Does Learning Syntax Help Models Learn Language?

Stanford CS224N Custom Project

Lian Wang
Department of Computer Science
Stanford University
lianwang@stanford.edu

Abstract

Papadimitriou and Jurafsky (2020) showed that LSTMs trained on nonlinguistic structural data performed significantly better than random baselines when evaluated zero-shot on language tasks, which suggests that models can learn generalizable structure independent of specific vocabularies. In this paper, I replicate that finding for transformer models and introduce a new synthetic corpus that captures a different type of structure (i.e., distributional categories). I find that models trained on this corpus outperform models trained on corpora with binary dependency structures, which shows that models are sensitive to finer-grained structural differences, and certain types of structures are better than others as inductive biases for language learning.

1 Key Information to include

- Mentor: Isabel Papadimitriou
- External Collaborators (if you have any): N/A
- Sharing project: No

2 Introduction

Language models display a remarkable amount of syntactic "knowledge", despite never being taught combinatorial rules and only being optimized for next-word prediction. Much analytic work has been dedicated to trying to understand how much syntax models actually learn and how they represent it (e.g. Linzen et al., 2016; Hewitt and Manning, 2019; Chi et al., 2020). On the engineering side, understanding how these models work, and importantly, diagnosing where they excel and fail, is crucial for designing better systems.

Moreover, the question of how languages learn, represent, and use syntax is also of theoretical interest. For human language, it is hypothesized that, through a combination of biological bias and later language exposure, we learn a set of syntactic operations and constraints, which are independent of specific lexical items or even the features of a specific language (Chomsky, 1965; Hauser et al., 2002). On the other hand, neural models rely heavily on semantic features and co-occurrence statistics of specific lexical items (Papadimitriou et al., 2021), yet they are able to display somewhat human-like linguistic behavior. Thus it is interesting for linguists to see how well a purely probabilistic model can learn language and how language is represented in such a system, and also important for computer scientists to study their limitations and fundamental differences from the human language system.

Beyond simply asking how much syntax models know, there is a more specific question that pertains to the above motivations: How much purely structural knowledge do models learn and utilize, independent of the semantics of a specific system?

Using the method proposed in Papadimitriou and Jurafsky (2020), I probe this question by studying the extent to which different types of structural information serve as inductive biases for further training and evaluation on natural language.

3 Related work

Past work has approached the question of how models represent syntax in many different ways. Using a probing approach, Hewitt and Manning (2019) recovers syntactic tree distances between two words in a sentence through a linear transformation on their vector representations, showing that models do implicitly encode structural distance. Chi et al. (2020) further shows that models also encode dependency labels in a way that holds cross-linguistically. By training classifiers to detect certain grammatical features, Papadimitriou et al. (2021) shows that models are furthermore sensitive to subtle language-specific parameters, like morphosyntactic alignment.

Interestingly, Papadimitriou et al. (2021) also shows that classifier decisions about subjecthood, usually considered a syntactic notion, are highly dependent on semantic features like animacy and agency, which tend to co-occur with subjects. This highlights that, while models encode a substantial amount of syntax, there is no clear separation between their syntactic and semantic knowledge.

Thus I'm interested in whether and to what extent models encode and utilize syntactic knowledge independent of specific vocabulary semantics. A recent approach that gets at this question is proposed by Papadimitriou and Jurafsky (2020), where they trained models on several non-linguistic corpora (L1) designed to capture various types of structural information, then finetuned and evaluated the models on a natural language dataset (L2). They found that models trained on structural L1s performed better than random baseline when evaluated zero-shot on the L2 Spanish, despite no overlap in vocabulary. This suggests that models were able to learn structure that generalized across different systems, independent of specific vocabulary semantics, and use that knowledge in natural language predictions. They also found performance differences between the various structural L1s, which raises the question of what led to those differences.

