Chapter I.

Introduction

The NASA Astrophysics Data System (ADS) is the primary tool that researchers in the fields of astronomy and astrophysics use to locate scholarly literature. It processes tens of thousands of user requests per day and is also commonly used among researchers and students outside astrophysics domains because it indexes the entire arXiv. ADS provides more powerful tools for searching and filtering that what is available on other academic search engines like Google Scholar.

Requests can be highly expressive and precise because of the complex search and filtering logic supported by Apache Solr, the database technology underlying ADS. This comes with a tradeoff: the system may be more difficult to utilize, and users must learn the syntax of Solr as well as read the ADS documentation to understand the meanings of the > 50 search fields to take full advantage of available features.

This thesis proposes to develop an alternative, simpler to use, search method: interacting with the ADS database through requests made in natural language. We will build an application leveraging the power of transformer-based large language models (LLMs). The LLM will be used to perform the natural language processing

(NLP) task of translating a user's natural language request into a Solr query that, when executed, returns the results they were looking for.

To date, much work has been done on translating natural language requests into the Structured Query Language (SQL). This is to be expected because SQL and its variants are the standard query languages for relational databases. The problem of translating natural language requests into query languages like Solr, which queries document databases, remains neglected. Creating a system capable of this task is one of the primary contributions of this project.

1.1. Project Goals

The research questions associated with this thesis are as follows:

(i) Can LLMs consistently translate natural language search queries into ADS-specific Solr queries that capture the original intent of the user?

(ii) Does providing the LLM with examples of the translation task (few-shot learning), improve the translation task?

(iii) Is the translation performance further improved when the examples are chosen to be similar to the natural language request made by the user?

(iv) Will a synthetic dataset produced via backtranslation of human-generated Solr queries be of sufficiently high quality to significantly improve the performance of the forward translation task?

Chapter II.

Background

2.1. Language Models

Modern approaches to the problem of natural language to structured query language translation rely on large language models (Katsogiannis-Meimarakis & Koutrika, 2023). These are statistical models that aim to capture the syntactic and semantic relationships encoded in sequences of text. Formally, a generative language model can be described as the process of autoregressively sampling from the distribution:

$$P(x_{i+1} \mid x_0 \cdots x_i)$$

where $x_0 \cdots x_i$ represents the concatenation of the starting sequence and any tokens generated up to time step i and x_{i+1} is the next token to generate.

In essence, a language model is a distribution that assigns probabilities to sequences of tokens, where tokens are chunks of text, usually smaller than words. So a common, grammatical English sentence (eg. "Hi, how are you?") has many orders of magnitude higher probability of appearing in written English text than a

meaningless, ungrammatical sequence of tokens (eg. "alkdf kdfj aier iejrn kd").

Given the abstract formalization of a language model as probabilities over sequences of tokens, there are many mathematical models that could be language models. The simplest language model is the n-gram model (Jurafsky & Martin, 2009). To fit a 3-gram (trigram) model, for example, all instances of 3 tokens adjacent to one another within a corpus of text would be counted. These frequencies could be normalized, which would then give a probability mass function over the set of all unique sequences of 3 tokens that occurred in the text.

This PMF could then be used in a generative setting were the goal was to do next token prediction. To finish the sentence (assuming tokens are words): "The weather outside is beautiful today. I am feeling _____", one could sample from the conditional distribution $P(x \mid \text{am feeling})$. Notice that while a model of this type trained on a large corpus of English text would likely sample an emotion word like "happy", "sad", or "excited", it fails to capture long term dependencies between the words. That is, $P(x \mid \text{The weather outside is beautiful today. I am feeling}) = P(x \mid \text{am feeling})$ using a trigram model. This naive model does not incorporate the information about the weather because the probability of the next token is only conditioned on the previous two tokens.

Modeling language is a requirement for doing many useful tasks involving language, such as translation, sentiment analysis, and image captioning. These examples correspond to the three functional classes of language models: sequence-to-sequence,

sequence-to-vector, and vector-to-sequence. We are most concerned with a task like language translation. For example, a language model may be trained on a dataset consisting of sentences in Spanish alongside their translation into English. During the training process, the model should learn about the structure of the Spanish and English languages and the features that relate Spanish and English sentences. Given a new Spanish sentence outside the training set, it should be able to successfully translate it into English.

Of course, this type of sequence-to-sequence translation task is exactly what the system developed for this thesis aims to achieve. For our task, we want a model that can take a search request for a scholarly paper expressed in English, and translate that into a sequence of tokens that make up a well-formed Solr query that captures the semantic intent of the English-language request.

2.2. Transformers

Today, much more sophisticated models built on neural networks, like the RNN or LSTM, are used to model language and generate text. However, the most widely used LLM architecture by far is the transformer. First introduced in the 2017 paper "Attention is All You Need", transformer models are now the foundation of services like ChatGPT, Gemini, and Claude.

