural errors in Tree Transformer revealed that for 19.52% of the sentences, it failed to identify prepositional phrases. This aligns with the finding from Wang et al. (2019), where Tree Transformer's recall for PPs on WSJ dataset was 52.3%. In contrast, BiLSTM benefits from supervised training with POS tags. For such simple 7-tuple sentences, it can more effectively detect prepositional phrases, leading to a reduction in errors. In addition, Tree Transformer generally tends to attach decisions to verbs, but for sentences with be/have verbs and their variants, Tree Transformer lean towards attaching to nouns and a higher prediction performance.

In exploring Tree Transformer's performance in the garden path effect caused by MV/RR and NP/Z ambiguities, we found that Tree Transformer exhibits the garden path effect similar to those of two large LSTMs, RNNG and TinyLSTM. This effect is modulated by different lexical cues such as verb form, verb transitivity and the presence of an object. However, Tree Transformer is not as strong as two large LSTMs in fully capturing the effect of incremental state and the potential subtle lexical cues that cause the state to change. This aligns with previous conclusions from Futrell et al. (2019) that utilizing fine-grained lexical cues for syntactic structure representation requires large data. Specifically, Tree Transformer's sensitivity falls between the two large LSTMs and TinyLSTM. Compared to RNNG, Tree Transformer has lower sensitivity to verb form but higher sensitivity to verb transitivity. Additionally, in eliminating the garden path effect caused by NP/Z ambiguity, Tree Transformer performs better when an overt object is added compared to changing the verb to intransitive. However, it is worth noting that the current datasets for the garden path effect are relatively small, comprising only around 30 examples of each. This introduces the possibility of errors when comparing models with the same training set size.

7 Conclusion

Due to the low cost of unsupervised parsers, their research is crucial. One such parser is the Tree Transformer. Our study reveals that the Tree Transformer performs poorly in disambiguating PP attachments at the sentence level, far less effective than BiLSTM, primarily due to its weak ability to identify prepositional phrases. Furthermore, the Tree Transformer demonstrates garden path effects across multiple datasets and exhibits varying sensitivity to different subtle lexical cues.

However, there is still room for improvement, and many related aspects warrant further investigation. For example, enriching the diversity and quantity of pp attachment sentences can enhance the accuracy of results. Improving Tree Transformer's ability to correctly identify prepositional phrase structures is essential for increased accuracy. Additionally, training Tree Transformer on larger datasets could be attempted to explore differences in performance of garden path effect compared to large LSTMs. Furthermore, we observed that Tree Transformer demonstrates stronger sensitivity to subtle lexical cues compared to TinyLSTM trained with the same training set. In the future, further comparisons could be conducted on larger and more diverse datasets to provide a more robust evaluation.

References

Bever, T. G. (1970). The cognitive basis for linguistic structures. *Cognition and the development of language*.

Brill, E. and Resnik, P. (1994). A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the 15th Conference on Computational Linguistics*, COLING '94, page 1198–1204.

Clark, A. (2001). Unsupervised induction of stochastic context-free grammars using distributional clustering. In *Proceedings of the ACL 2001 Workshop on Computational Natural Language Learning (ConLL)*.

Collins, M. and Brooks, J. (1995). Prepositional phrase attachment through a backed-off model. *ArXiv*, abs/cmp-lg/9506021.

Demberg, V. and Keller, F. (2008). Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, 109(2):193–210.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018a). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018b). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Drozdov, A., Verga, P., Yadav, M., Iyyer, M., and McCallum, A. (2019). Unsupervised latent tree induction with deep inside-outside recursive autoencoders. *arXiv preprint arXiv:1904.02142*.

Dyer, C., Kuncoro, A., Ballesteros, M., and Smith, N. A. (2016). Recurrent neural network grammars. In *North American Chapter of the Association for Computational Linguistics*.

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

Fellbaum, C. (1998). *WordNet: An electronic lexical database*. MIT press.

Ferreira, F. and Clifton Jr, C. (1986). The independence of syntactic processing. *Journal of memory and language*, 25(3):348–368.

Frank, S. L. and Bod, R. (2011). Insensitivity of the human sentence-processing system to hierarchical structure. *Psychological science*, 22(6):829–834.

Frazier, L. (1979). *On comprehending sentences: Syntactic parsing strategies.* University of Connecticut.

Frazier, L. and Fodor, J. D. (1978). The sausage machine: A new two-stage parsing model. *Cognition*, 6(4):291–325.

Futrell, R., Wilcox, E., Morita, T., Qian, P., Ballesteros, M., and Levy, R. (2019). Neural language models as psycholinguistic subjects: Representations of syntactic state. *arXiv preprint arXiv:1903.03260*.

Gómez-Rodríguez, C. and Vilares, D. (2018). Constituent parsing as sequence labeling. *arXiv preprint arXiv:1810.08994*.

Goodkind, A. and Bicknell, K. (2018). Predictive power of word surprisal for reading times is a linear function of language model quality. In *Proceedings of the 8th workshop on cognitive modeling and computational linguistics (CMCL 2018)*, pages 10–18.

