6.8610 Project Proposal

Kimberly Llajaruna, Emilia Mazzolenis, Michael Sam, Clara Ye

November 2023

Motivation

Entity tracking is an important task in NLP that involves identifying and connecting named entities in text to specific entries in a knowledge base or reference database.

Models pre-trained using both text and code have been found to demonstrate non-trivial entity-tracking behavior under a two-shot setting, which models pre-trained on text only demonstrate after fine-tuning (Kim and Schuster, 2023). However, one limitation of this body of research is the potential influence of model architecture disparities. The models under examination, including variations of GPT-3, GPT-3.5, and Flan-T5, not only differ in their training data but also in their underlying architectural designs. This raises the possibility that the observed variations in entity-tracking capabilities may be attributed to model design as much as, or even more than, the training data alone.

Therefore, we propose a more rigorous way to test the effect of pre-training models with code. Moreover, we expand on the original paper's proposal by evaluating the effect of pre-training using mathematical reasoning and comparing it to only factual information and no reasoning at all. This knowledge not only advances our comprehension of NLP models but also has practical implications for downstream tasks. It can inform the design of more effective models and enhance their performance across various applications.

Related Works

We based our research question and proposed methods on the literature that addressed similar topics, the most relevant of which can be found below.

There has been plenty of effort on pre-training models on code. For instance, CodeBERT (Feng et al., 2020) is an encoder-only model trained for natural language code search and code documentation generation, and CodeT5 (Wang et al., 2021) is an encoder-decoder transformer model for code defect detection, clone detection, and code generation. However, none of the models have been applied to entity tracking or even natural language reasoning tasks in general. In addition, these models also have distinct architecture which, as in our motivating paper, prevents us from isolating the effect of training data from model design. Therefore, our project would aim to fill the gap in literature regarding the relationship between pre-training on code and entity-tracking ability.

Why is it important to investigate the relationship be

Another related track of work centers around natural language reasoning. The authors of the paper "Chain of Thought Prompting Elicits Reasoning in Large Language Models" (Wei et al., 2022) have found that including intermediate reasoning steps via chain of thought prompting enables LLMs to engage in complex reasoning. Given the potential benefits of reasoning in enhancing our entity tracking task, particularly in the context of monitoring how attributes or relationships evolve over time, we will adopt a similar approach. In our model, we are integrating reasoning through two distinct avenues: mathematical reasoning, involving problems with mathematical questions and comprehensive step-by-step answers, and computational reasoning, which encompasses problems with coding-related questions and well-defined, step-by-step coding solutions.

Are you expecting code LMs will be more powerful for

A paper that has worked with similar transformer models on the topic of entity recognition is "When a sentence does not introduce a discourse entity, Transformer-based models still sometimes refer to it" (Schuster and Linzen, 2022). In this work, the authors adapted the psycho-linguistic assessment of language models paradigm to higher-level linguistic phenomena and introduced an English evaluation suite that targets the knowledge of the interactions between sentential operators and indefinite NPs. They used this evaluation suite for a fine-grained investigation of the entity tracking abilities of the Transformer-based models GPT-2 and GPT-3 where they concluded that they did not see the entity tracking ability. This work is highly relevant as it not only uses a model like GPT, but generalizes to transformer models, which is entitled to the BERT

model that we are planning to use. Moreover, such an approach would also allow us to test if the lack of entity tracking ability showed in this paper replicates to our models with continual pre-training using code and mathematical reasoning.

Measures for Success

We would use the same benchmark dataset as proposed by Kim and Schuster (2023) to evaluate the entity-tracking performance of the models. Specifically, the model is prompted with a general instruction to the task, two demonstration examples, a description of the initial state and subsequent operations, and an incomplete sentence for the model to fill in the final state. We then compute the following metrics that compare model output to correct answers:

- **Accuracy** calculates the ratio of all correct predictions to the total entities tracked.
- **Entity precision** calculates the ratio of correctly tracked entities to the total entities predicted by the model.

Dataset

Our approach involves utilizing three distinct datasets, each designed to capture different facets of knowledge and skills:

- General Knowledge Fact-Based Question-Answer Pairs: To develop a comprehensive understanding of general knowledge without explicitly including reasoning capabilities in the examples, one version of our model undergoes continual pre-training using a dataset consisting of question-answer pairs. This dataset predominantly contains factual information on a wide range of topics, with a focus on facts rather than reasoning. This dataset will allow us to get a model that acts as a baseline continual pre-training, where the structure of the data is the same as in the other two datasets below, but no reasoning is included.
- LeetCode Code and Solution Dataset: Another version of our model undergoes continual pre-training using a dataset derived from LeetCode. This dataset features coding problems and their solutions, designed to facilitate code understanding. Problem prompts are provided as comments at the top of the code, with corresponding Java solutions appended below. This approach equips our model with proficiency in comprehending and generating code-related content.
- Mathematical Question-Answer and Latex Dataset: The third version of our model undergoes continual pre-training using a dataset that encompasses question-answer pairs related to mathematical topics. This dataset includes mathematical questions and their respective natural language explanations, mathematical facts, and uses LaTeX to represent mathematical expressions. This combination enriches our model's capabilities in handling mathematical and scientific content, including mathematical reasoning.

Ideas for Solving the Problem / Summary

We are planning to use DistillBERT (Sanh et al., 2020) as our base models and train different variations, each using one of text data, code data, or math data. For each variant, we would use two objectives: masked language modeling (MLM), and text-to-text (T2T) matching. In T2T matching, the model would take a batch of problem-solution pairs as input, compute similarity scores for all possible pairs, and is optimized using a contrastive loss that encourages higher similarity for matching pairs and lower similarity for non-matching pairs.

We would compare the performance of the three models on the benchmark as described in Measures for Success. Using the model trained on text question-answer data as a baseline would control for the difference in model architecture and isolate the effect pre-training using code or math on entity-tracking ability.

We expect to use PyTorch to implement our models. We have access to GPUs as offered in Colab Pro.

References

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547, Online. Association for Computational Linguistics.

Najoung Kim and Sebastian Schuster. 2023. Entity tracking in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3835–3855, Toronto, Canada. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Sebastian Schuster and Tal Linzen. 2022. When a sentence does not introduce a discourse entity, transformer-based models still sometimes refer to it. *ArXiv*, abs/2205.03472.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. 2021. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8696–8708, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.