Doğal Dilden Veritabanı Sorgusuna: Büyük Dil Modelleri ve Güncel Gelişmeler

Prof. Dr. Pınar Karagöz & Prof. Dr. İsmail Hakkı Toroslu
Recep Fırat Çekinel & Aslı Umay Öztürk
Orta Doğu Teknik Üniversitesi

22 Aralık 2023
Çevrimiçi

Program

- Problem tanımı: “text-to-SQL”
- Literatürdeki yaklaşımlar
  - Kural tabanlı yöntemler (Rule-based methods)
  - Derin öğrenme yöntemleri (Deep learning methods)
  - Büyük dil modeli tabanlı yöntemler (LLM-based methods)
- Büyük dil modelleri (Large Language Models - LLMs)

- Büyük dil modelinden verim almak: “Prompt Engineering”
- Prompt Engineering - Demo
- Büyük dil modellerinde ince-ayar: “Fine-tuning LLMs”
- Fine-tuning LLMs - Demo

Not: Sunum Türkçe anlatılacak, fakat slaytlar İngilizce ilerleyecektir.

Problem Statement

- In general, database management systems, serving as an important resource for users to make decision in many fields, such as health care, sports, and entertainment, has emerged frequently because of the big data era.
- Despite the effectiveness and efficiency of using DBMSs, learning query languages causes a barrier for non-technical users.
- The task of text-to-query aims to convert natural language questions into executable DB queries

Example

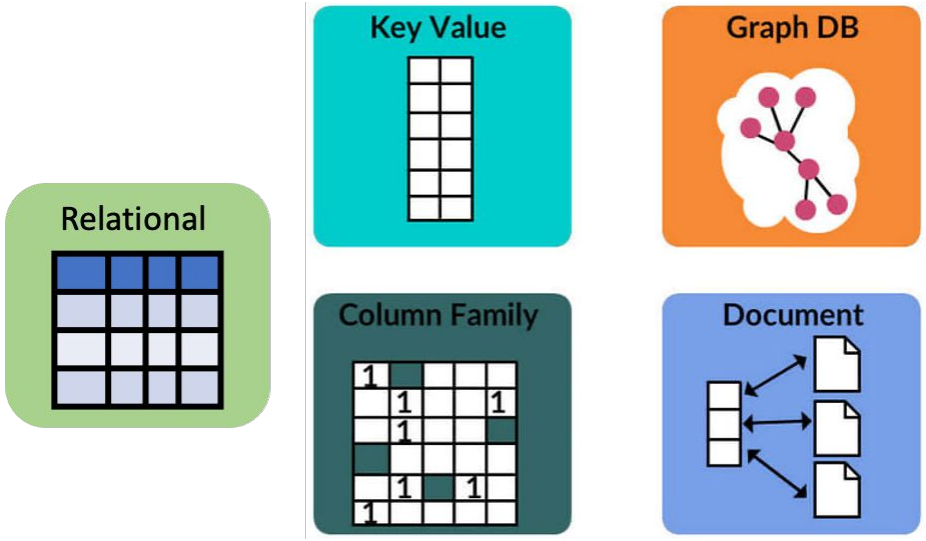
Natural Language Query:

1234 numaralı öğrencinin
aldığı dersler ve notları listele.

Example

Natural Language Query:

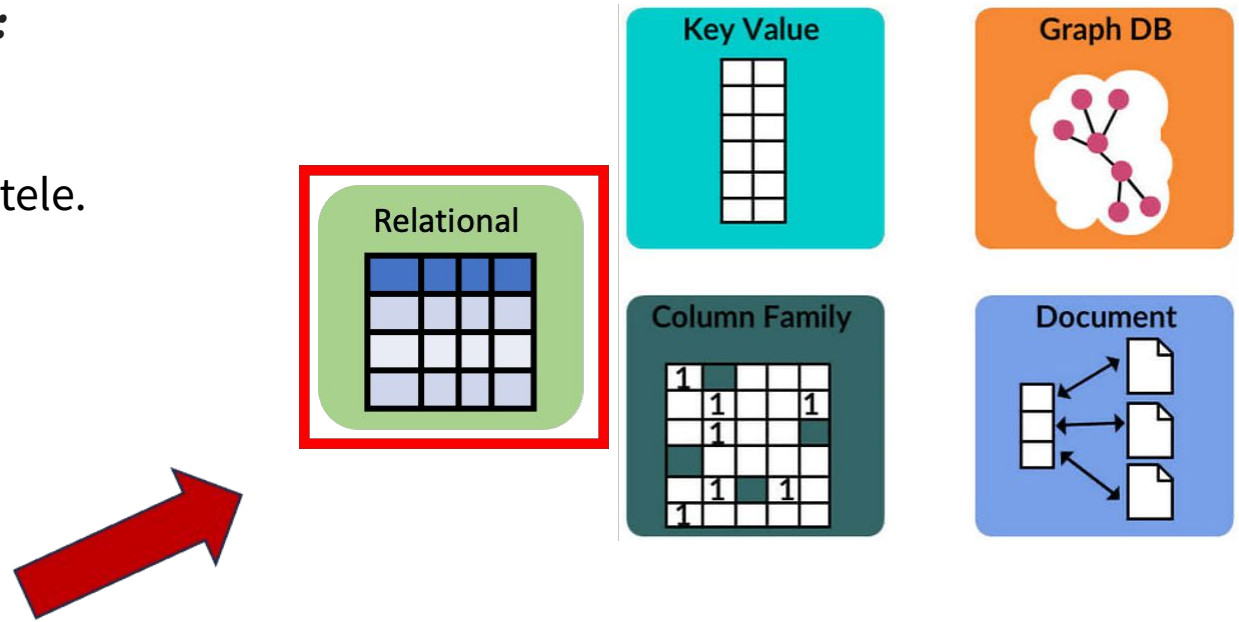
1234 numaralı öğrencinin aldığı dersleri ve notları listele.