This current study adopts the same approach as Papadimitriou and Jurafsky (2020) and further explores the latter question of how different types of structural biases correspond to differences in performance. Their original L1s do not represent the structure of natural language, but rather various other nonlinguistic systems. Thus I introduce a new structural L1 that captures the distribution of syntactic categories in an actual natural language dataset, in order to test to what extent structural closeness to natural language and what types of structural abstractions contribute to effectiveness as an inductive bias.

4 Approach

4.1 Structural L1s

In order to capture natural language syntax, I designed and wrote code to create several corpora that abstracted different structural aspects of natural language syntax, and ultimately chose to use one that comprises the parts-of-speech (POS) tags of a language corpus. This POS corpus captures dependencies without losing information about the relationship between types of words and their distribution. Specifically, I use a neural parser to parse English language data. I then extracted the POS tags from the parsed dataset and reshaped them into the correct format to be loaded as a Hugging Face Dataset object (Lhoest et al., 2021), with each tag represented as an integer over a vocabulary of all possible POS tags. The vocabulary size of the corpus is 76, corresponding to the number of POS tags utilized by the parser.

I also re-implemented the Flat Parentheses and Nested Parentheses corpora from Papadimitriou and Jurafsky (2020) using existing code. The Flat Parentheses corpus consists of pairs of identical integers placed independently, thus allowing crossing dependencies. The Nested Parentheses corpus consists of pairs of identical integers nested hierarchically thus not allowing crossing dependencies, which is a constraint found in natural language. I created these two corpora with a vocabulary size of 100 and corpus size of around 102M tokens, to control for vocabulary and corpus size across the different corpora.

4.2 Baselines

Consistent with Papadimitriou and Jurafsky (2020), I trained my model on two random corpora as baselines. These corpora comprise integers sampled randomly, one from a Uniform distribution over the vocabulary (where each word is equally likely to be sampled) and one from a Zipfian distribution

(where common words are more likely to be sampled than others; this is taken to resemble the actual distribution of natural language words). Both the corpora have a vocabulary size of 76 and corpus size of around 102M. Across all synthetic corpora, the line length is fixed at 512.

I make use of two additional baselines, both of which are not trained on any synthetic corpora. I train a GPT-2 Small model from scratch during the finetune stage, thus it is randomly initialized and trained only for a small number of steps. I also use a pretrained GPT-2 Small model, which I finetune along with all the other models, thus forcing it to re-learn embeddings.

4.3 Model architecture

Different from the experiments in Papadimitriou and Jurafsky (2020), which used LSTM models, I probe the behavior of a transformer model. Specifically, I use the GPT-2 Small model architecture, which consists of 12 decoder transformer blocks and 124M parameters (Radford et al., 2019). Additionally, different from the original paper, where they froze parameter weights after the pretraining stage and only finetuned the embedding layer, I do not freeze the parameter weights, but instead simply let the models train for a small number of steps.

I trained all my models using code adapted from the Mistral codebase (Karamcheti et al., 2021)[1], which is built on Hugging Face models. I adapt finetuning code written by the authors of the original paper, shared privately.

5 Experiments

5.1 Data

For both extracting POS tags and finetuning models, I use wikitext-103 as hosted on Hugging Face, which comprises 103M English tokens extracted from Wikipedia articles (Merity et al., 2016). The POS corpus uses the "en_core_web_sm" parser provided by spaCy (Honnibal et al., 2020), which uses POS tags based on the Penn Treebank annotation scheme (Marcus et al., 1993).

5.2 Evaluation method

To evaluate my models, I first use the standard metric of perplexity (PPL), which is simply the exponent of the loss, calculated from the evaluation loss at the finetuning stage. A lower perplexity indicates that the model was able to perform well on the test set, which I take to reflect how well it learned English in the limited number of finetuning steps.

I also use the syntax challenge set provided by SyntaxGym (Gauthier et al., 2020)[2] as an additional targeted evaluation metric, which the original paper did not make use of. This is to compare model performance on syntax-specific tasks, and see if they perform differently relative to each other from the broad PPL metric. SyntaxGym consists of 33 test suites, where each test suite contains 20-80 minimal pairs designed to test knowledge of a specific grammatical construction. A sample minimal pair may include a grammatical and an ungrammatical sentence, and the model makes the correct prediction if it assigns higher probability to the grammatical sentence. I integrated this as a Hugging Face Metric into my evaluation code. I excluded six of the test suites that targeted garden path sentences, as those did not evaluate grammaticality but rather likeness to human processing effects.

