On Eliciting Syntax from Language Models via Hashing

Yiran Wang **Masao Utiyama**
National Institute of Information and Communications Technology (NICT)
yiran.wang@nict.go.jp mutiyama@nict.go.jp

Abstract

Unsupervised parsing, also known as grammar induction, aims to infer syntactic structure from raw text. Recently, binary representation has exhibited remarkable information-preserving capabilities at both lexicon and syntax levels. In this paper, we explore the possibility of leveraging this capability to deduce parsing trees from raw text, relying solely on the implicitly induced grammars within models. To achieve this, we upgrade the bit-level CKY from zero-order to first-order to encode the lexicon and syntax in a unified binary representation space, switch training from supervised to unsupervised under the contrastive hashing framework, and introduce a novel loss function to impose stronger yet balanced alignment signals. Our model[1] shows competitive performance on various datasets, therefore, we claim that our method is effective and efficient enough to acquire high-quality parsing trees from pre-trained language models at a low cost.

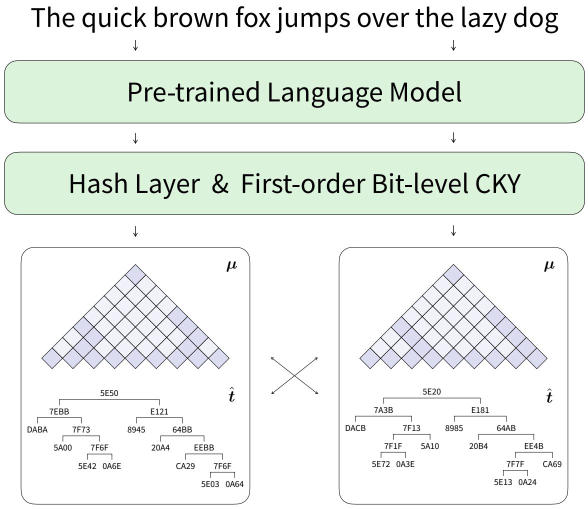

Figure 1: The model architecture. The hash layer produces scores of all spans, and the following first-order bit-level CKY (§3.1) returns marginal probabilities $\boldsymbol{\mu}$ and predicts the most probable trees $\hat{\boldsymbol{t}}$. Sentences are fed into the network twice, We select span marginal probabilities from one pass according to the predicted trees from the other pass, and perform contrastive hashing (§3.2, §3.3) on their corresponding score and code vectors. The purple cells represent the marginal probabilities, and the dark purple indicate the selected ones.

1 Introduction

Grammars form the backbone of languages, providing the essential framework that dictates how lexicons are arranged to convey meaning. Understanding and generating language heavily relies on grasping these latent structures. Unsupervised parsing, which aims to deduce sentence structure without relying on costly manually annotated treebanks, has been widely studied in academia. However, despite its importance, advancements have been slow due to the intrinsic complexity of this task. Nowadays, addressing these challenges becomes even more crucial for further exploring the capabilities of large language models.

Word embedding and language model techniques (Mikolov et al., 2013a,b; Radford, 2018; Devlin et al., 2019) have shown that training models to predict tokens in specific contexts is remarkably effective in implicitly capturing lexical features. A well-known example is the captured lexical relationship of *king* - *man* + *woman* = *queen*. As one of the most widely accepted explanations for this phenomenon, the distributional hypothesis (Harris, 1954; Mikolov et al., 2013a,b) suggests this is because tokens appearing in similar contexts tend to be assigned similar meanings. Specifically, similar contexts achieve this by placing tokens in analogous syntactic structures. This phenomenon naturally prompts us to consider whether there is a representation learning method that can explicitly encode both lexical and syntactic information in a unified format, making it possible to capture syntactic structures as well as lexical relationships by training language models solely with conventional

[1] https://github.com/speedcell4/parserker

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 8412–8427
November 12-16, 2024

conditional token prediction procedures.

Fortunately, the recently proposed binary representation meets these requirements perfectly. Wang et al. (2023) proposed a binary representation that bridges the gap between the continuous nature of deep learning and the discrete intrinsic property of natural languages. Instead of directly applying contrastive learning on the high-dimensional continuous hidden states of pre-trained language models, Wang et al. (2023) project them as K-dimensional score vectors. These scores can easily be binarized into K-bit codes, and token-level contrastive learning is applied among these scores and their binarized codes. They demonstrate that lexical information can be properly preserved within only 24 bits. Following this, Wang and Utiyama (2024) additionally take spans on the target parsing trees into consideration. They use marginal probabilities to construct a novel similarity function that reflects not only lexical information but also the boundary of each span, and then perform contrastive hashing across spans rather than tokens. In their supervised parsing experiments, they show the effectiveness of the structured binary representation by achieving comparable performance to conventional parsers.

Although in the supervised settings, Wang and Utiyama (2024) achieves satisfactory results, we found that for unsupervised settings, their model is insufficient to induce meaningful parsing trees. In this paper, we aim to elicit constituency parsers from pre-trained language models without training them on annotated treebanks. We analyze the existing issues of their structured binary representation and explore the possibility of further enhancing the unified information-preserving capability. To achieve this, we upgrade the bit-level CKY module from zero-order (§2.1) to first-order (§3.1) to integrate lexicon and syntax in a unified format, convert parsing from supervised (§2.2) to unsupervised (§3.2), and propose a novel objective function (§3.3) to impose stronger yet balanced alignment signals. Besides, we also discuss how the learning objective of contrastive hashing aligns with the target of parsing. This provides an explanation (§3.2) different from the distributional hypothesis and explains why our training leads to syntactic structures rather than other structures. Experiments show that our models achieve competitive performance and indicate that acquiring high-quality syntactic annotations at a low cost is becoming practicable. We refer to our parser as `Parserker2`, following the original `Parserker` (Wang and Utiyama, 2024).

2 Background

2.1 Zero-order Constituency Parsing

Given sequence $w_1, \ldots, w_n$, constituency parser returns the most probable binary-branching parsing tree $\boldsymbol{t} = \{\langle l_i, r_i, y_i \rangle\}_{i=1}^{2n-1}$, which is represented as a list of labeled spans indicating constituents at different hierarchies. Where l_i and r_i refer to the left and right boundaries of the i-th constituent, and $y_i \in \mathcal{Y}$ stands for its assigned label. Previous models (Kitaev and Klein, 2018; Yu et al., 2020; Zhang et al., 2020) commonly employ encoders to transform inputs into hidden states $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n$ first, use classifiers to predict span scores $g(l, r, y)$ and tree scores $g(\boldsymbol{t})$, and then normalize them among all valid trees to obtain tree probability $p(\boldsymbol{t})$. Training and inference stages aim at maximizing the probabilities of target trees and searching trees with the maximal probabilities $\hat{\boldsymbol{t}}$, respectively.

$$g(\boldsymbol{t}) = \sum_{\langle l,r,y \rangle \in \boldsymbol{t}} g(l, r, y) \tag{1}$$

$$p(\boldsymbol{t}) = \frac{\exp g(\boldsymbol{t})}{Z \equiv \sum_{\boldsymbol{t}' \in \mathcal{T}} \exp g(\boldsymbol{t}')} \tag{2}$$

$$\hat{\boldsymbol{t}} = \{\langle l_i, r_i, y_i \rangle\}_{i=1}^{2n-1} \leftarrow \arg\max_{\boldsymbol{t} \in \mathcal{T}} \; p(\boldsymbol{t}) \tag{3}$$

Apart from being used to normalize probabilities of trees, the log partition term Z, which stands for the total scores of all valid constituency trees, can also be used to compute span marginal probabilities. As Eisner (2016) mentioned, computing the partial derivative of the log partition with respect to span scores yields marginal probabilities efficiently.

$$\mu(l, r, y) = \frac{\partial \log Z}{\partial \, g(l, r, y)} \tag{4}$$

Intuitively speaking, marginal probability reflects the joint probability of selecting tokens $w_l, \ldots, w_r$ as a constituent with label y assigned to it. If a span is not likely to be selected, its marginal probability will not be high regardless of its label. Therefore, similar to hidden states, marginal probabilities are considered a format containing not only lexical but also syntactic features. Unlike hidden states, these marginal probabilities explicitly correspond to the specific boundaries and labels of spans in parsing trees globally normalized under the CKY framework, whereas hidden states implicitly preserve this information in a high-dimensional, human-unreadable format.