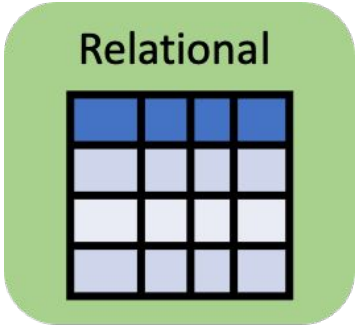
Example

Natural Language Query:

1234 numaralı öğrencinin aldığı dersler ve notları listele.



Example



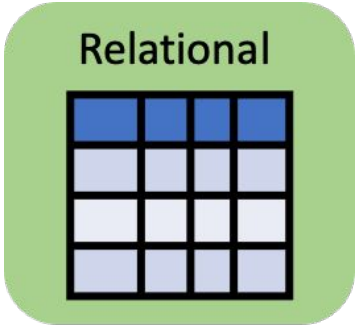
Natural Language Query:

1234 numaralı öğrencinin aldığı dersler ve notları listele.

Derse_Kayıt(ogrenciID, dersID, puan)

```
SELECT *
FROM Derse_Kayıt DK
WHERE DK.ogrenciID = 1234;
```

Example



Natural Language Query:

CENG bölümündeki her bir ders için dersID'yi ve derse kayıtlı öğrenci sayısını listele.

Ders(dersID, dersAd, bolum)

Derse_Kayıt(ogrenciID, dersID, puan)

```
SELECT D.dersID, count(*)
FROM Ders D INNER JOIN Derse_Kayıt DK on D.dersID=DK.dersID
WHERE D.bolum = 'CENG'
GROUP BY D.dersID;
```

Problem Statement

- The task of text-to-***SQL*** aims to convert natural language questions into executable ***SQL*** queries

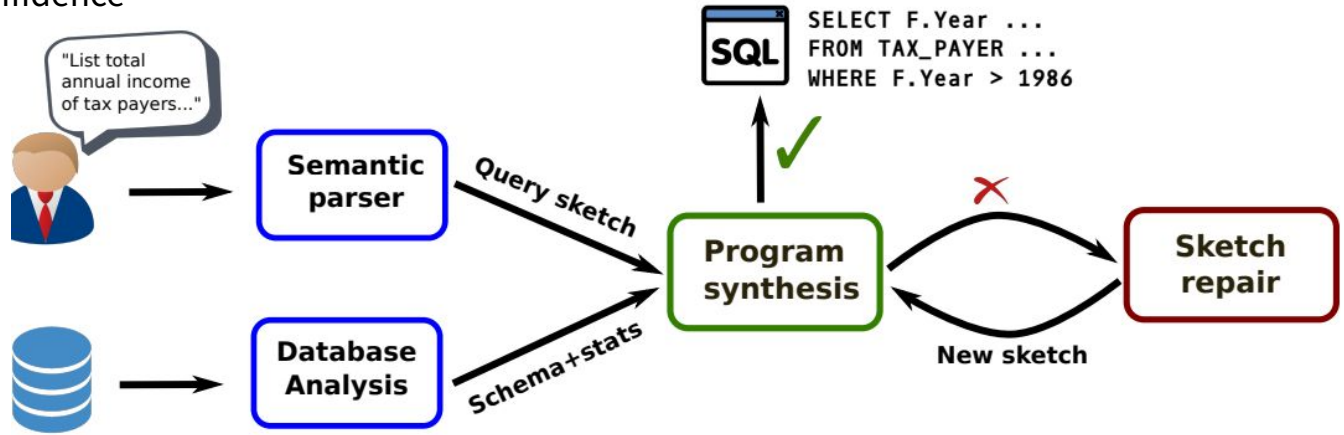
- Challenges
  - Complex queries involving Join, Aggregation etc.
  - Recognizing correct values, synonyms
  - Domain generalization, i.e. , how to generalize well to unseen databases.