5.3 Experimental details

I trained five GPT-2 Small models on the five different synthetic corpora (POS, Nested Parentheses, Flat Parentheses, Random Zipf, Random Uniform) as L1s for 10,000 steps. I then finetuned each of the trained models once with sampled embeddings, and once with pretrained embeddings. In the former condition, new embeddings were sampled from the old embeddings learned in the training stage. In the latter condition, the GPT pretrained embeddings were used. Thus I ended up with two finetuned models for each of the synthetic corpora.

[1] https://github.com/stanford-crfm/mistral

[2] https://syntaxgym.org/; https://huggingface.co/spaces/cpllab/syntaxgym

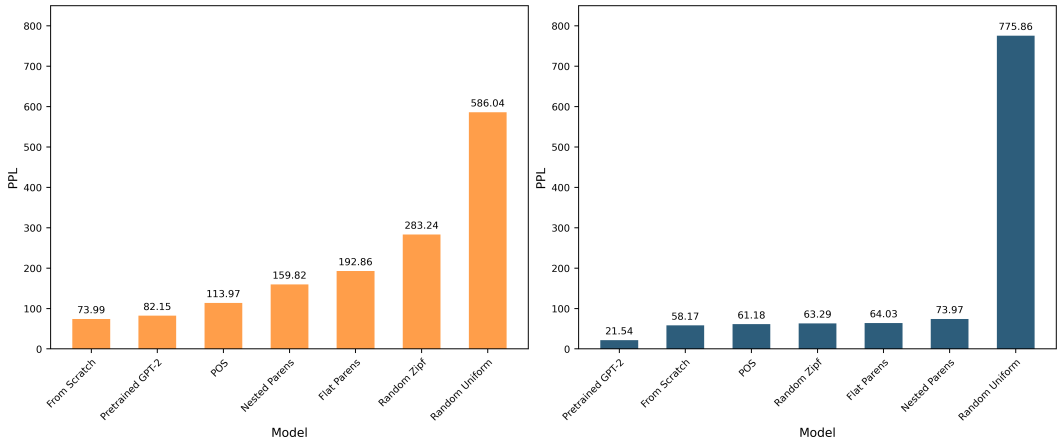
At the finetuning stage, I also introduced two new baselines: a GPT-2 Small pretrained model and a GPT-2 Small model trained from scratch. Both models were finetuned with the same settings as the other synthetic-corpora models. For the Pretrained GPT-2 Small model, this meant that the model had to relearn its embeddings. The From Scratch model was trained from scratch for only the number of finetuning steps. All models, including the synthetic-L1 models, were finetuned for 1,000 steps.

For training, I mostly used the default Mistral optimizer configurations, which uses the AdamW optimizer with a starting learning rate of 6e-7. I trained my models on the synthetic corpora with a device batch size of 14 and effective batch size of 518, and I finetuned with a batch size of 8. The synthetic-L1 models all trained for 1-2 days, and finetuning took around 10 hours to complete.

5.4 Results

For the synthetic-L1 models, I found the general expected gradation of perplexities that corresponds to the amount of structure in the synthetic corpus (see Figure 1a). The ordering of their perplexities (in the sampled embeddings condition) is as follows: POS < Nested Parentheses < Flat Parentheses < Random Zipf < Random Uniform. The differences between each of the perplexity scores is substantial. The two non-synthetic-trained baseline models (From Scratch and Pretrained GPT-2) both perform better than the other models.

For the models finetuned with pretrained embeddings (Figure 1b), we see a much smaller difference between the perplexities of the From Scratch, POS, Random Zipf, and Flat Parentheses models. On the two extremes, the Pretrained GPT-2 model performed significantly better, while the Random Uniform model performed significantly worse. The Nested Parentheses model, in contrary to the results with sampled embeddings, performed worse than both the Flat Parentheses and Random Zipf models, but still significantly better than the Random Uniform model.