Recently, Wang and Utiyama (2024) extended constituency parsers by replacing discrete labels $y \in \mathcal{Y}$ with binary codes $\boldsymbol{c} \in \{-1, +1\}^K$. In their approach, the code-level scores $g(l, r, \boldsymbol{c})$ are obtained by summing up bit-level scores $g_k(l, r, c^k)$.

$$g(\boldsymbol{t}) = \sum_{\langle l, r, \boldsymbol{c} \rangle \in \boldsymbol{t}} g(l, r, \boldsymbol{c}) \tag{5}$$

$$g(l, r, \boldsymbol{c}) = \sum_{k=1}^{K} g_k(l, r, c^k) \tag{6}$$

Moreover, to compute these bit-level scores, they retained the one-head-one-bit design of Wang et al. (2023) and employed a multi-head attention module to predict the score of being assigned $+1$.

$$g_k(l, r, +1) = \frac{(\mathbf{W}_k^Q \boldsymbol{h}_l)^\top (\mathbf{W}_k^K \boldsymbol{h}_r)}{\sqrt{d_k}} \tag{7}$$

$$g_k(l, r, -1) = 0 \tag{8}$$

Where $\mathbf{W}_k^Q, \mathbf{W}_k^K \in \mathbb{R}^{\lceil \frac{d}{K} \rceil \times d}$ are the query and key matrices used to produce the k-th bit. They assign a score of 0 for the -1 case and extend the marginal probability and decoding to the bit level.

$$\mu_k(l, r, c^k) = \frac{\partial \log Z}{\partial g_k(l, r, c^k)} \tag{9}$$

$$\hat{\boldsymbol{t}} = \{\langle l_i, r_i, \boldsymbol{c}_i \rangle\}_{i=1}^{2n-1} \leftarrow \arg\max_{\boldsymbol{t} \in \mathcal{T}} p(\boldsymbol{t}) \tag{10}$$

2.2 Supervised Contrastive Hashing

To perform contrastive learning, Wang et al. (2023) and Wang and Utiyama (2024) define their similarity functions in a similar manner, both first binarize one input and then calculate the similarity between the continuous one and the binarized one. However, the former binarizes scores via taking their signs, while the latter leans bits towards the sides with larger marginal probabilities.

$$\boldsymbol{c} = [c^1, \ldots, c^K] \in \{-1, +1\}^K \tag{11}$$

$$c^k = \begin{cases} +1 & \mu_k(l, r, +1) > \mu_k(l, r, -1) \\ -1 & \text{otherwise} \end{cases}$$

As mentioned above, marginal probabilities contain both label and structural information. To impose supervision on lexicon and syntax simultaneously by leveraging this property, they proposed defining the novel similarity as the average of bit-level marginal probabilities of the i-th constituent with the binary label of j-th constituent.

$$s(i, j) = \frac{1}{K} \sum_{k=1}^{K} \mu_k(l_i, r_i, c_j^k) \tag{12}$$

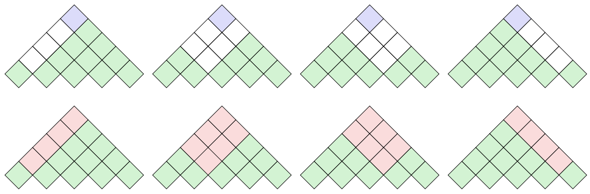

Figure 2: Charts of the zero-order (above §2.1) and the first-order parsing (below §3.1). At this time step, zero-order parsers separately determine the splitting positions on the left and right children and predict labels according to the top-most cell. In contrast, first-order parsers make these two decisions jointly by averaging all the cells that cross the left and right children to unify the representation of lexicon and syntax.

During the training stage, Wang and Utiyama (2024) select spans from target trees to perform contrastive hashing with the similarity function described above. Naively contrasting all spans would increase the time complexity to $\mathcal{O}(n^4)$. To avoid this, they restrict supervision to spans on the target trees, reducing the number of spans to $2n - 1$ and maintaining the time complexity at $\mathcal{O}(n^2)$. In their supervised settings, they only allow the model to determine the binary codes, without predicting the boundaries, thus, the procedure can be reinterpreted as searching in a constrained space.

$$\hat{\boldsymbol{t}} = \{\langle l_i, r_i, \hat{\boldsymbol{c}}_i \rangle\}_{i=1}^{2n-1} \leftarrow \arg\max_{\boldsymbol{t} \in \mathcal{T}[\boldsymbol{l}, \boldsymbol{r}, \cdot]} p(\boldsymbol{t}) \tag{13}$$

Where l_i and r_i denote the boundaries of the target spans, and $\mathcal{T}[\boldsymbol{l}, \boldsymbol{r}, \cdot]$ means only searching in the constrained space to ensure target are always included. Besides, the positive and negative sets are divided according to the ground-truth labels y_i.

$$\begin{aligned} \mathcal{P} &= \{j \mid y_i = y_j\} \\ \mathcal{N} &= \{j \mid y_i \neq y_j\} \end{aligned} \tag{14}$$

3 Proposed Methods

3.1 First-order Constituency Parsing

Efficient computing requires batchifying the inside pass of the CKY algorithm for parallel dynamic programming on GPUs (Stern et al., 2017; Zhang et al., 2020). Within the CKY framework, Wang and Utiyama (2024) introduce a large tensor as the chart for dynamic programming, where $G(l, r, \boldsymbol{c})$ refers to the total scores of all trees spanning from l to r with code $\boldsymbol{c}$ as the top label, while $g(l, r, \boldsymbol{c})$ stands for a single constituent. The algorithm starts from single-word spans and incrementally computes larger spans by enumerating splitting positions and summing children with the top span.

$$G(l,r,\boldsymbol{c}) \leftarrow \sum_{m=l}^{r-1} G(l,m,\cdot) + G(m+1,r,\cdot) + g(l,r,\boldsymbol{c}) \tag{15}$$

This procedure has been widely employed as a practical standard (Zhang et al., 2020; Wang et al., 2023). However, we notice that natively using it for unsupervised parsing is not sufficient. As shown in Figure 2, the crux is that even though Equation 15 enumerates all valid splitting positions, the span score $g(l,r,\boldsymbol{c})$ does not take the splitting positions into consideration. According to Equation 7, this score depends only on the leftmost and rightmost tokens, regardless of the chosen splitting positions. In other words, different choices of splitting positions do not vary the code scores of top spans. Therefore, performing contrastive hashing by using such scores barely provides any effective information for unsupervised parsing. We refer to this kind of CKY as zero-order CKY.

Naturally, the most intuitive solution is upgrading to first-order CKY by taking the splitting position m into consideration through introducing a novel span score function $g(l,r,m,\boldsymbol{c})$.

$$G(l,r,\boldsymbol{c}) = \sum_{m=l}^{r-1} G(l,m,\cdot) + G(m+1,r,\cdot) + g(l,r,m,\boldsymbol{c}) \tag{16}$$

And instead of relying only on the leftmost and the rightmost hidden states, we use the averaged representation of the left and right children, respectively.

$$g(l,r,m,\boldsymbol{c}) = \sum_{k=1}^{K} g_k(l,r,m,c^k) \tag{17}$$

$$g_k(l,r,m,+1) = \frac{(\mathbf{W}_k^Q \overline{\boldsymbol{h}}_{l:m})^\top (\mathbf{W}_k^K \overline{\boldsymbol{h}}_{m+1:r})}{\sqrt{d_k}}$$

$$\overline{\boldsymbol{h}}_{l:m} = \operatorname*{mean}_{l \le i \le m} \boldsymbol{h}_i$$

$$\overline{\boldsymbol{h}}_{m+1:r} = \operatorname*{mean}_{m < j \le r} \boldsymbol{h}_j$$

Where $\overline{\boldsymbol{h}}_{l:m}$ and $\overline{\boldsymbol{h}}_{m+1:r}$ are the averaged representation of the left and right children, respectively. In this way, the splitting position influences the scores of binary codes through children hidden states.