Basic Approaches in the Literature

- Rule-based methods
- Deep learning based methods
- LLM-based methods

Rule-based Methods

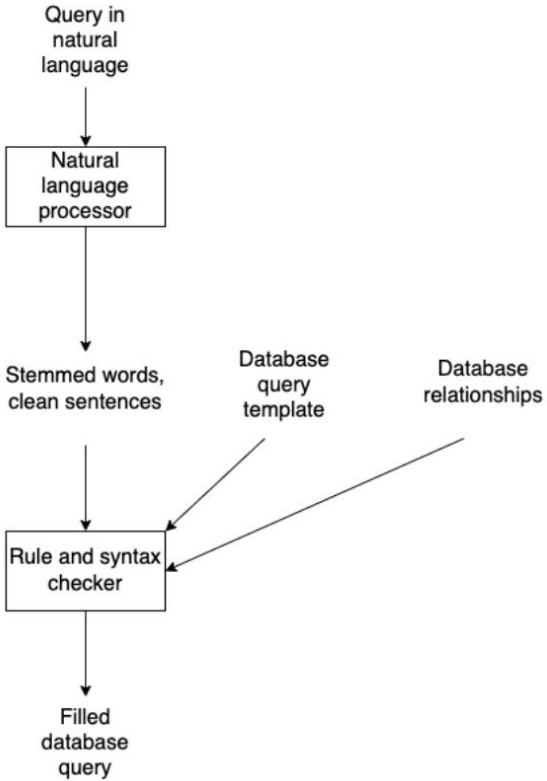
- SQLizer: Query Synthesis from Natural Language
  - Confidence-driven synthesis methodology
  - Assign a confidence score to each well-typed sketch completion
  - Semantic parser on the input sentence and improve the query sketch until it reaches to a specific confidence



(Yaghmazadeh et al., 2017)

Rule-based Methods

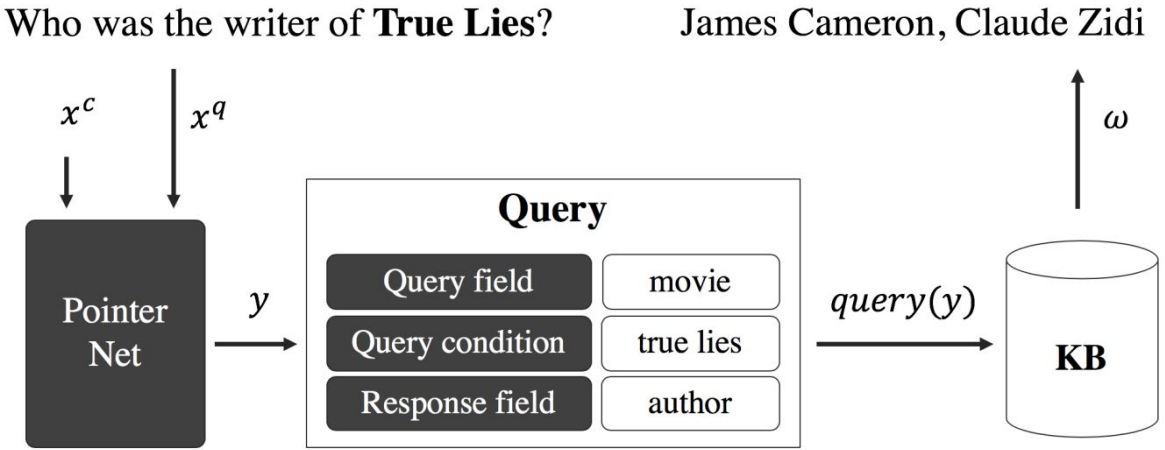
- Natural language query to NoSQL generation using query-response model
  - Natural Language Query to NoSQL Generation using Query-Response Model
  - Perform semantic analysis and eliminate irrelevant words
  - Generate queries by using synonym table for table fields



(Mondal et al., 2019)

