Does Training an LLM on a Parsing Objective Improve Eye-Tracking Prediction?

Short Technical Report

Lukas Mielczarek[1], **Laura Kallmeyer**[1], **Hassan Sajjad**[2], **Katharina Spalek**[1]

[1]Heinrich Heine University Düsseldorf, [2]Dalhousie University, Halifax

[1]{lukas.mielczarek, laura.kallmeyer, katharina.spalek}@hhu.de

[2]hsajjad@dal.ca

September 2024

1 GPT-2 Finetuning on Syntactic Parsing

We are interested in whether incorporating syntactic knowledge in a large language model (LLM) makes it more human-like. We assume that the ability to predict eye-tracking metrics from some component(s) of the model is an indicator of similarity to human language processing. Thus, we calculate the error rates for eye-tracking prediction using an LLM pre-trained on next token prediction and a pre-trained LLM that was further fine-tuned on syntactic parsing.

The language model we opted for is GPT-2 (Radford et al., 2019). We use its final transformer output layer as a representation of each token. The fine-tuning task is syntactic parsing using the incremental attach-juxtapose parser of Yang & Deng (2020), with the strictly incremental implementation of Ezquerro et al. (2024). When the parser is trained, GPT-2's parameters are optimised as well. Eye-tracking metric prediction is based on the metrics and data provided as part of the CMCL 2021 shared task on eye-tracking prediction (Hollenstein et al., 2021). We take the per-token GPT-2 final representation and train a simple linear regressor using stochastic gradient descent. Training is done on the shared task's training set; early stopping is done using the development set. Results are also given for the development set. We take improvements in error rates for a linear regressor to signify a strong indication for overall better predictability. While one could possibly achieve better results by investigating non-linear correlations, we are only interested in relative changes in scores between our models of investigation here.

We explore using the representations generated by the stock pre-trained GPT-2 model (Vanilla), the fine-tuned GPT-2 model (Attach-Juxtapose) and a concatenation of the representations of both models (Both) for eye-tracking predictions. Furthermore, for this selection of models we explore three settings for predicting the

current word's eye-tracking metrics: using the current word's final layer (Table 1), using the previous word's final layer (Table 2) and using both the current and the previous word's final layer (Table 3).

2 Results

In tables 1 to 3 you can find the results of our experiment. Overall, results improve with the syntactically fine-tuned model (column 2), in particular when combining it with the vanilla model (column 3). This means that fine-tuning towards incremental parsing adds information relevant to predicting eye-tracking data that is not already fully covered in the vanilla GPT2 model.

When using both the current word final layer and the previous word final layer (Table 3), improvements over the vanilla parser are smaller than when taking only the current word (Table 1). This may have the following reason: The Attach-Juxtapose parser is trained to predict for each token the correct parsing action and is given the final layers up to the current token. Thus, when finetuning the GPT-2 model in the parser training process, the information stored in the activation for the current token is shifted from information about the next token (GPT-2 objective: next token prediction) to information about the current token. Thus, using this finetuned GPT-2 model for eye tracking metric prediction of the current word works better. This advantage is lost when including the previous word's output layer.

For this reason, attach-juxtapose finetuning makes eye tracking predictions worse when taking only the preceding word (Table 2). The preceding word's parsing action is not informative for the current word's eye tracking metrics.

3 Future Work

Given the discussion above, one could condition the preceding word's hidden representation on parsing action prediction for the following word (i.e. predicting where to insert the word that follows as well as the non-terminal to insert). This would be akin to the exploration of syntactic surprisal in LLMs by Arehalli et al. (2022) who predict the following token's supertag, based on the assumption that human processing difficulty is guided by the predictability of the input stream. They find that syntactic surprisal improves but does not fully explain human processing difficulties observed for garden path sentences. We believe that one reason for this might be the lack of an explicit incremental structure modelling component such as a parser to act as a model of working memory and to complement expectation-based surprisal. Enriching LLMs both with speculation about future structure and with the incorporation of incoming words into the structure built so far should make their activations informative both in terms of prediction and structure processing.

However, such an experiment is not trivially possible with the attach-juxtapose parser implementation used in this experiment since it applies the attention mechanism to find the best element in the right fringe that the current word should attach/juxtapose to. When speculating about the next word's syntactic structure in relation to the partial tree built so far, its representation vector is not available to calculate attention since it has not been seen yet. One would need to devise a mechanism to find an attach/juxtaposition point (and optional non-terminals) with only the current partial tree as the input.

References

Arehalli, S., B. Dillon & T. Linzen. 2022. Syntactic Surprisal From Neural Models Predicts, But Underestimates, Human Processing Difficulty From Syntactic Ambiguities. In A. Fokkens & V. Srikumar (eds.), *Proceedings of the 26th Conference on Computational Natural Language Learning (CoNLL)*, 301–313. Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics. doi: 10.18653/v1/2022.conll-1.20. https://aclanthology.org/2022.conll-1.20.

Ezquerro, A., C. Gómez-Rodríguez & D. Vilares. 2024. From Partial to Strictly Incremental Constituent Parsing. In Y. Graham & M. Purver (eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, 225–233. St. Julian's, Malta: Association for Computational Linguistics. https://aclanthology.org/2024.eacl-short.21.

Hollenstein, N., E. Chersoni, C. L. Jacobs, Y. Oseki, L. Prévot & E. Santus. 2021. CMCL 2021 Shared Task on Eye-Tracking Prediction. In E. Chersoni, N. Hollenstein, C. Jacobs, Y. Oseki, L. Prévot & E. Santus (eds.), *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, 72–78. Online: Association for Computational Linguistics. doi:10.18653/v1/2021.cmcl-1.7. https://aclanthology.org/2021.cmcl-1.7.

Radford, A., J. Wu, R. Child, D. Luan, D. Amodei & I. Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

Yang, K. & J. Deng. 2020. Strongly Incremental Constituency Parsing with Graph Neural Networks. https://arxiv.org/abs/2010.14568.

Metric	Vanilla	Attach-Juxtapose	Both
Number of fixations (nFix)	4.7	4.4	4.3
First fixation duration (FFD)	0.72	0.68	0.67
Total reading time (TRT)	1.8	1.7	1.7
Go-past time (GPT)	2.8	2.6	2.6
Fixation Proportion (FixProp)	19	20	19

Table 1: Mean absolute error for taking current word GPT-2 final layer as input for a linear classifier.

Metric	Vanilla	Attach-Juxtapose	Both
Number of fixations (nFix)	6.7	6.8	6.6
First fixation duration (FFD)	1.0	1.1	1.0
Total reading time (TRT)	2.5	2.6	2.6
Go-past time (GPT)	3.4	3.5	3.4
Fixation Proportion (FixProp)	19	20	19

Table 2: Mean absolute error for taking preceding word GPT-2 final layer as input for a linear classifier.

Metric	Vanilla	Attach-Juxtapose	Both
Number of fixations (nFix)	4.4	4.4	4.2
First fixation duration (FFD)	0.7	0.68	0.66
Total reading time (TRT)	1.7	1.7	1.6
Go-past time (GPT)	2.8	2.6	2.6
Fixation Proportion (FixProp)	12	12	12

Table 3: Mean absolute error for taking current and preceding word GPT-2 final layer as input for a linear classifier.