However, naively computing $g_k(l,r,m,+1)$ requires additional computational resources for averaging vectors and performing dot products in real-time, which heavily slows down training and inference. Fortunately, through simple derivation, we note that the new score can be obtained by merely averaging the old scores. Upgrading CKY from zero-order to first-order then introduces almost no additional delay by applying this trick.

$$g_k(l,r,m,+1) = \operatorname*{mean}_{l \le i \le m < j \le r} g_k(i,j,+1) \tag{18}$$

According to this definition, the new scores can be interpreted as being obtained by averaging the left and right children, respectively, and then calculating the scores for construing a span across them. Different choices of splitting positions result in different representations of the left and right children, leading to different bit scores for the top span. Since scores reflect the substructure of spans, aligning and uniformalizing these scores in Hamming space using contrastive learning is equivalent to aligning and uniformalizing the subtrees in syntactic structure space. Hence, our method can also be considered relevant to syntactic distance (Shen et al., 2018a, 2019). Additionally, we also assign a score of 0 for the -1 case.

$$g_k(l,r,m,-1) = 0 \tag{19}$$

And extend the marginal probabilities as well.

$$\mu_k(l,r,m,c^k) = \frac{\partial \log Z}{\partial g_k(l,r,m,c^k)} \tag{20}$$

3.2 Unsupervised Contrastive Hashing

We define our similarity in a manner similar to Wang and Utiyama (2024). As we have upgraded the bit-level CKY module from zero-order to first-order, we also upgrade the binarization procedure.

$$\boldsymbol{c} = [c^1, \ldots, c^K] \in \{-1,+1\}^K \tag{21}$$

$$c^k = \begin{cases} +1 & \mu_k(l,r,m,+1) > \mu_k(l,r,m,-1) \\ -1 & \text{otherwise} \end{cases}$$

and the similarity function as follows.

$$s(i,j) = \frac{1}{K} \sum_{k=1}^{K} \mu_k(l_i, r_i, m_i, c_j^k) \tag{22}$$

Unlike in the supervised settings of Wang and Utiyama (2024), we aim to obtain constituency parsers without training them on annotated treebanks, i.e., $\{\langle l_i, r_i, y_i \rangle\}_{i=1}^{2n-1}$. Therefore, it is difficult for us to constrain the search space as Equation 13 and to divide spans according to ground-truth labels as Equation 14. Thus, we unlock these restrictions and let parsers determine constituent boundaries and binary labels jointly through searching in an unconstrained space $\mathcal{T}[\cdot,\cdot,\cdot]$.

$$\hat{\boldsymbol{t}} = \left\{ \langle \hat{l}_i, \hat{r}_i, \hat{\boldsymbol{c}}_i \rangle \right\}_{i=1}^{2n-1} \leftarrow \underset{\boldsymbol{t} \in \mathcal{T}[\cdot,\cdot,\cdot]}{\arg\max}\ p(\boldsymbol{t}) \tag{23}$$

After that, since we neither have access to the ground-truth labels y_i, we turn to use the lexicons in spans $w_{\hat{l}_i}, \dots, w_{\hat{r}_i}$ as the labels to divide these selected spans. In this way, pulling or pushing spans is determined solely on surface textual features. Besides, since a portion of input tokens are masked out during the augmentation stage, our parsers can be considered a masked language model as well, except that they are trained with a contrastive objective at the span level.

$$\begin{aligned} \mathcal{P} &= \left\{ j \mid w_{\hat{l}_i:\hat{r}_i} = w_{\hat{l}_j:\hat{r}_j} \right\} \\ \mathcal{N} &= \left\{ j \mid w_{\hat{l}_i:\hat{r}_i} \neq w_{\hat{l}_j:\hat{r}_j} \right\} \end{aligned} \tag{24}$$

From the perspective of training, as Wang et al. (2023) mentioned, one of the most appealing properties of contrastive learning is that it can convert tasks from *wh-questions* to *yes-no questions*. Conventional classification approaches demand embedding vectors for all spans, but enumerating them all is clearly intractable. According to Appendix A, we note that even disregarding the sparsity, employing an embedding with millions of entries is barely practical due to its huge memory consumption. In contrast, our contrastive hashing only needs to know if spans are identical or not, allowing it to pull or push their representations directly without needing to introduce specific embeddings. This property makes previously intractable training feasible and efficient.

From the perspective of representation learning, contrastive learning aims to maximize the distinguishability between instances. In our model, this corresponds to maximizing the distinguishability between subtrees. For parsing, choosing the splitting positions that minimize the internal differences within subtrees is equivalent to maximizing the differences across subtrees. In other words, parsing can be considered as a procedure of searching the minimum entropy tree formed by repeatedly merging the most similar contiguous subtrees, thus, it aligns with the learning objective of contrastive hashing. We believe this explains why such a contrastive hashing procedure results in syntactic trees rather than other structures, and this provides justification for our use of contrastive learning.

3.3 Instance Selection

Contrastive learning (Gao et al., 2021) learns informative representation through pulling together positive and pushing apart negative instances. Wang and Utiyama (2024) enumerate each instance i and compare it with all instances in the batch $j \in \hat{\boldsymbol{t}}$ to compute the instance-level loss $\ell(i, \mu, \hat{\boldsymbol{t}})$, and then aggregate all these losses as the batch-level loss $\mathcal{L}$. By using $\log \sum \exp$ as a approximation of $\max$,

$$\max_{x \in \mathcal{X}} (x) \approx \log \sum_{x \in \mathcal{X}} \exp(x) \tag{25}$$

They tweaked those commonly used contrastive objectives into unified formats as follows, where $\mathcal{S} = \{i\}$ is simply defined as the instance itself.

$$\ell_{\text{self}} \approx \max_{\mathcal{N} \cup \mathcal{P}} s(i,j) - s(i,i) \tag{26}$$

$$\ell_{\text{sup}} \approx \max_{\mathcal{N} \cup \mathcal{P}} s(i,j) - \underset{\mathcal{P}}{\text{mean}}\, s(i,j) \tag{27}$$

$$\ell_{\text{hash}} \approx \max_{\mathcal{N} \cup \mathcal{S}} s(i,j) - s(i,i) \tag{28}$$

$$\begin{aligned} \ell_{\text{max}} &\approx \max_{\mathcal{N} \cup \mathcal{S}} s(i,j) - \max_{\mathcal{P}} s(i,j) \\ &= \max\left(\max_{\mathcal{N}} s(i,j), s(i,i) \right) - \max_{\mathcal{P}} s(i,j) \end{aligned} \tag{29}$$

Objective function ℓ_{self} is commonly utilized in scenarios involving only a single positive instance. Khosla et al. (2020) then extended it as ℓ_{sup} to handle multiple positive instances scenarios, and it was later surpassed by ℓ_{hash} and ℓ_{max}. For more details, we refer readers to their original papers (Wang et al., 2023; Wang and Utiyama, 2024).