Deep Learning based Methods

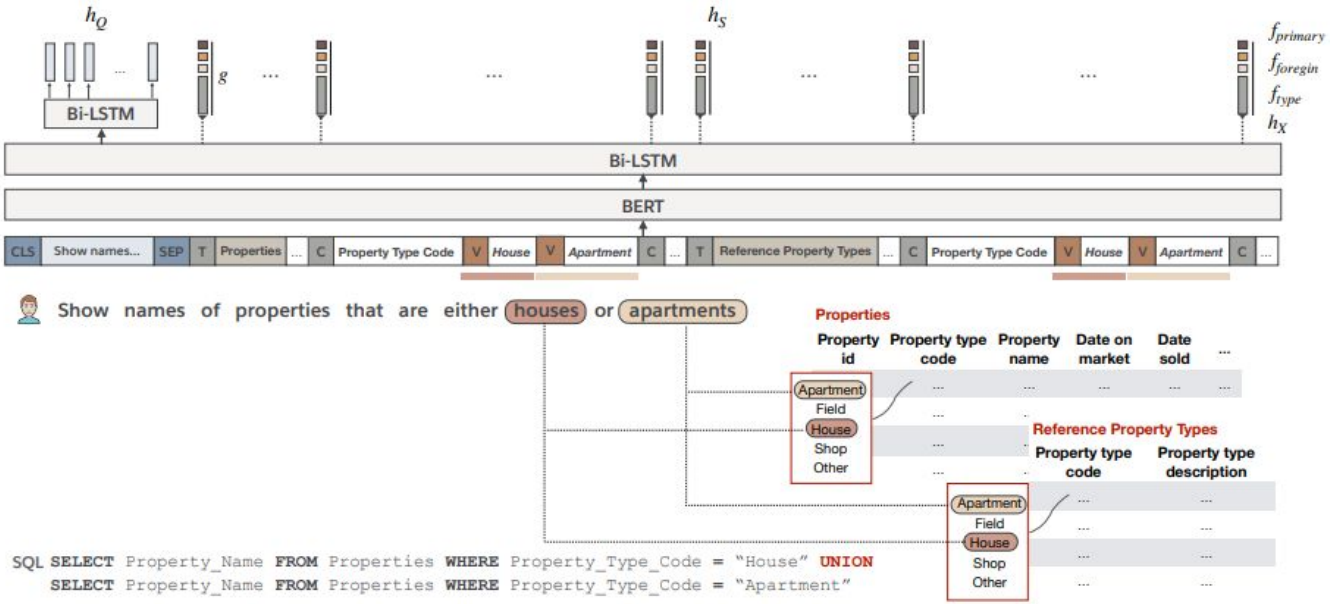
- Querying NoSQL with deep learning to answer natural language questions
  - Employ LSTM & GloVe word embeddings to extract entities in the input query
  - Reinforcement learning for training the model and targeted NoSQL databases



(Blank et al., 2019)

Deep Learning based Methods

- Bridging textual and tabular data for cross-domain text-to-sql semantic parsing
  - Employs BERT encoders by serializing the input question with database schema components

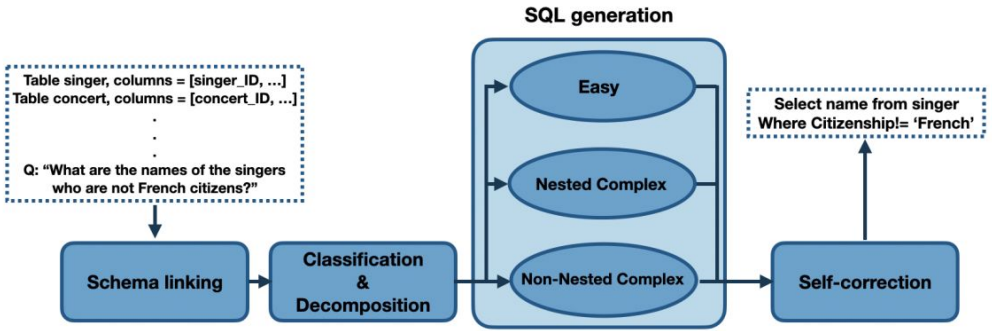


(Lin et al., 2020)

LLM-based Methods

- In-context (Zero-shot and Few-shot)
  - Embed the database schema into the prompts and ask language models to generate SQL queries
    - ChatGPT (Dong et al., 2023) : zero-shot Text-to-SQL with proper prompting
    - DIN-SQL + GPT-4 (Pourreza and Rafiei, 2023) : breaking down the SQL generation problem into sub-problems and feeding the solutions of those sub-problems into LLMs can be an effective approach for significantly improving their performance

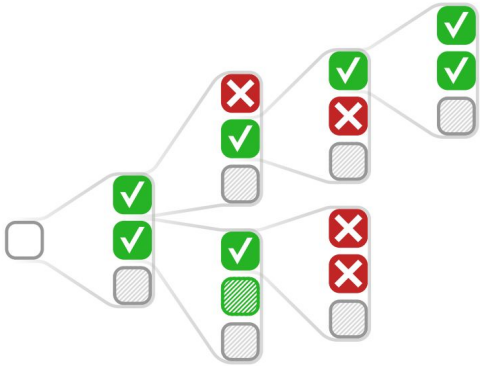
Instruction	### Complete sqlite SQL query only and with no explanation.
Context (Schema)	### Sqlite SQL tables, with their properties: # # Owners (state, owner_id, first_name) # Professionals (state, first_name, last_name) # Dogs (owner_id, dog_id, name, age) #
Question	### Which States have both owners and professionals living there? SELECT



(Dong et al., 2023; Pourreza and Rafiei, 2023)

LLM-based Methods

- Fine-tuning
  - PICARD (Scholak et al., 2021)
    - fine-tuning the T5 language model and parsing incrementally the decoded tokens
  - RATSQL (Rai et al., 2023)
    - token preprocessing with semantic parsing
    - use special tokens to mark the boundaries of components aligned between input and output
    - These special tokens implicitly help the LM-based parser build more precise input-output correspondences that are crucial for compositional generalization.



