A Grammar-Based Method for Instilling Empirical Dependency Structure in LLMs

Olle Torstensson and **Oskar Holmström**
Linköping University
{olle.torstensson, oskar.holmstrom}@liu.se

Abstract

We investigate whether synthetic pretraining data generated from a formal grammar modeling syntactic dependencies can improve English language models. Building upon the structured pretraining data approach of Papadimitriou and Jurafsky (2023), we develop a grammar that more closely mirrors empirical dependency structures. Our results are negative – this type of pretraining significantly degrades model performance, with both our and their pretraining approach performing worse than no pretraining at all. We analyze potential explanations for these findings and discuss implications for future work on structured-data pretraining.

1 Introduction

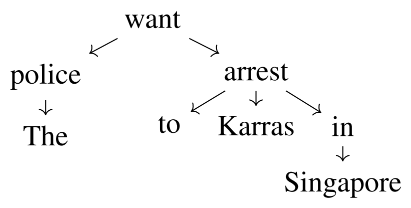
Language learners – human and artificial ones alike – are able to pick up on structural features of natural language without being subjected to any *explicit* structural supervision. Recently, Papadimitriou and Jurafsky (2023) investigated the question of what makes this learning possible – specifically by what *structural biases* it is facilitated (throughout, we will refer to this paper as (PJ23)). In this undertaking, they lean on theories about structural predispositions, such as recursion, being inherent in humans (Hauser et al., 2002). To test this, they instilled an artificial language learner with *structural knowledge* of language, without teaching it anything about the actual *contents* of language. In more practical terms, they pretrained a neural language model on synthetic languages – whose vocabulary consists of integers rather than words – meant to model dependency structures of natural language, in order to then finetune it on actual natural language.

While their main goal was understanding human language acquisition, their results suggest language models may benefit from this pretraining approach. So, with the alternative goal of improving language models, the question of whether there are more suitable pretraining languages for this type of transfer learning arises. We construct a weighted context-free grammar that formally generates one such language, in which the distribution of word dependencies more closely follows that of natural language. The weights in our grammar model the frequencies of dependency links in a given natural language; for higher-resourced languages these frequencies can be directly extracted from dependency treebanks, whereas for lower-resourced languages they would need to be estimated in some other way. Formalisms like Constraint Grammar (Karlsson, 1990) are perfectly capable of this task and well-developed for several low-resourced languages (Pirinen et al., 2023). In such a setting our approach could be particularly valuable since existing linguistic knowledge can be leveraged to increase the speed of model convergence during training without consuming additional textual resources.

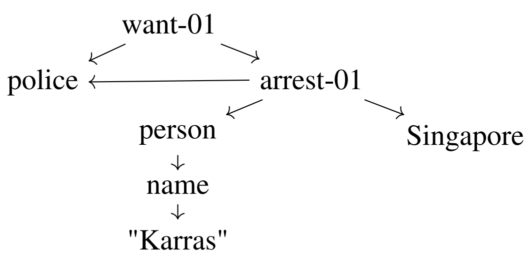
We use our generated language to pretrain a neural language model and compare the performance, measured in terms of perplexity after finetuning on English, to that of (PJ23).

2 Background: Word dependencies

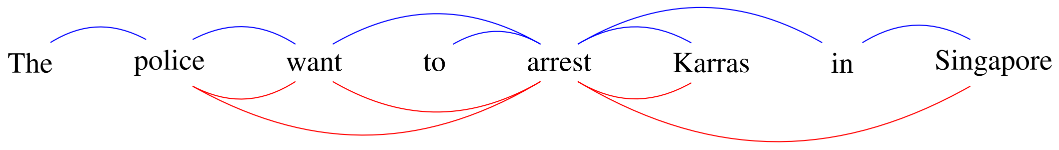
Word dependencies appear in both syntactic and semantic analysis of natural language. In the case of syntax, the structure of a sentence is usually represented by a tree (see Figure 1a) in which edges denote dependencies, whereas the semantics may be represented by a directed acyclic graph (DAG) (see Figure 1b). Some subsets of dependencies form recursive patterns, i.e., ones in which edges are nested, whereas others contain crossing edges (see Figure 2). The distribution of dependency lengths (the distance between two dependent words) is largely language-dependent (Oya, 2021)



(a) Syntax tree



(b) Abstract meaning representation (AMR) graph



(c) Flattened and undirected version of the syntax tree (top, in blue) and the AMR (bottom, in red)

Figure 1: Syntactic and semantic representations of the sentence *The police want to arrest Karras in Singapore*. Example taken from (Wang et al., 2015).

