**Text-to-SQL**

1. text-to-sql과 중요한 점

[Text-to-SQL](http://dsba.korea.ac.kr/seminar/?mod=document&uid=1482)이란 데이터베이스와 자연어 질문이 주어지면 질문에 맞는 SQL 쿼리를 출력해주는 Task이다. text-to-sql 태스크를 위한 모델을 만들 때 풀어야 하는 두가지 문제가 있다. 첫번째는 데이터베이스에는 보통 테이블이 여러개가 있고 테이블간의 관계가 존재하는데, 정확한 쿼리를 출력하기 위해서는 테이블간의 복잡한 관계를 모델이 이해해서 질문을 답하는 데 필요한 관계를 추론해야 한다는 점이다. 두번째는 사용자의 데이터베이스 구조가 다양하기 때문에 일반화 성능이 좋아야 한다.

2. Spider 데이터셋

[Spider](https://yale-lily.github.io/spider) 데이터셋에 Multi-table queries 관련 데이터가 있다. 단순한 Select, Where부터 group by, order by, nested query 등 여러 복잡한 쿼리에 대한 데이터를 가지고 있고 또한, 정규화 성능을 측정할 수 있도록 train과 test set으로 나눠져서 학습 데이터에 없는 새로운 데이터베이스와 쿼리 데이터가 있다. 관계 이해와 일반화 성능을 측정하기 좋은 데이터셋이다.

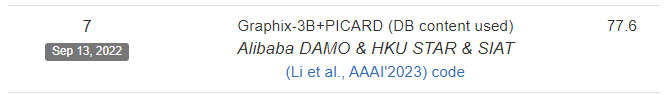
Spider와 비슷한 데이터셋으로 [SParC](https://yale-lily.github.io/sparc)가 있다. 한개의 테이블로 여러 질문을 하는 [WikiSQL](https://github.com/salesforce/WikiSQL)과 Spider를 대화형으로 만든 [CoSQL](https://yale-lily.github.io/cosql) 또한 존재한다.

2. 성능을 높이는 방법

**모델의 구조 및 미세튜닝**

Encoder와 Decoder 각각에 다른 목적을 가지고 연구가 진행되었다. Encoder의 경우 데이터베이스의 테이블들 간의 관계를 모델에 더 잘 이해시키기 위한 접근이 많았다. Encoder에 Graph-Aware Layers를 적용시키는 [Graphix-T5](https://arxiv.org/abs/2301.07507) 모델이 있는데, 2020년도 모델인 T5를 베이스로 사용했음에도 뒤에서 설명된 PICARD와 함께 적용되어 Spider1.0에서 준수한 성능을 보여준다.

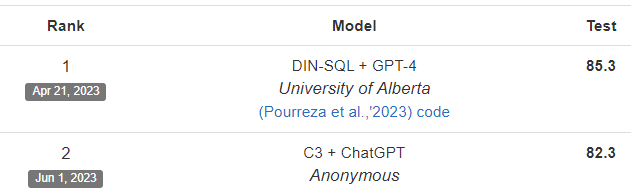
Decoder의 경우 syntax에 맞는 SQL을 출력하기 위한 접근이 있었다. 2017년에는 SQL Query문의 구조에 대한 사전 지식을 기반으로SELECT / WHERE에 어떤 값을 넣을 지 판단하는 Subtask 형식을 사용했는데, LLM이 발전하면서 추론 능력이 향상이 되었고, decoding 전략 또한 바뀌었다. 언어 모델에 자연어를 주고 SQL을 출력하라고 하면 많은 토큰들 중에 고르기 때문에 틀린 문법이나 잘못된 코드를 뱉을 때가 많았다. 코드를 돌리기 전에 컴파일러로 Syntax를 확인하는 것과 비슷한 방식을 Decoding 단계에 적용한 것이 [PICARD](https://arxiv.org/abs/2109.05093)(Parsing Incrementally for Constrained Auto-Regressive Decoding) 이다. Decoding을 진행하면서 후보들 중 Lexical, Grammatical하게 SQL의 Syntax를 따르는 후보들로만 parsing을 해서 최종 결과를 출력하는 방식인 것 같다.



**학습 데이터 및 프롬프트 엔지니어링**

프롬프트 엔지니어링은 일반적으로 inference에서 적용되지만, 학습에서도 필요하다. 학습 파이프라인을 구성할 때 언어모델이 데이터베이스 스키마를 보다 쉽게 이해할 수 있는 스키마 데이터 설명 템플릿을 만들어야 하기 때문이다. 이렇게 만들어진 템플릿은 학습이 완료된 후 inference에서도 적용이 될 수 있다.