(Scholak et al., 2021)

Before preprocessing	After preprocessing
Snake case in schema items (add space)	
booking_status_code	booking _ status _ code
document_type	document _ type
Dot notation in column references (add space)	
farm.cows	farm . cows
origin.flight	origin . flight
SQL keyword (expand spelling)	
avg	average
desc	descending

Table 2: Three token preprocessing types. Coloring indicates tokenization, same as Table 1.

(Rai et al., 2023)

Large Language Models

- A language model is a machine learning model that aims to predict and generate plausible language. Autocomplete is a language model, for example.
- These models work by estimating the probability of a token or sequence of tokens occurring within a longer sequence of tokens.

Large Language Models

Consider the following sentence:

When I hear rain on my roof, I _____ in my kitchen.

If you assume that a token is a word, then a language model determines the probabilities of different words or sequences of words to replace that underscore. For example, a language model might determine the following probabilities:

cook soup *9.4%,* ***warm up a kettle*** *5.2%,* ***cower*** *3.6%,* ***nap*** *2.5%,* ***relax*** *2.2%, …*

Large Language Models

- As models are built bigger and bigger, their complexity and efficacy increases. Early language models could predict the probability of a single word; modern large language models can predict the probability of sentences, paragraphs, or even entire documents.
- The size and capability of language models has exploded over the last few years as computer memory, dataset size, and processing power increases, and more effective techniques for modeling longer text sequences are developed.
- Parameters are the weights the model learned during training, used to predict the next token in the sequence. "Large" can refer either to the number of parameters in the model, or sometimes the number of words in the dataset.

Large Language Models

- **Masked LM**: Some tokens in the input sequence are masked, and the model learns to predict the masked tokens based on the surrounding context. e.g. BERT
  - text classification, sentiment analysis, and named entity recognition

- **Sequence-to-sequence LM**: Consist of an encoder-decoder architecture, where the encoder processes the input sequence and the decoder generates the output sequence. e.g. T5
  - machine translation, summarization, and question-answering.

- **Causal LM**: The model is trained to predict the next token in a sequence given the previous tokens. e.g. Llama 2
  - text generation and summarization

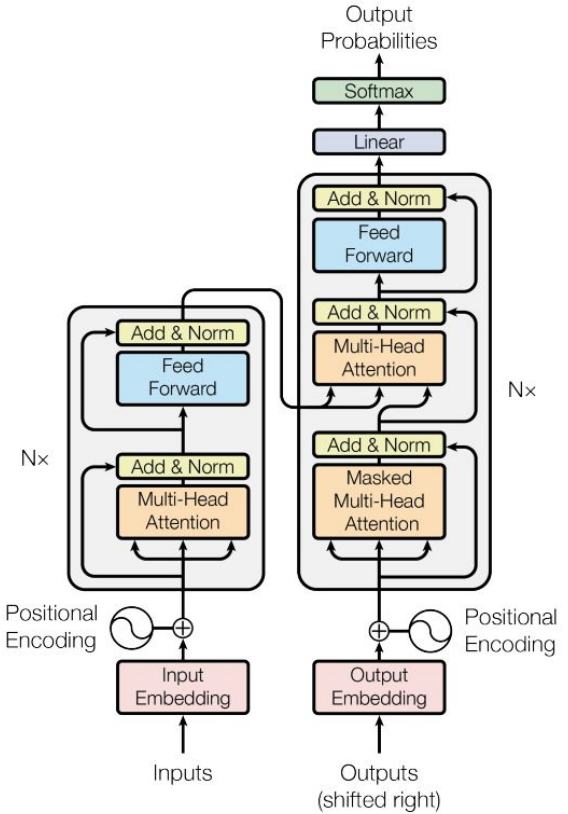
Transformers

- A key development in language modeling was the introduction in 2017 of Transformers, an architecture designed around the idea of attention. This made it possible to process longer sequences by focusing on the most important part of the input, solving memory issues encountered in earlier models.