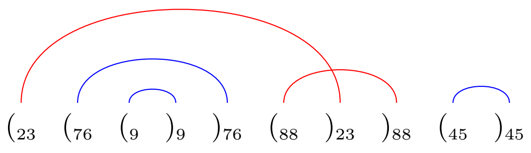


Figure 2: Example string from pretraining data. For the sake of presentation, we view matching numbers as numbered bracket pairs. Crossing edges are marked in red.

and furthermore very much dependent on sentence length (Ferrer-i-Cancho and Liu, 2014).

3 Generating pretraining data

Following (PJ23), to instill knowledge of dependency structure in an untrained neural language model without granting it access to the vocabulary, we generate synthetic pretraining data consisting of numeric sequences (see Figure 2), where matching numbers indicate a dependency between the indices.[1] While our data generation process shares these features with (PJ23), there are some key differences.

3.1 The NEST-MIX-p language of (PJ23)

In (PJ23), the authors define a formal probabilistic model for generating strings which they call NEST-MIX-p, where p is a parameter. A string is constructed sequentially, and at each step a new bracket pair is opened with probability 0.49 and the most recently opened one is closed with probability 0.51 – with the exception that with probability p, a newly opened bracket pair will be a crossing one and its length randomly decided based on an empirical distribution of dependency lengths.[2] This process is continued until the sequence is of length exactly 512.

3.2 Our EMP-DEP language

An advantage of generating the pretraining data in a completely artificial way without relying on an empirical distribution is that one can attempt to model both syntactic and semantic dependencies at the same time. Since we wish to model the dependencies from natural language, a choice between the two is required. Mainly due to data availability, we choose to model our data on *syntactic* dependencies. For the same reason, we also need to commit to a target language from which we model the dependencies already in the pretraining phase (although, technically this is also the case for NEST-MIX-p – see footnote 2).

To synthesize strings, we perform a weighted sampling from a formal language EMP-DEP defined via a *weighted context-free grammar* – a context-free grammar where production rules are equipped with weights, which are then multiplied upon application. In our grammar, these weights represent the frequencies at which a dependency

[1]Thus, dependencies have no direction in these strings.

[2]While the fact that the length of the crossing dependencies follow an empirical distribution makes the language somewhat grounded in reality, the distribution is not exclusive to crossing dependencies and does not take sentence length into account. Note also that because of this, a bias toward a *specific* natural language is introduced.

link between two indices occur in syntactical dependency trees, given the sentence length. Although these frequencies could in principle be estimated with the help of Constraint Grammar or other rule-based methods of dependency parsing, we have opted to utilize the readily available dependency trees found in the English Universal Dependencies Treebank[3]. From this collection of trees, we extract (1) a distribution over sentence lengths; and (2) for each sentence length, a distribution over dependence arcs (i.e., not just their lengths, but also their positions in the sentence).

In what follows, let m be the size of our integer vocabulary, let $n_{\max}$ denote a maximum sentence length, and, for all positive integers n, i, j with $i, j \leq n \leq n_{\max}$, let $p^{(n)}$ denote the (normalized) frequency of sentence length n and let $p_{ij}^{(n)}$ denote the (normalized) frequency of the undirected dependency arc i-j given sentence length n.

Before we define our grammar, we cover a couple of additional notational conventions: for any positive integer k, we let $[k] = \{1, \ldots, k\}$ and $[k]_0 = \{0, \ldots, k\}$, and, for any two non-negative integers i and j, we let $\overline{x}_{[i:j]}$ denote the string $x_i \cdots x_j$. If $i > j$, then $\overline{x}_{[i:j]} = \varepsilon$, where ε denotes the empty string. Finally, we construct EMP-DEP via the weighted context-free grammar $(N, [m-1]_0, P, S)$, defined as follows:

First, let $A = \{u\} \cup \{a_k \mid k \in [m-1]_0\}$, after which we define the set of non-terminals

$$N = \{S\} \cup \{\langle w \rangle \mid w \in A^* \text{ and } |w| \leq n_{\max}\}.$$

Then, for all positive integers $n \leq n_{\max}$ and $k, k_1, \ldots, k_n \in [m-1]_0$, P contains the rules