(a) With sampled embeddings

(b) With pretrained embeddings

Figure 1: Perplexity scores on English test set. Lower perplexity indicates better performance.

All the models performed poorly (below chance) on the SyntaxGym test set, with similar overall accuracies between 0.2-0.3 for sampled embeddings, and a wider range between 0.18-0.68 for pretrained embeddings (Table 1).

Model	Accuracy (Pretrained)	Accuracy (Sampled)
GPT-2 Pretrained	0.680	0.296
Nested Parens	0.324	0.209
From Scratch	0.271	0.254
Flat Parens	0.263	0.238
POS	0.260	0.216
Random Zipf	0.258	0.263
Random Uniform	0.186	0.220

Table 1: Overall accuracy scores on SyntaxGym test set. An accuracy score of 1.000 would indicate the model made all correct predictions.

If we only consider the test suites where mean accuracy across all models was above 0.5 (Figure 2), we do see an ordering of accuracy scores that reflects the degree of structure in the model, albeit with small differences that might not be significant.

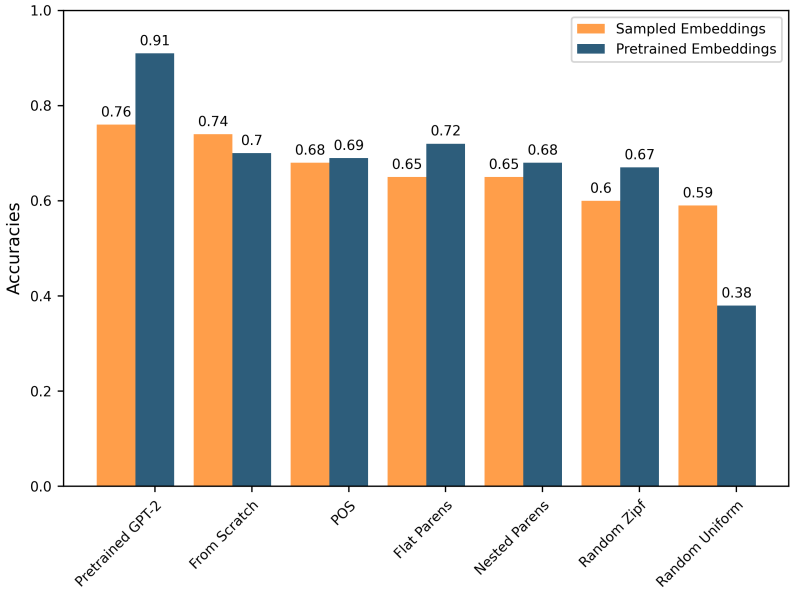


Figure 2: Overall accuracy scores on SyntaxGym test suites where mean accuracy across models was above 0.500. Six test suites fit this criterion.

6 Analysis

Replicating the results of the original paper, I found that models trained on any structure mostly outperformed the random baselines. This suggests that models are able to learn and utilize generalized structural knowledge independent of the semantics of an individual system, as there is no overlap in semantics or vocabulary between the nonlinguistic and linguistic corpora.[3]

Moreover, I showed that models are sensitive to finer-grained differences between different types of structure. Between the two random models, the Random Zipf model significantly outperformed Random Uniform in PPL and SyntaxGym accuracy for both the pretrained- and sampled-embeddings conditions. In fact, the Random Zipf results grouped closer to those of the structural models than of the Random Uniform model. This is contrary to what the original paper found, which was while the performance difference between Random Zipf and Random Uniform was significant, the overall performance for both those models was still much worse than other structural models. This difference

[3]One note of caution is that while I've shown models *can* learn generalizable structural knowledge independent of semantics, this doesn't imply that they actually do when training on natural language data.

can be attributed to many factors: the difference in model architecture (GPT-2 vs. LSTM), difference in vocabulary size (76 vs. 50,000), training configurations, or performance of the other models. Importantly, this result shows that models are sensitive to word distribution, even with no additional structural features.