- Encoder only: e.g. BERT (Devlin et al., 2018)

- Decoder only: e.g. GPT2 (Radford et al., 2019)

- Encoder-decoder: e.g. T5 (Raffel et al., 2020)



(Vasvani et al., 2017)

Large Language Models

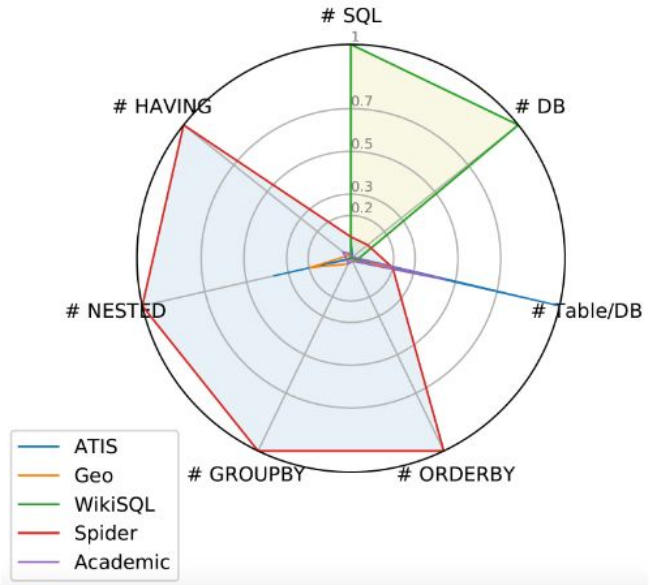
- The large language models are based on transformer architecture
- The attention mechanism allows LLMs to capture long-range dependencies between words, hence the model can understand context
- Large language model generates text autoregressively based on previously generated tokens
- LLMs have been trained on large amounts of raw text in a self-supervised fashion. Self-supervised learning is a type of training in which the objective is automatically computed from the inputs of the model. That means that humans are not needed to label the data!

Public Datasets

- ***ATIS, Geo, Academic*:** Each of them contains only a single database with a limited number of SQL queries, and has exact same SQL queries in train and test splits.

- ***WikiSQL*:** The numbers of SQL queries and tables are significantly large. But all SQL queries are simple, and each database is only a simple table without any foreign key.

- ***Spider*:** Large-scale *complex and cross-domain* semantic parsing and text-to-SQL dataset.



(Yu et al., 2018)

Use Cases for LLMs

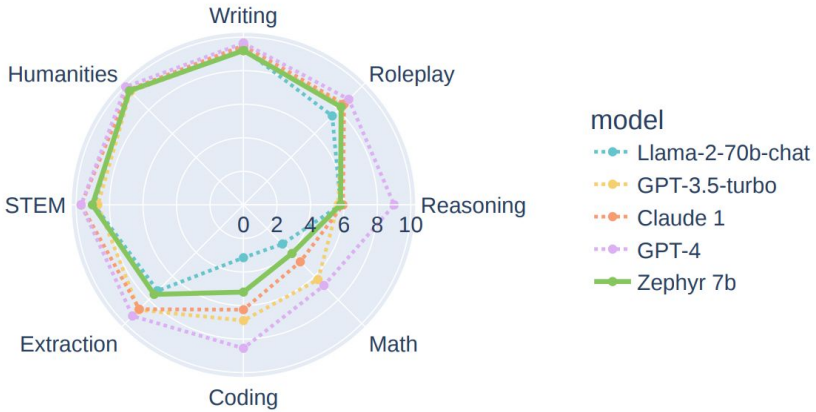
- LLMs are effective for
  - generating a plausible text in response to an input,
  - summarization, question answering, and text classification
  - solving some math problems and write code (though it's advisable to check their work!).
- LLMs are good at mimicking human speech patterns, combining information with different styles and tones.
- Recent LLMs have been used for sentiment detection, toxicity classification and image caption generation.

Use Cases for LLMs

- MT-bench: Generate answers for all 80 questions with models then use 2 kinds of judges: LLM judges and 58 expert-level human labelers. The labelers are mostly graduate students so they are considered experts and more skilled than average crowd workers. We let LLM judges evaluate all pairs and let each human evaluate at least 20 random multi-turn questions.

- On several conversational categories of MT-Bench, open-sourced language models have strong performance compared to proprietary models

- However, on more complex tasks like coding and mathematics, Open AI's GPT models outperform open-source language models



(Tunstall et al., 2023)

Open Source LLMs

- Llama2: by Meta AI, which encompasses pre-trained and fine-tuned generative text models with 7 to 70 billion parameters

- Bloom: by BigScience is a first multilingual LLM trained in complete transparency.

- Falcon: by Technology Innovation Institute (TII) can be used with chatbots to generate creative text, solve complex problems and reduce and automate repetitive tasks.

Open Source LLMs

Benefits

- **Transparency:** availability of codes used at all phases, training data, training logs and analysis results
- **Fine-tuning:** efficiently optimizing an open source LLM can reduce latency and increase performance
- **Community:** lets the enterprise take advantage of community contributions, multiple service providers and possibly internal teams to handle updates, development, maintenance and support

Risks - as for all LLMs

- **Hallucinations:** can result from the LLM being trained on incomplete, contradictory, or inaccurate data or from predicting the next accurate word based on context without understanding meaning.
- **Bias:** happens when the source of data is not diverse or representative.
- **Security:** cyber criminals using the LLM for malicious tasks such as phishing and spamming, and hackers changing original programming.