Briefly speaking, objective ℓ_{hash} assumes there is only one true positive and excludes potential false negatives and positives from both terms, with $\mathcal{P}$ replaced with $\mathcal{S}$. Moreover, ℓ_{max} adopts a different approach to handling multiple positive instances. They still assume there is only one true positive instance among $\mathcal{P}$, but they dynamically select the closest one as the true positive, instead of statically selecting $\mathcal{S}$. By imposing such a weak alignment signal, they also avoid the geometric center issue of ℓ_{sup}. However, we found that for tasks with large label vocabularies, such as language models, this signal turns out to be too weak. Therefore, instead of pulling only the closest pairs, we propose to mainly focus on the farthest pairs,

$$\begin{aligned} \ell_{\text{min}} &\approx \max_{\mathcal{N} \cup \mathcal{S}} s(i,j) - \min_{\mathcal{P}} s(i,j) \\ &= \max\left(\max_{\mathcal{N}} s(i,j), s(i,i) \right) - \min_{\mathcal{P}} s(i,j) \end{aligned} \tag{30}$$

by introducing a differentiable approximation of $\min$ operator in a simialar matter to Equation 25.

$$\min_{x \in \mathcal{X}} (x) \approx -\log \sum_{x \in \mathcal{X}} \exp(-x) \tag{31}$$

According to experimental results of previous work, excluding potential false negatives seems to be an effective solution, and it also balances the two terms well. However, since $\ell_{\min}$ introduces a strong alignment signal in the positive term, this balance is disrupted. We have consistently observed that the $\ell_{\min}$ model suddenly collapses and starts returning only trivial right-branching trees. We hypothesize the reason is that there is no corresponding uniformity signal in the negative term to balance this strong alignment signal. However, naively adding $\min_{\mathcal{P}}$ to the negative term as the balancing term leads to a new issue,

$$\max\left(\max_{\mathcal{N}} s(i,j), \min_{\mathcal{P}} s(i,j)\right) - \min_{\mathcal{P}} s(i,j)$$

that is when the number of positives is large, positives $\mathcal{P}$ dominate the gradients, leaving insufficient supervision signals to the true negatives $\mathcal{N}$. Therefore, we propose to limit the total gradients of positives to be the same magnitude as single positive by introducing another approximation of $\min$.

$$\overline{\min_{x \in \mathcal{X}}} (x) \approx -\log\left(\frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \exp(-x)\right) \tag{32}$$

In this way, we propose $\ell_{\overline{\min}}$ as a balanced version, with $\min_{\mathcal{P}}$ in the negative term replaced by $\overline{\min}_{\mathcal{P}}$.

$$\ell_{\overline{\min}} \approx \max_{\mathcal{N} \cup \{\overline{\min}_{\mathcal{P}}\}} s(i,j) - \min_{\mathcal{P}} s(i,j) \tag{33}$$

$$= \max\left(\max_{\mathcal{N}} s(i,j), \overline{\min_{\mathcal{P}}} s(i,j)\right) - \min_{\mathcal{P}} s(i,j)$$

3.4 Architecture

Following Wang and Utiyama (2024), our model also consists of a pre-trained language model, an attention hash layer, and a bit-level CKY module. The only difference is that we upgrad CKY from zero-order to first-order, which enhances its ability to unify the representation of lexicon and syntax.

Although it is also a masked language model, our model does not require introducing a large embedding matrix for calculating token classification in the output layer. Since it relies on the attention hash layer to produce binary codes of spans, the number of parameters in the output layer is reduced from $|\mathcal{Y}| \times d$ to two $K \times \lceil \frac{d}{K} \rceil \times d$.

3.5 Training and Inference

During the training stage, sentences are fed into the model twice to obtain two different views by being augmented with different dropout masks. We calculate the marginal probabilities μ^1 and μ^2, and then predict constituency trees $\hat{\boldsymbol{t}}^1$ and $\hat{\boldsymbol{t}}^2$ on these two versions, respectively. For each view, we select the corresponding span scores from the marginal probabilities of one view, according to the predicted tree of the other view, and then perform span-level contrastive hashing by using the objectives above and average them as the batch loss.

$$\mathcal{L} = \operatorname*{mean}_{i \in \hat{\boldsymbol{t}}^2} \ell(i, \mu^1, \hat{\boldsymbol{t}}^2) + \operatorname*{mean}_{i \in \hat{\boldsymbol{t}}^1} \ell(i, \mu^2, \hat{\boldsymbol{t}}^1) \tag{34}$$

Since unsupervised constituency parsing only aims at detecting the span boundaries without needing to predict labels, we do not need to build the code vocabulary as Wang and Utiyama (2024) did. During the inference stage, we simply search for the most probable constituency parsing trees in an unconstrained space with the Cocke-Kasami-Younger (CKY) algorithm (Kasami, 1966).

4 Experiments

4.1 Settings

Experiments are conducted on the commonly used datasets Penn Treebank (PTB) (Marcus et al., 1993) and Chinese Treebank 5.1 (CTB) (Xue et al., 2005).

Following previous settings (Shen et al., 2018b, 2019; Zhao and Titov, 2021), we use the same preprocessing pipeline to discard punctuation in all splits. Although this pipeline may not be the best choice for pre-trained language models and might result in some information loss, since language models are commonly trained with punctuated corpora, we follow this setting only to provide comparable results to previous work. Regarding the evaluation metric, we follow Kim et al. (2019a) to remove trivial spans, i.e., single-word and entire-sentence spans, calculate unlabeled sentence-level F1 scores, and take the average across all sentences.

We use the deep learning framework `PyTorch` (Paszke et al., 2019) to implement our models and download checkpoints of pre-trained languages from `huggingface/transformers` (Wolf et al., 2020). Different from some recent work (Yang et al., 2022; Liu et al., 2023), which require customizing CUDA kernels with `Triton` (Tillet et al., 2019), our model can be easily and efficiently implemented with pure `PyTorch`.

Model	PTB Mean	PTB Max
PRPN (Shen et al., 2018b)$^\flat$	37.4	38.1
URNNG (Kim et al., 2019b)$^\flat$	-	45.4
ON-LSTM (Shen et al., 2019)$^\flat$	47.7	49.4
R2D2 (Hu et al., 2021)$^\flat$	48.1	-
Fast R2D2 (Hu et al., 2022)$^\flat$	48.9	-
StructFormer (Shen et al., 2021)$^\flat$	54.0	-
C-PCFG (Kim et al., 2019a)$^\sharp$	55.2	60.1
NL-PCFG (Zhu et al., 2020)$^\sharp$	55.3	-
DIORA (Drozdov et al., 2019b)$^\flat$	55.7	56.2
GPST (Hu et al., 2024a)$^\flat$	57.5	-
S-DIORA (Drozdov et al., 2020)$^\flat$	57.6	63.9
TN-PCFG (Yang et al., 2021b)$^\sharp$	57.7	61.4
NBL-PCFG (Yang et al., 2021a)$^\sharp$	60.4	-
CT (Cao et al., 2020)†	62.8	65.9
Co (Maveli and Cohen, 2022)†	63.1	66.8
Rank-PCFG (Yang et al., 2022)$^\sharp$	64.1	-
ReCAT (Hu et al., 2024b)$^\flat$	65.0	-
SN-PCFG (Liu et al., 2023)$^\sharp$	**65.1**	-
For Reference		
Ensemble (Shayegh et al., 2024)	70.4	71.9
Left Branching	8.7	8.7
Right Branching	39.5	39.5
Oracle	84.3	84.3
Ours$^\flat$ (BERT$_{\text{Base}}$ - 16 bits)	55.3	58.8
Ours$^\flat$ (BERT$_{\text{Base}}$ - 20 bits)	56.7	59.8
Ours$^\flat$ (BERT$_{\text{Base}}$ - 24 bits)	**57.4**	59.6
Ours$^\flat$ (BERT$_{\text{Base}}$ - 28 bits)	54.5	60.9
Ours$^\flat$ (RoBERTA$_{\text{Base}}$ - 8 bits)	56.5	63.1
Ours$^\flat$ (RoBERTA$_{\text{Base}}$ - 12 bits)	58.0	62.9
Ours$^\flat$ (RoBERTA$_{\text{Base}}$ - 16 bits)	**62.4**	64.1
Ours$^\flat$ (RoBERTA$_{\text{Base}}$ - 20 bits)	59.6	63.9

Table 1: Experiments of unsupervised constituency parsing on the PTB dataset. The columns Mean and Max display the averaged and the maximal unlabeled sentence-level F_1 scores. The **bold numbers** and the underlined numbers indicate the best and the second-best performance. $\flat\sharp\dagger$ stands for implicit grammar, explicit grammar, and probing methods, respectively.