템플릿 방식은 아니지만, Spider 스코어보드를 보면 상용화된 LLM(gpt 3.5, 4; codex 등)에 In-Context-Learning과 Self-Correction을 할 수 있는 프롬프트의 체인 그리고, 복잡한 태스크를 여러개의 작은 태스크로 분해함으로써 Spider 데이터셋에서의 성능을 높였다(하지만 해당모델들은 Spider 데이터셋을 학습과정에서 봤을 가능성이 존재한다).

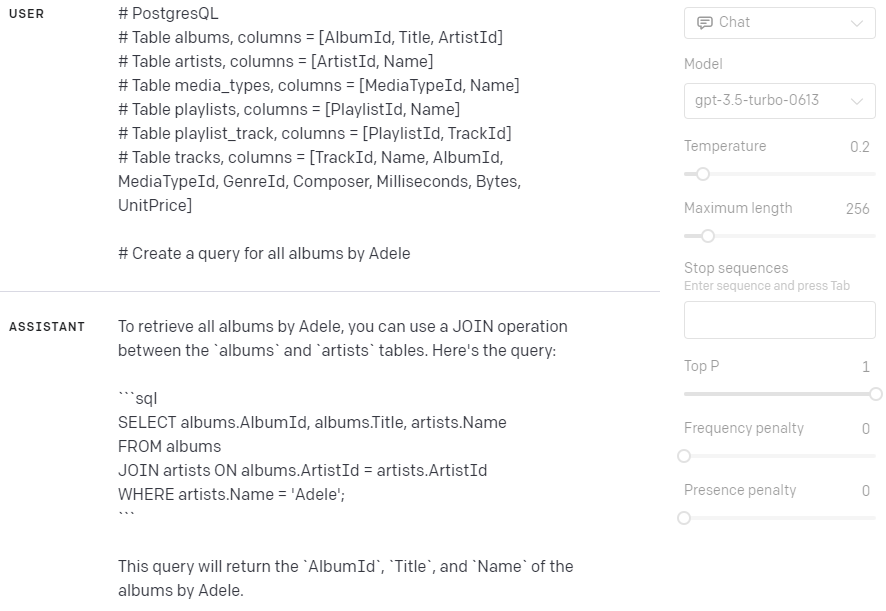




3. 출시된 제품

**OpenAI의 codex와 gpt3.5, 4(Copilot, Copilot X)**

codex는 천만개의 공개 레포지토리 데이터를 사용해 학습된 모델이다. 해당 모델은 Copilot에서 사용된 모델로 다양한 프로그래밍 언어와 SQL부터 셸 스크립트등의 생성 또는 완성 태스크가 가능한 모델이다. 현재 codex는 OpenAI에서 deprecated 되었으며, 마이크로소프트에서 사용이 가능한 것으로 보인다. gpt3.5와 4모델 또한 이러한 태스크에 좋은 성능을 보인다.



**[TEXT2SQL.AI](https://www.text2sql.ai/)**

다양한 데이터베이스의 text-to-sql 태스크와 Regex 및 엑셀 수식 질문 답변 태스크가 가능한 서비스이다. OpenAI의 Codex 모델을 사용하고 있다. OpenAI 모델을 사용하기 때문에 다양한 언어지원과 생성된 답변을 설명하는 기능 또한 가지고 있다.

4. Back-bone 모델의 성능

모델의 reasoning 능력과 성능이 좋을 수록 주어진 데이터베이스의 스키마와 질문과의 관계를 더욱 잘 이해할 수 있다. 그래서 성능이 높은 모델을 back-bone 모델로 사용해 미세튜닝을 진행하면 같은 데이터셋과 학습방식을 사용하더라도 Text-to-SQL 태스크에서도 보다 나은 성능을 보일 수 있다.

좋은 성능의 공개 모델들은 대부분 영어에 유리한 설정과 성능을 가지고 있고, 한국어보다 영어로된 공개 데이터셋과 자료들이 많기 때문에 영어 환경에서 진행하는 것도 고려해볼만한 사항이다.

5. 결론

연구에 사용된 back-bone 모델이 최근 언어모델과 성능 차이가 있기 때문에 다른 언어모델을 미세튜닝하여 성능을 향상해볼 수 있다. 또한 템플릿의 경우 비슷한 경험을 가지고 있기 때문에 다양한 실험을 통해 개선해볼 여지가 많다.