Prompt Engineering

The art of asking the right question to get the best output from an LLM.

- Enables direct interaction with the LLM using only plain language prompts.
- Effects can vary a lot among models, thus requiring heavy experimentation and heuristics.
- Has the goal of steering the model.

Prompt Engineering: tips

- Choice of prompt format, training examples, and the order of the examples can lead to dramatically different performance
- A general suggestion is to keep the selection of examples diverse, relevant to the test sample and in random order to avoid majority label bias and recency bias.
- When the validation set is limited, consider choosing the order such that the model does not produce extremely unbalanced predictions or being overconfident about its predictions. (Lu et al. 2022)

Zero-shot Prompting

Directly feed the task text to the model and ask for results. For example:

> *Text: i'll bet the video game is a lot more fun than the film.*
>
> *Sentiment:*

Few-shot Prompting

- Present a set of high-quality demonstrations, each consisting of both input and desired output, on the target task.
- As the model first sees good examples, it can better understand human intention and criteria for what kinds of answers are wanted.

  → Therefore, few-shot learning often leads to better performance than zero-shot.

Few-shot Prompting: example

Text: (lawrence bounces) all over the stage, dancing, running, sweating, mopping his face and generally displaying the wacky talent that brought him fame in the first place.

Sentiment: positive

Text: despite all evidence to the contrary, this clunker has somehow managed to pose as an actual feature movie, the kind that charges full admission and gets hyped on tv and purports to amuse small children and ostensible adults.

Sentiment: negative

Text: for the first time in years, de niro digs deep emotionally, perhaps because he's been stirred by the powerful work of his co-stars.

Sentiment: positive

Text: i'll bet the video game is a lot more fun than the film.

Sentiment:

Instruction Prompting

- Describe the task instruction to the model in the form of demonstrations.
- Few-shot can be expensive in terms of token usage and restricts the input length due to limited context length. So, why not just give the instruction directly?
- When interacting with instruction models, we should describe the task requirement in details, trying to be *specific* and *precise* and avoiding say “not do something” but rather specify what to do. For example:

  Please label the sentiment towards the movie of the given movie review. The sentiment label should be "positive" or "negative".

  Text: i'll bet the video game is a lot more fun than the film.

  Sentiment:

Chain-of-Thought (CoT)

- **Zero-shot CoT**: Use natural language statement like “Let's think step by step” to explicitly encourage the model to first generate reasoning chains
- **Few-shot CoT**: It is to prompt the model with a few demonstrations, each containing manually written (or model-generated) high-quality reasoning chains.

Chain-of-Thought (CoT): example

Question: Tom and Elizabeth have a competition to climb a hill. Elizabeth takes 30 minutes to climb the hill. Tom takes four times as long as Elizabeth does to climb the hill. How many hours does it take Tom to climb up the hill?

*Answer: It takes Tom 30*4 = <<30*4=120>>120 minutes to climb the hill.*

It takes Tom 120/60 = <<120/60=2>>2 hours to climb the hill.

So the answer is 2.

===

Question: Jack is a soccer player. He needs to buy two pairs of socks and a pair of soccer shoes. Each pair of socks cost $9.50, and the shoes cost $92. Jack has $40. How much more money does Jack need?

*Answer: The total cost of two pairs of socks is $9.50 x 2 = $<<9.5*2=19>>19.*

The total cost of the socks and the shoes is $19 + $92 = $<<19+92=111>>111.

Jack need $111 - $40 = $<<111-40=71>>71 more.

So the answer is 71.

Prompt Engineering - Demo

Fine-tuning LLMs

- Needed for tuning the model to work better for the task at hand.
- Step-by-step:
  - **Download** the pre-trained model from repository hub: Huggingface, OpenAI, …
  - **Define** the fine tuning ***configuration***
  - **Train** the model on your task-specific dataset using the training set. Monitor the performance on the validation set and adjust hyperparameters accordingly.
  - **Evaluate** your model on the test set to get an unbiased estimate of its performance once the training is complete.
  - **Save** the weights and the architecture of your fine-tuned model so you can reuse it later.
  - **Deploy** your fine-tuned model in your application or environment.

Fine-tuning LLMs

- How to define the fine-tuning configuration?
  - **Task-specific Architecture:** Create or modify the model architecture to suit your task. For example, add task-specific layers on top of the pre-trained model.
  - **Loss Function and Metrics:** Define an appropriate loss function for your task and choose evaluation metrics. This guides the model during training and helps assess its performance.
  - **Optimizer and Learning Rate:** Choose an optimizer (e.g., Adam, SGD) and set an initial learning rate. You may need to experiment with different values to find the optimal learning rate.