The performance differences between the models trained on structural L1s were less consistent but still present. The POS model outperformed both parentheses models in PPL for both embedding conditions. I speculate that this is because the POS language captures richer structural information than either of the parentheses models. The dependencies between "words" in the POS corpus are subject to categorical differences, and each word can be consistently dependent on multiple words. For example, a verb is consistently distributed in a certain pattern (perhaps often between two nouns), and words like prepositions may form dependencies with both the verb and the following noun. This is the type of dependency we find in human language. Additionally, since the POS tags were directly extracted from an actual English dataset, the distribution of categories corresponds directly to the distribution in human language text. So the POS L1 is directly closer to human language in this implementational way. One baseline that could be included in future work is to sample the 76 vocabulary items with general consideration for word order, but not from a direct parse of language data, in order to tease apart the roles of the type of abstracted structure and the distribution that directly reflects human language text.

In the parentheses models, on the other hand, each word is only dependent on one other word in the corpus (namely, its closest identical twin). This representation is a very reduced abstraction of dependencies that only allows one "word" to be dependent on a single other word, and does not allow differentiation between categories of dependencies. Moreover, while the dependency lengths were sampled from a distribution of dependency lengths found in human language, the placement of the "parentheses" (i.e. integer pairs) relative to each other did not follow any language-based distribution, thus these parentheses L1s also do not capture information about how dependency lengths are distributed in a human language sample. There seemed to be no significant difference between the performance of the Nested Parentheses and Flat Parentheses model (which is consistent with the results in the original paper). This result may be due to several factors: Perhaps while models are sensitive to broad types of structure (distributional/categorical as in POS tags vs. binary dependencies), they simply aren't sensitive to the level of detail as to distinguish between different binary dependency structures. One other possibility is that both types of structures are sufficient for models to learn language to a certain degree. There are also implementational considerations: We cannot definitively claim that nested hierarchy doesn't matter, as the parentheses corpora may simply not adequately capture that type of structure.

Looking at the SyntaxGym accuracy scores, we also notice that the ranking of models based on perplexity is only roughly reflected in how well the models performed on the challenge set. The accuracies may be too low across-the-board to make definitive claims about the types of tasks the different models are good at. It is nevertheless illuminating that the models performed so poorly on the challenge set; just learning a small amount of language poorly seems to greatly hinder performance on difficult syntactic subtleties. I later discuss a possible reason for the poor SyntaxGym results.

While this project is mostly focused on how models trained on different structural corpora perform compared to each other and against random baselines, it is worth noting that all of the models pretrained on a synthetic L1 performed significantly worse than both the From Scratch and Pretrained GPT-2 models. This shows that for substantially different systems (perhaps especially ones with very different vocabulary sizes), training model parameters on one system is detrimental to transferring it to the other. The difficulty of transfer learning offsets any potential gains by general structural knowledge.

Wu et al. (2022) showed that the primary difficulty in transferring learning is learning new embeddings. This is illustrated in the performance difference of the Pretrained GPT-2 model between the two embedding conditions. When embeddings were sampled, the Pretrained GPT-2 model did similar to or worse than the From Scratch model, despite being trained on much more data. But when Pretrained GPT-2 used the pretrained embeddings, its performance skyrocketed (as most notably evidenced from its high accuracy on SyntaxGym metrics), presumably because the direct correspondence between its own embeddings and the new ones.

The difficulty in learning new embeddings may also help explain the poor performance of all models (excluding Pretrained GPT-2) on the SyntaxGym challenge set. Although designed to test syntactic

knowledge, most of the test suites are centered around very subtle grammatical differences that depend on rich knowledge of the relevant lexical items; i.e., good embeddings are a prerequisite. Thus the inability to learn the embedding layer well likely masked any possible difference between models' syntax abilities.[4]

7 Conclusion

I showed that models can learn and utilize generalized structure across different systems, and are sensitive to fine-grained differences in types of structure. Specifically, distributional structures that retain categorical dependencies are more effective inductive biases for language learning than binary dependency structures, possibly because they capture richer structural information and are closer to natural language syntax.