Model	CTB Mean	CTB Max
ON-LSTM (Shen et al., 2019)$^\flat$	25.4	25.7
PRPN (Shen et al., 2018b)$^\flat$	30.4	31.5
Rank-PCFG (Yang et al., 2022)$^\sharp$	32.4	-
C-PCFG (Kim et al., 2019a)$^\sharp$	36.0	39.8
TN-PCFG (Yang et al., 2021b)$^\sharp$	39.2	-
Co (Maveli and Cohen, 2022)†	41.8	43.3
SC-PCFG (Liu et al., 2023)$^\sharp$	42.9	-
R2D2 (Hu et al., 2021)$^\flat$	44.9	-
Fast R2D2 (Hu et al., 2022)$^\flat$	**45.3**	-
For Reference		
Left Branching	9.7	9.7
Right Branching	20.0	20.0
Oracle	81.1	81.1
Ours$^\flat$ (BERT$_{\text{Base}}$ - 28 bits)	41.2	49.0
Ours$^\flat$ (BERT$_{\text{Base}}$ - 32 bits)	43.1	49.5
Ours$^\flat$ (BERT$_{\text{Base}}$ - 36 bits)	**47.1**	49.6
Ours$^\flat$ (BERT$_{\text{Base}}$ - 40 bits)	43.6	49.5
Ours$^\flat$ (RoBERTA$_{\text{Base}}$ - 36 bits)	46.4	50.2
Ours$^\flat$ (RoBERTA$_{\text{Base}}$ - 40 bits)	45.4	50.0
Ours$^\flat$ (RoBERTA$_{\text{Base}}$ - 44 bits)	**48.5**	49.6
Ours$^\flat$ (RoBERTA$_{\text{Base}}$ - 48 bits)	47.0	50.3

Table 2: Experiments of unsupervised constituency parsing on the CTB dataset.

We collect sentences until the total number of spans reaches 1024 to keep the contrastive hashing stable, since it is performed at the span level. We use the Adam optimizer (Kingma and Ba, 2015; Loshchilov and Hutter, 2019) and set the number of warmup and training steps to 4,000 and 20,000, respectively. We randomly select a portion of tokens and replace them with `[MASK]`, following the standard augmentation strategy of masked language models. For PTB experiments, we use checkpoints `bert-base-cased` (Devlin et al., 2019) and `roberta-base` (Liu et al., 2019). For CTB experiments, we use checkpoints `bert-base-chinese` (Devlin et al., 2019) and `chinese-roberta-wwm-ext` (Cui et al., 2020).

We use a single NVIDIA Tesla V100 graphics card to conduct our experiments. Training takes around 30 minutes, which is much faster than the several days of training required by Cao et al. (2020) and Drozdov et al. (2019a). Since we do not modify the architecture of the language model but simply append a hash layer to it, we can fine-tune existing pre-trained language models without needing to train them from scratch, as done by Hu et al. (2022, 2024a). For each setting, we run it four times with different random seeds and report the averaged scores in the following tables.

4.2 Main Results

On the English dataset PTB, as shown in Table 1, our model reaches its peak performance at 24 bits and 16 bits when using BERT and RoBERTa pre-trained language models, respectively. We consis-

NEG	POS	LOSS	PTB MEAN	PTB MAX
$\max_{\mathcal{N}\cup\mathcal{P}}$	$\mathcal{S}$	ℓ_{self}	39.9	40.4
	$\max_{\mathcal{P}}$	-	44.0	54.0
	$\min_{\mathcal{P}}$	-	48.8	61.8
$\max_{\mathcal{N}\cup\mathcal{S}}$	$\mathcal{S}$	ℓ_{hash}	39.9	40.3
	$\max_{\mathcal{P}}$	$\ell_{\max}$	45.5	50.1
	$\min_{\mathcal{P}}$	$\ell_{\min}$	<u>58.2</u>	60.6
$\max_{\mathcal{N}\cup\{\overline{\min}_{\mathcal{P}}\}}$	$\mathcal{S}$	-	35.2	49.1
	$\max_{\mathcal{P}}$	-	47.5	53.9
	$\min_{\mathcal{P}}$	$\ell_{\overline{\min}}$	**62.4**	64.1

Table 3: Ablation study of instance selection strategies in constituency parsing experiments. Columns NEG and POS display the selection strategies for negatives and positives, respectively. LOSS shows this combination corresponds to which loss definition.

tently surpass all other implicit grammar models. Due to the relatively small size of PTB, the probing methods by Cao et al. (2020) and Maveli and Cohen (2022) utilized additional text data for training. Even without using such extra data, our model still achieves performance very close to theirs.

Our model outperforms all existing models by a large margin on the Chinese dataset CTB, as shown in Table 2. Explicit grammar models that perform well on English datasets (Yang et al., 2022; Liu et al., 2023) do not achieve similar success on the Chinese dataset. Additionally, we notice that our model requires much more bits than on the English dataset, i.e., 36 and 44, to reach their full potential. We hypothesize that this is due to the relatively small size of the Chinese dataset, as shown in Appendix A, which prevents the models from being fully trained to encode lexicon and syntax features within only a few bits.

4.3 Ablation Studies

Table 3 reveals how the different combinations of negative and positive terms affect performance. First of all, we notice that once $\min_{\mathcal{P}}$ is employed, regardless of which negative terms are used along with it, the models consistently result in high scores in the MAX column. On the contrary, without employing $\min_{\mathcal{P}}$, these scores dramatically drop. This confirms our statement that for tasks with large label vocabularies, positive terms require strong alignment signals to learn effective representations. Moreover, when it comes to the MEAN column, whether the term $\max_{\mathcal{N}\cup\{\overline{\min}_{\mathcal{P}}\}}$ is employed determines whether the high scores of $\min_{\mathcal{P}}$ can be maintained. We also notice that $\max_{\mathcal{N}\cup\mathcal{S}}$ consistently outperforms $\max_{\mathcal{N}\cup\mathcal{P}}$. This indicates that simply pushing away all instances of $\mathcal{P}$ indeed introduces the false negatives issue. As Wang et al. (2023); Wang and Utiyama (2024) claims, retaining only $\mathcal{S}$ mitigates this issue, but when $\mathcal{P}$ is introduced back to the positive term under a strong alignment, the lack of uniformity signals brings a new imbalance issue, and our solution $\ell_{\overline{\min}}$ rebalances them by using $\overline{\min}_{\mathcal{P}}$ in both terms.

4.4 Case Studies

Figure 3 shows an example of our parsing results, with more examples available in Appendix B. Relying on the implicitly induced grammar, our model provides remarkably accurate parsing results, with all constituents correctly selected. Additionally, the hashing results also demonstrate the impressive capability in discovering syntactic categories. For instance, both preterminal symbols like adjectives, e.g., *quick* (`5A00`), *brown* (`5E42`), and *lazy* (`5E03`), and nonterminal symbols like noun phrases, e.g., *the quick brown fox* (`7EBB`) and *the lazy dog* (`EEBB`), are assigned similar and relevant binary codes to each other. This phenomenon can also be observed in sentences in Appendix B, indicating that our parser can accurately reveal both part-of-speech and constituent features at different hierarchical levels.