Spider Dataset

- The goal of the Spider challenge is to develop natural language interfaces to cross-domain databases.

- It consists of 10,181 questions and 5,693 unique complex SQL queries on 200 databases with multiple tables covering 138 different domains. In Spider 1.0, different complex SQL queries and databases appear in train and test sets.

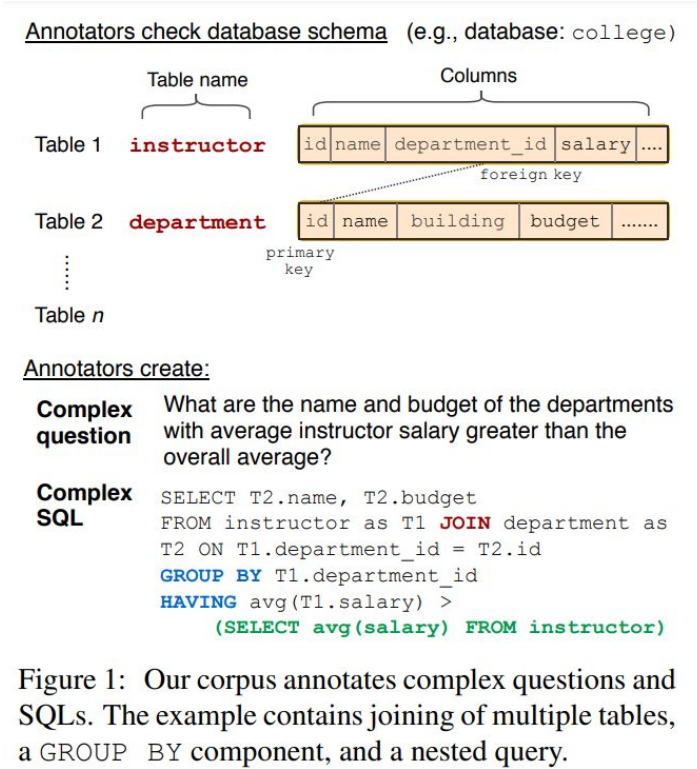


Figure 1: Our corpus annotates complex questions and SQLs. The example contains joining of multiple tables, a GROUP BY component, and a nested query.

(Yu et al., 2018)

Evaluation Metrics

- Exact Set Match without Values: conducting string comparison between the predicted and gold SQL queries, we decompose each SQL into several clauses, and conduct set comparison in each SQL clause. Measure whether the predicted query as a whole is equivalent to the gold query

- Execution Accuracy: How many predicted queries' execution results are identical to the of actual query result rows

Fine-tuning LLMs - Demo

Yaghmazadeh, N., Wang, Y., Dillig, I., & Dillig, T. (2017). SQLizer: query synthesis from natural language. Proceedings of the ACM on Programming Languages, 1(OOPSLA), 1-26.
Mondal, S., Mukherjee, P., Chakraborty, B., & Bashar, R. (2019, December). Natural language query to NoSQL generation using query-response model. In 2019 International Conference on Machine Learning and Data Engineering (iCMLDE) (pp. 85-90). IEEE.
Blank, S., Wilhelm, F., Zorn, H. P., & Rettinger, A. (2019, July). Querying nosql with deep learning to answer natural language questions. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, No. 01, pp. 9416-9421).
Lin, X. V., Socher, R., & Xiong, C. (2020). Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. arXiv preprint arXiv:2012.12627.
Dong, X., Zhang, C., Ge, Y., Mao, Y., Gao, Y., Lin, J., & Lou, D. (2023). C3: Zero-shot Text-to-SQL with ChatGPT. arXiv preprint arXiv:2307.07306.
Pourreza, M., & **Rafiei**, D. (2023). Din-sql: Decomposed in-context learning of text-to-sql with self-correction. arXiv preprint arXiv:2304.11015.
Scholak, T., Schucher, N., & Bahdanau, D. (2021). PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. arXiv preprint arXiv:2109.05093.
Rai, D., Wang, B., Zhou, Y., & Yao, Z. (2023). Improving Generalization in Language Model-Based Text-to-SQL Semantic Parsing: Two Simple Semantic Boundary-Based Techniques. arXiv preprint arXiv:2305.17378.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.
Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., ... & Radev, D. (2018). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. arXiv preprint arXiv:1809.08887.
Tunstall, L., Beeching, E., Lambert, N., Rajani, N., Rasul, K., Belkada, Y., ... & Wolf, T. (2023). Zephyr: Direct distillation of lm alignment. arXiv preprint arXiv:2310.16944.
Lu, Y., Bartolo, M., Moore, A., Riedel, S., & Stenetorp, P. (2021). Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. arXiv preprint arXiv:2104.08786.

BERT
Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
GPT2
Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.
T5
Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research, 21(1), 5485-5551.

Teşekkürler!