However, training on a different system with a completely different vocabulary in both items and size is detrimental to model performance and offsets the advantage gained by structural bias. The difficulty in learning new embeddings is a limitation of the transfer learning approach to probe certain questions, as the poorly learned embeddings masks possible differences in language behavior. It is still interesting that models are sensitive to different types of structures from an analytic perspective, but bolstering language syntax through targeted structure-training may not be practically desirable.

Future work should expand upon this approach and train models on a wider range of structural corpora and explore other training processes. For all of the current corpora, we should run trials with different vocabulary and corpus sizes to control for these effects. We should also create additional synthetic corpora to represent several broader types of structure (e.g. other ways to represent binary dependencies, or distributional categories), to find the relevant level of structural difference that models are sensitive to. In addition, future work should explore alternative training schemes or evaluation methods that can lessen the effect of poor embeddings.

References

Ethan A. Chi, John Hewitt, and Christopher D. Manning. 2020. Finding universal grammatical relations in multilingual BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5577, Online. Association for Computational Linguistics.

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*, 50 edition. The MIT Press.

Jon Gauthier, Jennifer Hu, Ethan Wilcox, Peng Qian, and Roger Levy. 2020. SyntaxGym: An online platform for targeted evaluation of language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 70–76, Online. Association for Computational Linguistics.

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.

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.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Siddharth Karamcheti, Laurel Orr, Jason Bolton, Tianyi Zhang, Karan Goel, Avanika Narayan, Rishi Bommasani, Deepak Narayanan, Tatsunori Hashimoto, Dan Jurafsky, Christopher D. Manning, Christopher Potts, Christopher Ré, and Percy Liang. 2021. Mistral - a journey towards reproducible language model training.

[4]Though it is still unclear to me why these models performed well below chance.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies.

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.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

Isabel Papadimitriou, Ethan A. Chi, Richard Futrell, and Kyle Mahowald. 2021. Deep subjecthood: Higher-order grammatical features in multilingual BERT. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2522–2532, Online. Association for Computational Linguistics.

Isabel Papadimitriou and Dan Jurafsky. 2020. Learning Music Helps You Read: Using transfer to study linguistic structure in language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6829–6839. ACL.

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

Zhengxuan Wu, Isabel Papadimitriou, and Alex Tamkin. 2022. Oolong: Investigating what makes crosslingual transfer hard with controlled studies. *CoRR*, abs/2202.12312.

Appendix

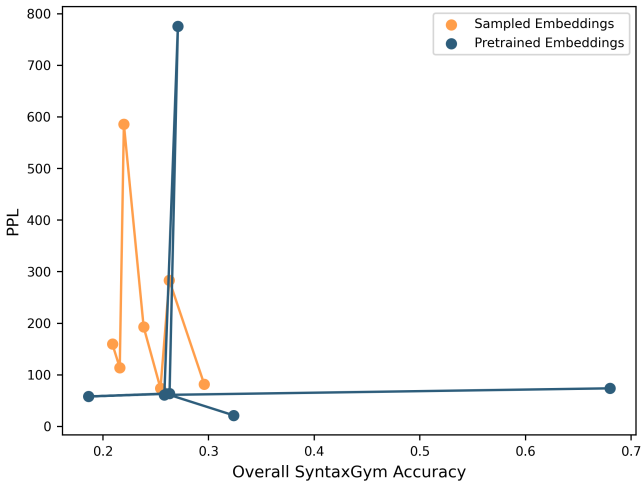


Figure 3: Model's SyntaxGym accuracy score (x-axis) against perplexity (y-axis). This graph is just a visualization of how the relationship between model perplexity and SyntaxGym accuracy is not direct. I think the differences in performance between models on the SynaxGym challenge set may largely be random or at least minimally informative.