"Can Large Language Models Write Code?" (2022) by Chen, et al

https://www.sciencedirect.com/science/article/pii/S2666920X2300019X

Generative Large Language Models (LLMs) demonstrate impressive results in different writing tasks and have already attracted much attention from researchers and practitioners. However, there is limited research to investigate the capability of generative LLMs for reflective writing. To this end, in the present study, we have extensively reviewed the existing literature and selected 9 representative prompting strategies for ChatGPT – the chatbot based on state-of-art generative LLMs to generate a diverse set of reflective responses, which are combined with student-written reflections. Next, those responses were evaluated by experienced teaching staff following a theory-aligned assessment rubric that was designed to evaluate student-generated reflections in several university-level pharmacy courses. Furthermore, we explored the extent to which Deep Learning classification methods can be utilised to automatically differentiate between reflective responses written by students vs. reflective responses generated by ChatGPT. To this end, we harnessed BERT, a state-of-art Deep Learning classifier, and compared the performance of this classifier to the performance of human evaluators and the AI content detector by OpenAI. Following our extensive experimentation, we found that (i) ChatGPT may be capable of generating high-quality reflective responses in writing assignments administered across different pharmacy courses, (ii) the quality of automatically generated reflective responses was higher in all six assessment criteria than the quality of student-written reflections; and (iii) a domain-specific BERT-based classifier could effectively differentiate between student-written and ChatGPT-generated reflections, greatly surpassing (up to 38% higher across four accuracy metrics) the classification performed by experienced teaching staff and general-domain classifier, even in cases where the testing prompts were not known at the time of model training.

Language Models of Code are Few-Shot Commonsense Learners

https://aclanthology.org/2022.emnlp-main.90.pdf

We address the general task of structured commonsense reasoning: given a natural language input, the goal is to generate a graph such as an event or a reasoning-graph. To employ large language models (LMs) for this task, existing approaches "serialize" the output graph as a flat list of nodes and edges. Although feasible, these serialized graphs strongly deviate from the natural language corpora that LMs were pre-trained on, hindering LMs from generating them correctly. In this paper, we show that when we instead frame structured commonsense reasoning tasks as code generation tasks, pre-trained LMs of code are better structured commonsense reasoners than LMs of natural language, even when the downstream task does not involve source code at all. We demonstrate our approach across three diverse structured commonsense reasoning tasks. In all these natural language tasks, we show that using our approach, a code generation LM (CODEX) outperforms natural-LMs that are fine-tuned on the target task (e.g., T5) and other strong LMs such as GPT-3 in the few-shot setting. Our code and data are available at https: //github.com/madaan/CoCoGen .

CODEBERTSCORE: EVALUATING CODE GENERATION WITH PRETRAINED MODELS OF CODE

https://arxiv.org/pdf/2302.05527.pdf

Since the rise of neural models of code that can generate long expressions and statements rather than a single next-token, one of the major problems has been reliably evaluating their generated output. In this paper, we propose CodeBERTScore: an automatic evaluation metric for code generation, which builds on BERTScore (Zhang et al., 2020). Instead of measuring exact token matching as BLEU, CodeBERTScore computes a soft similarity score between each token in the generated code and in the reference code, using the contextual encodings of large pretrained models. Further, instead of encoding only the generated tokens as in BERTScore, CodeBERTScore also encodes the programmatic context surrounding the generated code. We perform an extensive evaluation of CodeBERTScore across four programming languages. We find that CodeBERTScore achieves a higher correlation with human preference and with functional correctness than all existing metrics. That is, generated code that receives a higher score by CodeBERTScore is more likely to be preferred by humans, as well as to function correctly when executed. Finally, while CodeBERTScore can be used with a multilingual CodeBERT as its base model, we release five language-specific pretrained models to use with our publicly available code at https://github.com/neulab/code-bert-score. Our language-specific models have been downloaded more than 25,000 times from the Huggingface Hub.