The basic architecture of the transformer consists of alternating attention and MLP layers. Inside any attention layer, there are multiple attention heads, each

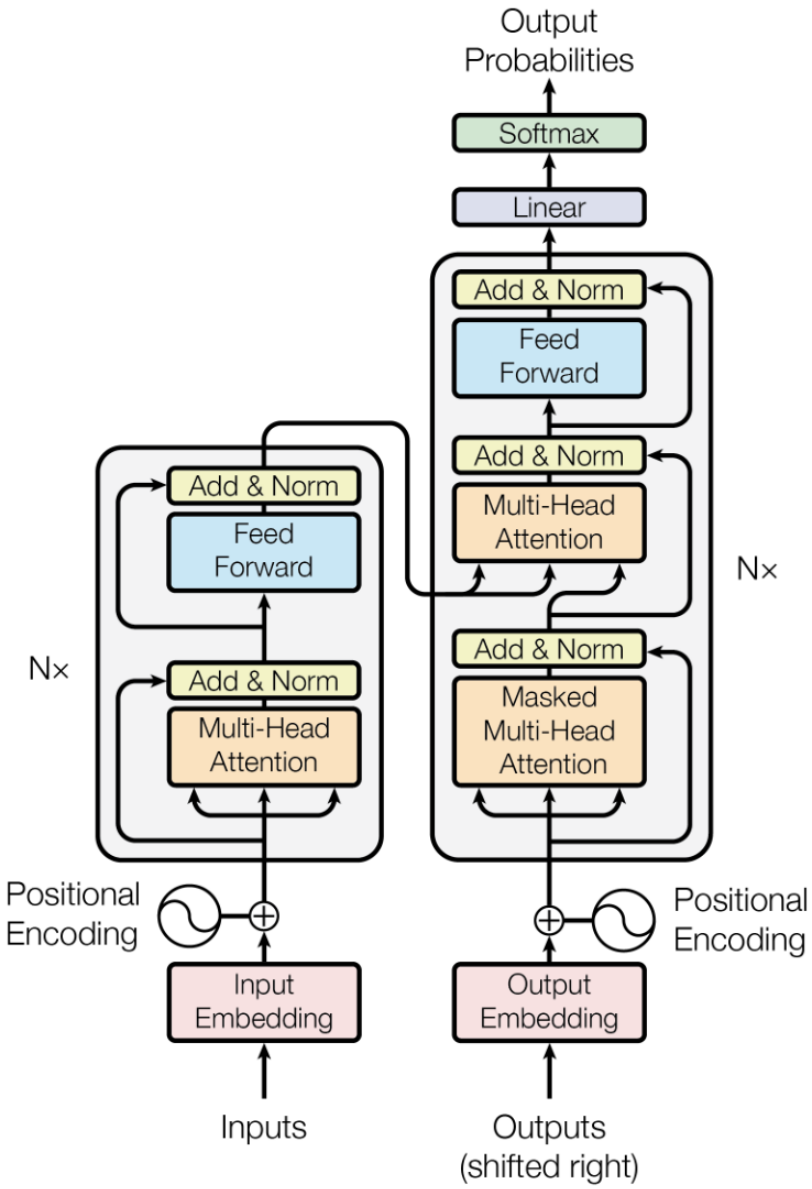

Figure 1: Architecture of a simple transformer. Reproduced from Attention is All You Need (Vaswani et al., 2017)

with their own attention pattern. The primary contribution of the transformer paper was the introduction of the self-attention mechanism. For each token in the input sequence, the attention head calculates a probability distribution over prior tokens that weights how much information to copy between token positions in the transformer residual stream. The head then moves information between token sequence positions according to the attention pattern (Elhage et al., 2021). This system allows the model to capture the complex dependencies and relationships between tokens more effectively.

2.3. Supervised Fine-Tuning

Unfortunately, even after being trained on trillions of tokens of text, frontier LLMs are not immediately capable of being helpful AI assistants like ChatGPT that answer questions, retrieve information, write complex code, and more. In their first stage of training, known as pretraining, the model objective is next token prediction. So if a sequence like "Explain the moon landing to a 6 year old in a few sentences" is fed into a GPT-4 immediately after pretraining, it will not output a helpful reply (Ouyang et al., 2022). Rather, the most likely completion would start with "Explain the theory of gravity to a 6 year old".

This happens because the pretrained language model has not been directly optimized to perform well on the desired task. The capabilities for answering questions, doing coding, etc. are there, but it is extremely difficult to find a prompt that

is able to elicit these capabilities from the model (Ouyang et al., 2022). In order to make useful capabilities easier to elicit, models undergo a second stage of training, instruction tuning, which is a form of supervised fine-tuning. This form of training is supervised because it involves labeled data. The model is optimized with respect to a loss function that compares the model's response to a (typically human written) ground-truth response.