5 Related Work

Syntactic language models, as a historical and important field of language models, had been widely studied even before the deep learning era (Chelba and Jelinek, 2000; Charniak, 2001; Roark, 2001; Klein and Manning, 2002, 2004; Bod, 2006a,b). After that, Shen et al. (2018a,b, 2019) added syntactic inductive bias to LSTM by introducing master gates to control the information flow in hierarchical directions, thereby enabling the model to learn syntactic distance. Under this framework, by training recurrent language models in the usual way, they can obtain parsers that implicitly structure sentences according to the learned syntactic distances. They have also successfully applied this method to transformers (Shen et al., 2021).

Implicit grammar models induce grammar during the training process by incrementally constructing larger span representations. Kim et al. (2019b) were the first to extend the recurrent neural network grammar (RNNG) (Dyer et al., 2016) from super-

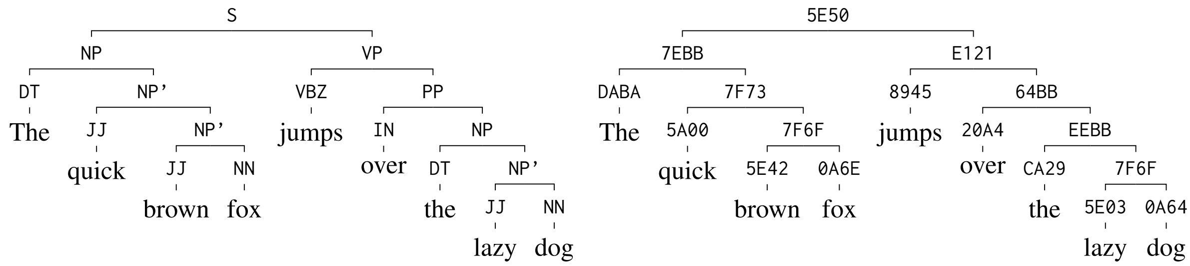

Figure 3: Derivation of the sentence *The quick brown fox jumps over the lazy dog*. The left side is the ground-truth consistency tree, and the right side is our parsing result with binary labels represented in `hexadecimal` format.

vised to unsupervised settings. They build parsing trees through continuously shifting and reducing, without introducing explicit production rules. Additionally, DIORA (Drozdov et al., 2019b, 2020) construct span representations and update charts in both inside and outside passes, and then encourage consistency between them. Similarly, R2D2 (Hu et al., 2021, 2022) is trained in a similar manner, but with Gumbel-softmax (Jang et al., 2017) introduced during the tree construction.

Apart from implicit grammar models, explicitly inducing probabilistic context-free grammar (PCFG) is also widely focused. Kim et al. (2019a) first brought PCFG approaches back with a neural parameterization technique and trained language models to reconstruct entire input sentences token by token. Zhu et al. (2020) shows that additionally modeling lexical dependencies (Collins, 2003) is effective, and Jiang et al. (2016); Yang et al. (2021a) further confirmed extending to bilexical dependencies is even more beneficial. Besides, Yang et al. (2021b, 2022); Liu et al. (2023) claimed that the limited number of symbols is a bottleneck of PCFG induction, and proposed to introduce more symbols by applying tensor decomposition to overcome the cubic computational complexity.

How much syntactic knowledge is preserved within the ordinary language model is also a question worth considering. Mareček and Rosa (2018, 2019) noticed the value of attention scores first. They defined distance functions similar to our Equation 17 for probing. However, they did not consider splitting positions, and relied only on fixed attention heads without having them fine-tuned. As a result, their methods resulted in limited accuracy. Hewitt and Manning (2019); Wang et al. (2019); Kim et al. (2020); Li et al. (2020); Bai et al. (2021) then introduced parameterized functions to prob syntactic distances on hidden states and attention scores, and then fine-tuned the entire models. Cao et al. (2020); Maveli and Cohen (2022) fine-tuned pre-trained language models with an additional classifier to distinguish manually generated constituents and distituents, and utilized predictions from this classifier to determine splitting positions on parsing trees during the evaluation stage.

In various senses, our approach confirmed the conclusions of many previous works and further pushed their limits. First, by switching the language models from token-level to span-level, we confirmed that modeling lexical dependencies is beneficial and extending this modeling to all tokens in children is more effective. Additionally, by introducing binary representation, we confirmed that employing more symbols is advantageous and further scaling up to 2^K can help parsers do further better. Third, we confirmed that constructing span representations and updating the chart is helpful, and unifying the representation of lexicon and syntax leads to more competitive results. Finally, we confirmed the multi-head attention scores already preserve syntactic information, and fine-tuning them can help probe for more insightful features.

6 Conclusions

In this paper, we confirmed that the information-preserving capability of binary representation is effective at both lexicon and syntax levels, and we demonstrated that it is feasible to elicit parsers from pre-trained language models by leveraging this capability. We achieved this by upgrading bit-level CKY from zero-order to first-order, extending contrastive hashing from supervised to unsupervised, and proposing a novel objective function to impose stronger yet balanced alignment signals. Experiments show our model achieves competitive performance, and also indicate that the technique for acquiring high-quality syntactic annotations at low cost has now reached a practical stage.

Limitations

We successfully obtain parsers in an unsupervised manner. Nonetheless, the number of bits remains a hyperparameter that needs to be tuned by testing them individually. Although, in practice, enumerating from 8 to 48 is sufficient for most cases, the relationship between the required number of bits and the specified task remains unclear. Therefore, we aim to explore this issue in future work. Moreover, we simply define the left and right span representations as the average of their token vectors, as shown in Equation 17. The reason for using such a naive definition is a compromise for the sake of speed. However, it is evident that this simple linear mapping may not efficiently preserve high-order information, and future work could explore more complex mechanisms.

References

Jiangang Bai, Yujing Wang, Yiren Chen, Yaming Yang, Jing Bai, Jing Yu, and Yunhai Tong. 2021. Syntax-BERT: Improving pre-trained transformers with syntax trees. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3011–3020, Online. Association for Computational Linguistics.

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

Rens Bod. 2006b. Unsupervised parsing with U-DOP. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 85–92, New York City. Association for Computational Linguistics.

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

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 124–131, Toulouse, France. Association for Computational Linguistics.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech & Language*, 14(4):283–332.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for Chinese natural language processing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 657–668, Online. Association for Computational Linguistics.

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

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

Andrew Drozdov, Patrick Verga, Yi-Pei Chen, Mohit Iyyer, and Andrew McCallum. 2019a. Unsupervised labeled parsing with deep inside-outside recursive autoencoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1507–1512, Hong Kong, China. Association for Computational Linguistics.

Andrew Drozdov, Patrick Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019b. Unsupervised latent tree induction with deep inside-outside recursive auto-encoders. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1129–1141, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Jason Eisner. 2016. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 1–17, Austin, TX. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference* on Empirical Methods in Natural Language Processing, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

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

Xiang Hu, Pengyu Ji, Qingyang Zhu, Wei Wu, and Kewei Tu. 2024a. Generative pretrained structured transformers: Unsupervised syntactic language models at scale. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2640–2657, Bangkok, Thailand. Association for Computational Linguistics.

Xiang Hu, Haitao Mi, Liang Li, and Gerard de Melo. 2022. Fast-R2D2: A pretrained recursive neural network based on pruned CKY for grammar induction and text representation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2809–2821, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xiang Hu, Haitao Mi, Zujie Wen, Yafang Wang, Yi Su, Jing Zheng, and Gerard de Melo. 2021. R2D2: Recursive transformer based on differentiable tree for interpretable hierarchical language modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4897–4908, Online. Association for Computational Linguistics.

Xiang Hu, Qingyang Zhu, Kewei Tu, and Wei Wu. 2024b. Augmenting transformers with recursively composed multi-grained representations. In *The Twelfth International Conference on Learning Representations*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2016. Unsupervised neural dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 763–771, Austin, Texas. Association for Computational Linguistics.

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

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673. Curran Associates, Inc.