(1) $S \xrightarrow{p^{(n)}} \langle \underbrace{u \cdots u}_{n} \rangle$

(2) $\langle \overline{x}_{[1:i-1]} \ u \ \overline{x}_{[i+1:j-1]} \ u \ \overline{x}_{[j+1:n]} \rangle \xrightarrow{p_{ij}^{(n)}/m} \langle \overline{x}_{[1:i-1]} \ a_k \ \overline{x}_{[i+1:j-1]} \ a_k \ \overline{x}_{[j+1:n]} \rangle$

for all $i, j \in [n]$ and $x_\ell \in A$, where $\ell \in [n]$, such that $p_{ij}^{(n)} \neq 0$

(3) $\langle \overline{x}_{[1:i-1]} \, u \, \overline{x}_{[i+1:n]} \rangle \xrightarrow{1/m} \langle \overline{x}_{[1:i-1]} \, a_k \, \overline{x}_{[i+1:n]} \rangle$

for all $i \in [n]$ and $x_\ell \in A$, where $\ell \in [n]$, such that $p_{i\ell}^{(n)} = 0$ or $x_\ell \neq u$

(4) $\langle a_{k_1} \cdots a_{k_n} \rangle \xrightarrow{1} k_1 \cdots k_n$

[3] `https://universaldependencies.org/`

The elements in A are used to denote whether an index is unassigned (u) or assigned a number k (a_k). Rules (1) produce a non-terminal with a number of unassigned indices depending on the sentence length distribution. Rules (2) pair indices by assigning them the same random number, according to the dependency distribution conditioned on the specific sentence length. Via rules (3), remaining indices that cannot be paired (these will exist, e.g., whenever n is odd) are filled with random numbers. Finally, in rules (4), the strings of numbers themselves are produced.

Generating the pretraining data in this way allows for a virtually unlimited number (depending on parameters) of samples, in which all dependencies appear in empirical data. Note however that dependencies that do not occur *together* empirically in a sentence, may do so in our samples, and that the (non-)crossing property of edges may not be preserved.

4 Experiments

All the necessary code to generate the data and run the experiments can be found at `https://github.com/olletorstensson/emp-dep`.

4.1 Data

For our experiments, we produce two datasets: one generated from our language EMP-DEP, and one generated by the NEST-MIX-0.1 procedure of (PJ23).[4] Following the experimental setup of (PJ23), each dataset consists of 1 billion tokens from a vocabulary of integers in the interval $[0, 499]$, and pretrain a language model each on them. The baseline is a randomly initialized model that is not subjected to any pretraining.[5] We then finetune all three models on the relatively modestly sized WikiText-103 English dataset (Merity et al., 2017) made up of 103 million tokens, and evaluate the models in terms of perplexity on its test set.

4.2 Setup

All three models in the experiments have a 124-million-parameter-sized GPT-2 architecture with a vocabulary size of 50,257 which were (with the exception of the baseline model) trained on the pretraining data for 5,000 steps using a batch size of

[4] While their procedure CROSS performs better than NEST-MIX-0.1 by a small margin, there seems to be no dedicated code or sufficiently precise description of it.

[5] This baseline is different from the one in (PJ23), due to our different end goals (see Section 5).

Pretraining	Perplexity (Mean ± Std)
None (baseline)	36.04 ± 0.07
NEST-MIX-0.1	69.66 ± 6.44
EMP-DEP	261.60 ± 69.29

Table 1: Perplexity of different pretrained models and the baseline on the finetuning test data. Average over 5 random seeds, lower is better.

512. The models are then finetuned on the English data during 2 epochs. To handle the different size of the pretraining and finetuning vocabularies, the new word embeddings are randomly sampled from the old ones as opposed to being randomly initialized, as this has been observed to facilitate transfer learning (Wu et al., 2023).

4.3 Results

The results of the evaluation are given in Table 1. It is clear from these numbers that pretraining in any of the two forms does more harm than good in this experimental setting, and that, in addition, the performance of the model biased using our EMP-DEP language falls far behind the one trained on NEST-MIX-0.1.

5 Discussion

Three key aspects of our results warrant an explanation.

The baseline model performed best. Our experimental setup largely follows that of (PJ23), which in the end might be more adapted to their research question, i.e., *"How is language learning affected by a structural bias?"* than ours, i.e., *"Can language models be improved with a structural bias?"*. Specifically, the effects of pretraining and finetuning data size could play a role here. In their experiments they use a different baseline model from ours, namely an already trained model for which the word embeddings had been resampled (before finetuning), as opposed to a randomly initialized one. It might be the case that the embedding resampling is such a setback that the amount of finetuning data is not enough for such a model to recover from it, whereas our baseline model has no such setback to recover from.