Gulordava, K., Bojanowski, P., Grave, E., Linzen, T., and Baroni, M. (2018). Colorless green recurrent networks dream hierarchically. *arXiv preprint arXiv:1803.11138*.

Hale, J. (2001). A probabilistic earley parser as a psycholinguistic model. In *Second meeting of the north american chapter of the association for computational linguistics*.

Hindle, D. and Rooth, M. (1993). Structural ambiguity and lexical relations. *Computational linguistics*, 19(1):103–120.

Kim, Y., Dyer, C., and Rush, A. M. (2019a). Compound probabilistic context-free grammars for grammar induction. *arXiv preprint arXiv:1906.10225*.

Kim, Y., Rush, A. M., Yu, L., Kuncoro, A., Dyer, C., and Melis, G. (2019b). Unsupervised recurrent neural network grammars. *arXiv preprint arXiv:1904.03746*.

Kimball, J. (1973). Seven principles of surface structure parsing in natural language. *Cognition*, 2(1):15–47.

Klein, D. and Manning, C. D. (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.

Lau, J. H., Clark, A., and Lappin, S. (2017). Grammaticality, acceptability, and probability: A probabilistic view of linguistic knowledge. *Cognitive science*, 41(5):1202–1241.

Levy, R. (2008). Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

Linzen, T., Dupoux, E., and Goldberg, Y. (2016). Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Liu, N. F., Gardner, M., Belinkov, Y., Peters, M. E., and Smith, N. A. (2019). Linguistic knowledge and transferability of contextual representations. *arXiv preprint arXiv:1903.08855*.

MacDonald, M. C. and Christiansen, M. H. (2002). Reassessing working memory: comment on just and carpenter (1992) and waters and caplan (1996).

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Annalen der Physik*, 19(2):313–330.

Masson, M. E. and Loftus, G. R. (2003). Using confidence intervals for graphically based data interpretation. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, 57(3):203.

Pantel, P. and Lin, D. (2000). An unsupervised approach to prepositional phrase attachment using contextually similar words. In *Proceedings of the 38th annual meeting of the Association for Computational Linguistics*, pages 101–108.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Ratnaparkhi, A., Reynar, J., and Roukos, S. (1994). A maximum entropy model for prepositional phrase attachment. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, page 250–255.

Shen, Y., Lin, Z., Huang, C.-W., and Courville, A. (2018a). Neural language modeling by jointly learning syntax and lexicon. *arXiv preprint arXiv:1711.02013*.

Shen, Y., Tan, S., Sordoni, A., and Courville, A. (2018b). Ordered neurons: Integrating tree structures into recurrent neural networks. *arXiv preprint arXiv:1810.09536*.

Smith, N. A. and Eisner, J. (2005). Guiding unsupervised grammar induction using contrastive estimation. In *Proc. of IJCAI Workshop on Grammatical Inference Applications*, pages 73–82.

Stetina, J. and Nagao, M. (1997). Corpus based pp attachment ambiguity resolution with a semantic dictionary. In *Fifth Workshop on Very Large Corpora*.

Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Taraban, R. and McClelland, J. L. (1988). Constituent attachment and thematic role assignment in sentence processing: Influences of content-based expectations. *Journal of memory and language*, 27(6):597–632.

Toutanova, K., Manning, C. D., and Ng, A. Y. (2004). Learning random walk models for inducing word dependency distributions. In *Proceedings of the twenty-first international conference on Machine learning*, page 103.

Trueswell, J. C., Tanenhaus, M. K., and Garnsey, S. M. (1994). Semantic influences on parsing: Use of thematic role information in syntactic ambiguity resolution. *Journal of memory and language*, 33(3):285–318.

Van Schijndel, M. and Linzen, T. (2018a). Modeling garden path effects without explicit hierarchical syntax. In *CogSci*.

Van Schijndel, M. and Linzen, T. (2018b). A neural model of adaptation in reading. *arXiv preprint arXiv:1808.09930*.

Van Schijndel, M. and Linzen, T. (2021). Single-stage prediction models do not explain the magnitude of syntactic disambiguation difficulty. *Cognitive science*, 45(6):e12988.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, Y.-S., Lee, H.-Y., and Chen, Y.-N. (2019). Tree transformer: Integrating tree structures into self-attention. *arXiv preprint arXiv:1909.06639*.

Whittemore, G., Ferraro, K., and Brunner, H. (1990). Empirical study of predictive powers of simple attachment schemes for post-modifier prepositional phrases. In *28th Annual Meeting of the Association for Computational Linguistics*, pages 23–30.

Wu, W., Wang, H., Liu, T., and Ma, S. (2018). Phrase-level self-attention networks for universal sentence encoding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3729–3738.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yang, J. and Zhang, Y. (2018). Ncrf++: An open-source neural sequence labeling toolkit. *arXiv preprint arXiv:1806.05626*.

Yogatama, D., Blunsom, P., Dyer, C., Grefenstette, E., and Ling, W. (2016). Learning to compose words into sentences with reinforcement learning. *arXiv preprint arXiv:1611.09100*.