Even fine-tuning on extremely small datasets (compared to the size of the training dataset) has been shown to dramatically change model performance. For alignment fine-tuning (ie. making the model helpful, honest, and harmless) researchers at OpenAI showed that fine-tuning on a 120KB dataset of Q&A examples, which was roughly 0.000000211% of GPT-3 training data was able to dramatically decrease harmful outputs and improve scores on benchmarks measuring toxicity (Solaiman & Dennison, 2021).

So users of commercially available and open-source LLMs don't have to train a model to understand language from scratch. They can start with a pretrained model and fine tune it specifically for their application. Although supervised fine-tuning can be extremely effective for many tasks, it was not the fine-tuning method chosen for this thesis.

2.4. In-context Learning

Supervised fine-tuning is not the only way to get a model to perform better on a particular task. Task-specific performance of a language model can be improved by including examples of the task when prompting the model (Brown et al., 2020). This approach is known as in-context learning (ICL) or few-shot learning. In the seminal paper Language Models are Few-Shot Learners, Brown et al. show that ICL significantly increased the performance of GPT-3 on most standard language benchmarks like MMLU. On some tasks, the performance of ICL was on par with the supervised fine-tuning (SFT) approaches.

The primary advantage of ICL over SFT is that the model weights remain fixed. Since the examples are included in the context during inference, no more training of the model needs to be done to improve performance. This is advantageous because training is compute-intensive and therefore expensive. While including some examples in the context increases the amount of compute that must be done at inference time, this is generally much less than what is required with SFT.

Brown et al. also found that larger models are able to make better use of in-context examples. That is, as the parameter size of the model increases, the gap between the zero-shot and few-shot performance also grows (Brown et al., 2020). The standard interpretation of this phenomenon is that as models get larger and larger, and more and more compute goes into training them, their capabilities increase significantly. As LLMs become more capable, there is a meaningful sense in which

they already know how to do the desired task immediately after pretraining. For larger models, the examples serve less as a form of training to teach the model how to do the task, but rather to elicit the model's preexisting capabilities that are useful for solving the task.

The observation that ICL works better on larger models motivated this project's focus on ICL over SFT. Next generation models are being trained with ever-more compute and it can be easy to swap one model out for another. Using ICL instead of fine tuning allows for the flexibility to switch to more capable models when they are released (Chapter 4.1). That is, no additional training needs to be done to use GPT-4 to perform better than GPT-3 on the ADS translation task.

Although in-context learning for the text-to-SQL task has proved to be very effective, there are still open questions related to the construction of the context and its effect on performance. For example, SOTA language models are often able to have every large context windows (eg. 4k - 100k tokens). It might therefore seem ideal to take advantage of these large windows by including as many examples in the context as possible. However, recent research shows that the position of examples in the context can strongly influence overall performance. Several SOTA models were shown to perform best on a question-answering task when the most relevant examples appeared at the beginning or at the end of the context (Liu et al., 2023).

2.5. Retrieval-augmented Generation

State-of-the-art LLMs are highly capable and possess enormous amount of knowledge. All model capabilities and knowledge are ultimately the result of training on the task of next token prediction. In other words, one way in which an LLM "knows" the capital of France is Paris is because that fact is extremely useful for predicting tokens in some sentences, like "The capital of France is _____". It is of course the case then, that a model's factual knowledge is limited to what it has seen in the training set.

However, having the training set as the only source of knowledge comes with some drawbacks:

(i) While LLMs are able to memorize large portions of their training sets (Carlini et al., 2022), they are not able to learn every fact encoded in the training distribution.

(ii) A model will be unaware of facts and events that take place after it is deployed—unless the model is retrained on new data.

(iii) Users may have private collections of documents that they would like to do useful NLP tasks on like summarizing, sentiment analysis, etc.

Retrieval-augmented generation (RAG) was developed to overcome these limitations. RAG allows LLMs to retrieve and use information from a large external

knowledge store that may be updated in real time (Lewis et al., 2020). RAG proceeds by retrieving the documents from the store that are most relevant to the user's input. Retrieval is usually done using a dense vector search on embeddings of all the chunks of the documents in the vector database. The text from the retrieved document is then inserted into the context along with the original input query, thus incorporating the document information into the generation process.

2.6. Related Work

2.6.1 Text-to-SQL

In recent years, deep learning methods have been applied to the text-to-SQL task. Historically, recurrent models like LSTMs were most common (Qin et al., 2022). Since the widespread growth of transformers, more and more of the top performing text-to-SQL models are transformer-based. Moreover, as transformer models like GPT have continued getting larger, in-context learning methods, as opposed to supervised fine-tuning approaches are becoming the norm. The Spider benchmark is one of the primary evaluations that measures performance on the text-to-SQL task. It consists of 5,693 unique complex SQL queries and associated questions from 138 different domains (Yu et al., 2019). As of March 2024, 8/10 of the top performers on the Spider evaluation leaderboard were systems involving transformers utilizing in-context learning.