The worst performing model, by far, is ours. In our data generation process, we make a number of compromises. Firstly, the commitment to syntax over semantics results in data that is perhaps less useful for learning some dependencies – especially long-term ones. Secondly, only empirical sentence lengths and dependencies appear in the produced data, without any extrapolation it is probably more difficult for the model to generalize in the finetuning phase. Thirdly, as sentence length is integral to the process, the examples from our data are of variable length, in contrast to the constant length produced by NEST-MIX-0.1. As a consequence, our pretraining language is more difficult for the model to learn[6] and may not model English structure as effectively as NEST-MIX-0.1.

The experiments fail to replicate the results of (PJ23). The fact that the perplexity of our NEST-MIX-0.1-infused model is much higher than reported in (PJ23) has several possible explanations, the most likely of which being that some of the training parameters not presented in (PJ23) differ between the two experiments. Our baseline's superior performance, unlike in (PJ23), results from using different baseline models (see first paragraph).

6 Conclusion and Future Work

In this paper, we have limited ourselves to only investigate the perplexity effects of pretraining a language model on specific structured data. While our results do not demonstrate immediate benefits, it would be valuable to examine whether our model actually learns *anything* useful from our type of pretraining by studying attention patterns. Future research directions include alternative grammars, different model architectures, and more targeted downstream tasks. One promising avenue, suggested by a reviewer, would be hybrid approaches using formalisms like Constraint Grammars to enrich training data with full dependency structure, which could help determine whether adding structural information to lexical information benefits model performance.

As recent work supports the potential of structured pretraining data (Lindemann et al., 2024b; Finn et al., 2017; Krishna et al., 2021; Lindemann et al., 2024a; Wu et al., 2021), the injection of structural biases into language models continues to be an important research direction, particularly for less-resourced languages where synthetic data could compensate for scarce resources.

[6]While the decrease in validation loss during pretraining indicates that the model made progress in this respect, it was higher than for NEST-MIX-0.1 in the end.

Acknowledgements

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation, and the National Graduate School of Computer Science in Sweden (CUGS). The computations were enabled by the Alvis cluster provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725. We also thank Marco Kuhlmann for inspiring discussions and for providing feedback on an early version of this paper, and Kevin Glocker for assisting with various code-related issues.

References

Ramon Ferrer-i-Cancho and Haitao Liu. 2014. The risks of mixing dependency lengths from sequences of different length. *Glottotheory*, 5(2):143–155.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

Marc D. Hauser, Noam Chomsky, and W. Tecumseh Fitch. 2002. The faculty of language: What is it, who has it, and how did it evolve? *Science*, 298(5598):1569–1579.

Fred Karlsson. 1990. Constraint grammar as a framework for parsing running text. In *COLING 1990 Volume 3: Papers presented to the 13th International Conference on Computational Linguistics*.

Kundan Krishna, Jeffrey Bigham, and Zachary C. Lipton. 2021. Does pretraining for summarization require knowledge transfer? In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3178–3189, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Matthias Lindemann, Alexander Koller, and Ivan Titov. 2024a. SIP: Injecting a structural inductive bias into a Seq2Seq model by simulation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6570–6587, Bangkok, Thailand. Association for Computational Linguistics.

Matthias Lindemann, Alexander Koller, and Ivan Titov. 2024b. Strengthening structural inductive biases by pre-training to perform syntactic transformations. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11558–11573, Miami, Florida, USA. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Masanori Oya. 2021. Three types of average dependency distances of sentences in a multilingual parallel corpus. In *Proceedings of the 35th Pacific Asia Conference on Language, Information and Computation*, pages 652–661, Shanghai, China. Association for Computational Lingustics.

Isabel Papadimitriou and Dan Jurafsky. 2023. Injecting structural hints: Using language models to study inductive biases in language learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8402–8413, Singapore. Association for Computational Linguistics.

Flammie Pirinen, Sjur Moshagen, and Katri Hiovain-Asikainen. 2023. GiellaLT — a stable infrastructure for Nordic minority languages and beyond. In *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 643–649, Tórshavn, Faroe Islands. University of Tartu Library.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A transition-based algorithm for AMR parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375, Denver, Colorado. Association for Computational Linguistics.

Yuhuai Wu, Markus N. Rabe, Wenda Li, Jimmy Ba, Roger B. Grosse, and Christian Szegedy. 2021. LIME: learning inductive bias for primitives of mathematical reasoning. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 11251–11262. PMLR.

Zhengxuan Wu, Alex Tamkin, and Isabel Papadimitriou. 2023. Oolong: Investigating what makes transfer learning hard with controlled studies. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3280–3289, Singapore. Association for Computational Linguistics.