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

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

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

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *The Third International Conference on Learning Representations*.

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

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

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

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

Wei Liu, Songlin Yang, Yoon Kim, and Kewei Tu. 2023. Simple hardware-efficient PCFGs with independent left and right productions. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1662–1669, Singapore. Association for Computational Linguistics.

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

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

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

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

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

Nickil Maveli and Shay Cohen. 2022. Co-training an Unsupervised Constituency Parser with Weak Supervision. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1274–1291, Dublin, Ireland. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Alec Radford. 2018. Improving language understanding by generative pre-training. *OpenAI*.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Behzad Shayegh, Yanshuai Cao, Xiaodan Zhu, Jackie CK Cheung, and Lili Mou. 2024. Ensemble distillation for unsupervised constituency parsing. In *The Twelfth International Conference on Learning Representations*.

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

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

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

Yikang Shen, Yi Tay, Che Zheng, Dara Bahri, Donald Metzler, and Aaron Courville. 2021. StructFormer: Joint unsupervised induction of dependency and constituency structure from masked language modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7196–7209, Online. Association for Computational Linguistics.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Philippe Tillet, H. T. Kung, and David Cox. 2019. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, MAPL 2019, page 10–19, New York, NY, USA. Association for Computing Machinery.

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

Yiran Wang and Masao Utiyama. 2024. To be continuous, or to be discrete, those are bits of questions. In *Proceedings of The 62nd Annual Meeting of the Association for Computational Linguistics*, Bangkok, Thailand. Association for Computational Linguistics.

Yiran Wang, Taro Watanabe, Masao Utiyama, and Yuji Matsumoto. 2023. 24-bit languages. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 408–419, Nusa Dua, Bali. Association for Computational Linguistics.

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

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Songlin Yang, Wei Liu, and Kewei Tu. 2022. Dynamic programming in rank space: Scaling structured inference with low-rank HMMs and PCFGs. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4797–4809, Seattle, United States. Association for Computational Linguistics.

Songlin Yang, Yanpeng Zhao, and Kewei Tu. 2021a. Neural bi-lexicalized PCFG induction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2688–2699, Online. Association for Computational Linguistics.

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

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics.

Yu Zhang, Houquan Zhou, and Zhenghua Li. 2020. Fast and accurate neural crf constituency parsing. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4046–4053. International Joint Conferences on Artificial Intelligence Organization. Main track.

Yanpeng Zhao and Ivan Titov. 2021. An empirical study of compound PCFGs. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, pages 166–171, Kyiv, Ukraine. Association for Computational Linguistics.

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

A Datasets Statistics

DATASET	TRAIN	DEV	TEST	WORD	SPAN
PTB	39,832	1,700	2,416	44,363	8,865,092
CTB	18,104	352	348	36,800	6,510,230

Table 4: Datasets statistics. Columns TRAIN, DEV, and TEST show the number of sentences in each split, while Columns WORD and SPAN indicate the number of words and spans, respectively.

B More Examples

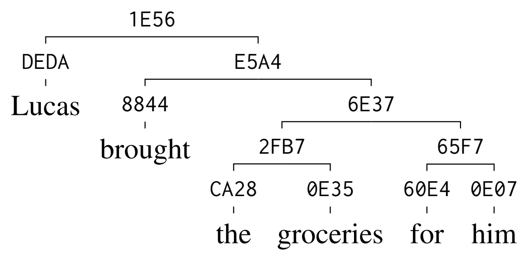

Figure 4: Derivation example.

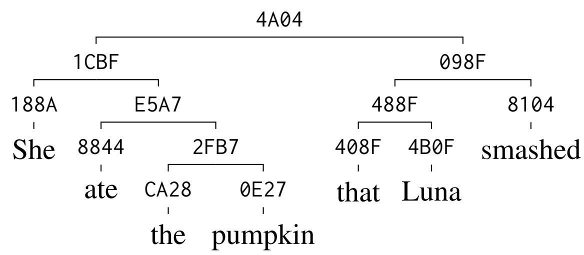

Figure 5: Derivation example.

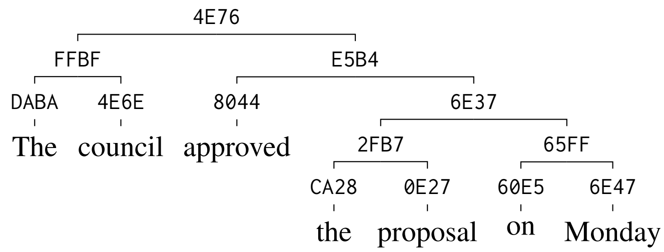

Figure 6: Derivation example.

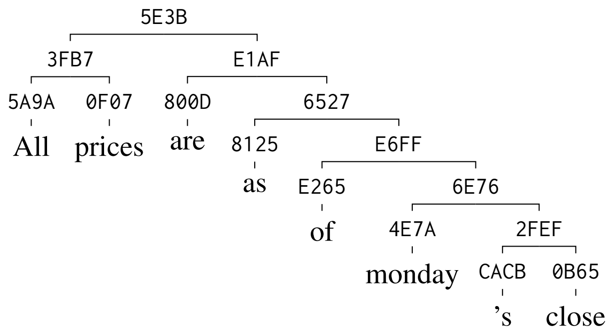

Figure 7: Derivation example.

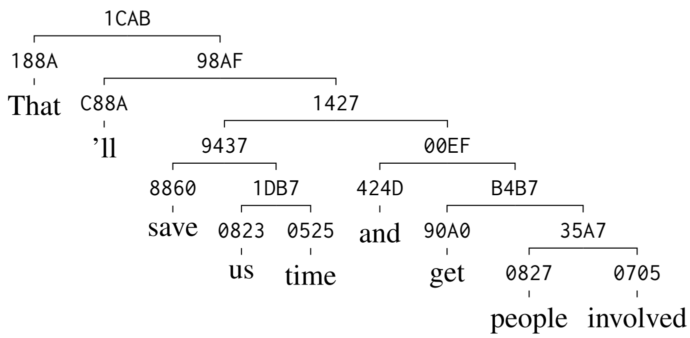

Figure 8: Derivation example.

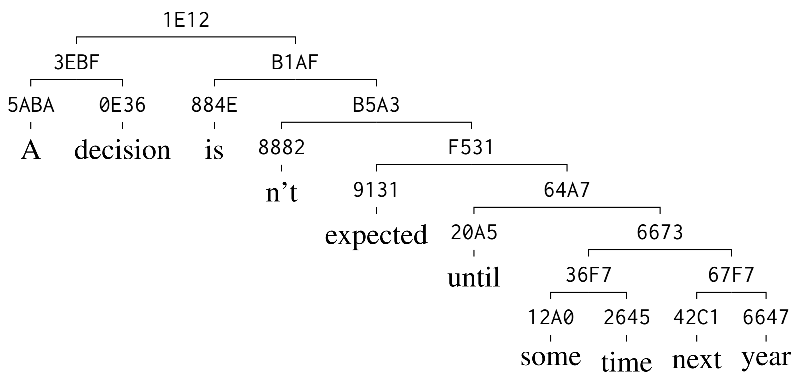

Figure 9: Derivation example.

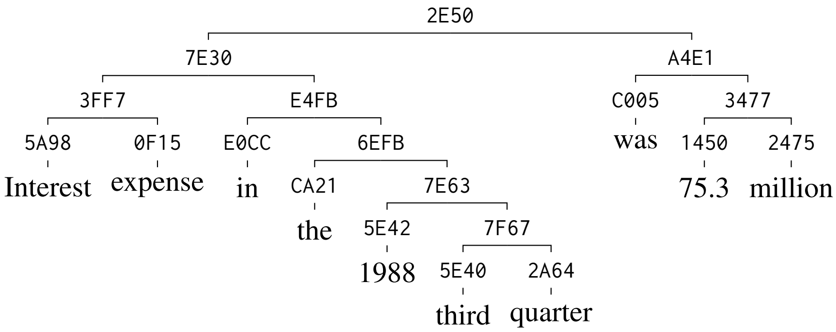

Figure 10: Derivation example.

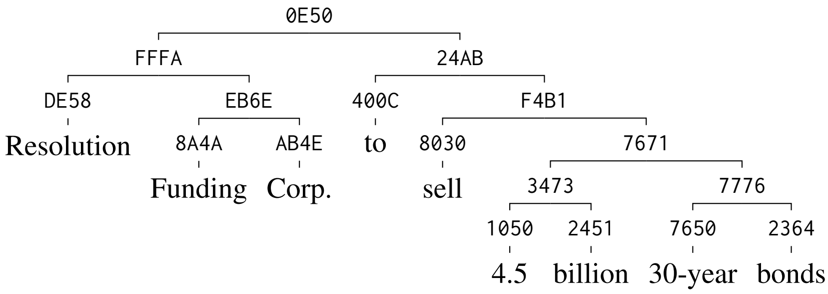

Figure 11: Derivation example.

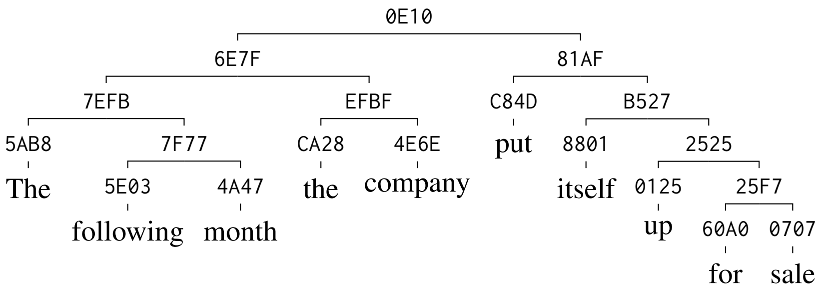

Figure 12: Derivation example.

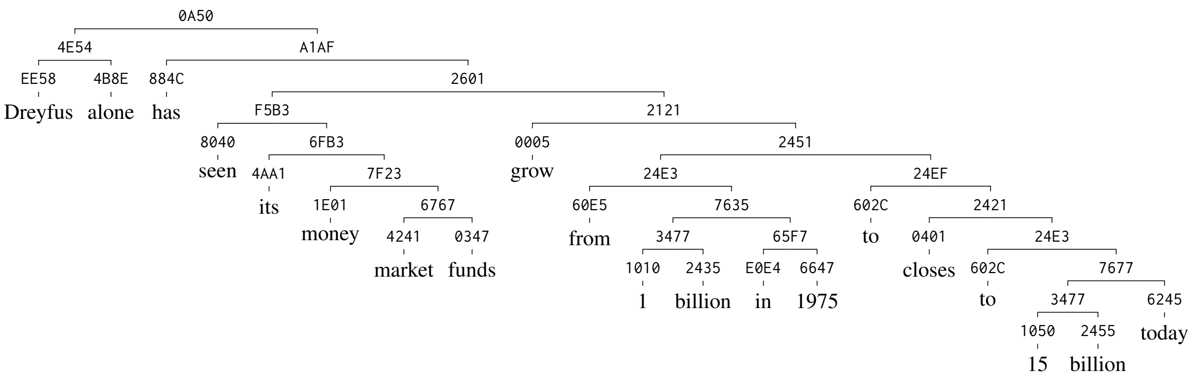

Figure 13: Derivation example.

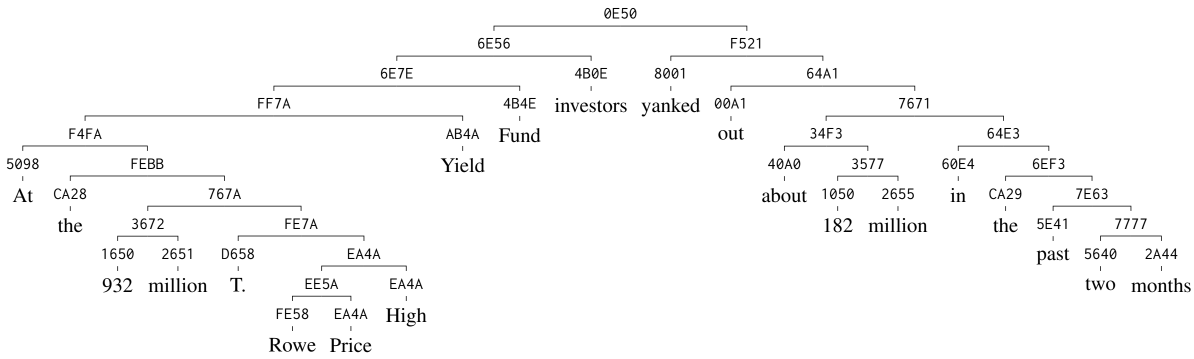

Figure 14: Derivation example.

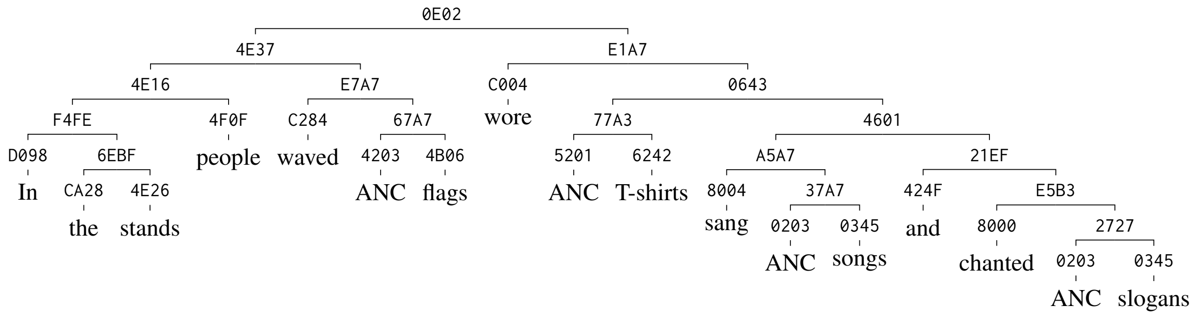

Figure 15: Derivation example.

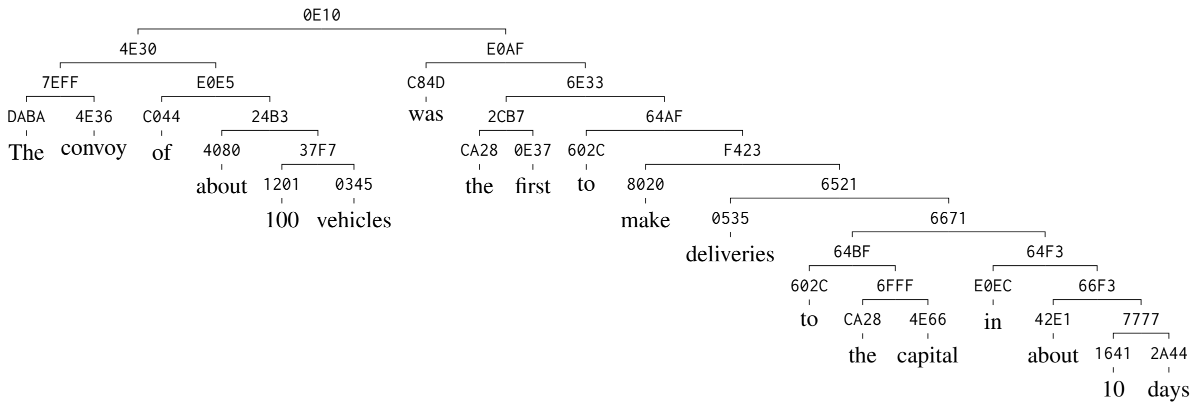

Figure 16: Derivation example.