Journal of the Korean Institute of Industrial Engineers
Vol. 48, No. 1, pp. 91-104, February 2022.
ISSN 1225-0988 | EISSN 2234-6457
https://doi.org/10.7232/JKIIE.2022.48.1.091

다국어 BERT를 활용한 한국어 자연어 질의의 SQL 변환

윤훈상 · 허재혁 · 김정섭 · 강필성†

고려대학교 산업경영공학부

Text-to-SQL for Korean Language based on Multilingual BERT

Hoonsang Yoon · JaeHyuk Heo · Jeong Sub Kim · Pilsung Kang

Department of Industrial & Management Engineering, Korea University

Text-to-SQL is one of semantic parsing methods that converts natural language questions into SQL queries, and it aims to extract data from any relational database without knowledge of SQL query configuration. Although development of large amounts of datasets (WikiSQL, SPIDER) and development of pre-trained language models (BERT) contributed to the improvement of Text-to-SQL performance in English, language-specific dataset construction and model research have not been much progressed. Therefore, this study proposes a multilingual BERT-based Text-to-SQL methodology that converts the natural language question in Korean into SQL query for an English database. To this end, four strategies for translating Korean queries into English were explored, and their effectiveness was verified by applying each strategy to three text-to-SQL model structures. As a result of the experiment, it was confirmed that it showed a significant SQL generation performance even for Korean questions. The proposed methodology is meaningful in that it shows semantic inferences between database tables, column information, and questions composed of different languages are possible, and it is expected to support efficient database access by Korean users who lack proficiency in writing SQL queries.

Keywords: Text-to-SQL, Multilingual BERT, WikiSQL, SQLova, HydraNet, Bridge, Back Translation

1. 서 론

관계형 데이터베이스는 의학, 경제, 경영 등 다양한 분야에서 정형화된 데이터를 효율적으로 저장하는 시스템이며, 대량의 데이터들의 안정적인 관리가 필수적인 디지털 트랜스포메이션에 대한 변화의 물결과 함께 모든 산업 분야에서 더욱 중요성이 부각되고 있다(Zhong *et al.*, 2017). 관계형 데이터베이스 사용자는 SQL 쿼리를 통해 데이터베이스에 접근 및 수정 등의 행동을 취한다. SQL 쿼리는 범용성과 활용성이 높은 언어이지만 사용자가 원하는 데이터를 자유롭게 다루기 위해선 최적의 쿼리를 생성하는 충분한 학습과 연습이 필요하기 때문에 데이터베이스에 대한 이해가 없는 기업의 일반 직원들이 자유롭게 데이터를 활용하지 못하는 제약으로 작용하기도 한다 (Zhong *et al.*, 2017).

Text-to-SQL은 관계형 데이터베이스에 대한 자연어 질의를 SQL 쿼리로 변환해주는 의미 분석 기법 중 하나로서, SQL 쿼리 구성에 대한 이해가 부족하더라도 데이터베이스에서 원하는 데이터에 쉽게 접근 및 추출할 수 있도록 하여 SQL 비전문가의 학습 비용을 절약하고 사용자의 데이터 활용도를 높이는 것을 목적으로 한다(Wang *et al.*, 2019). Text-to-SQL은 대용량의 벤치마크 데이터 셋이 공개된 이후에 관련 연구가 폭발적으로 증가하고 있으며 대표적인 벤치마크 데이터 셋으로는 단일 테이블에 대한 자연어 질의와 SQL 쿼리 쌍을 포함하고 있는 WikiSQL 데이터 셋(Zhong *et al.*, 2017)과 다중 테이블에 대한 자연어 질의와 SQL 쿼리 쌍을 포함하는 SPIDER 데이터 셋(Yu *et al.*, 2018)이 있다. 각 데이터 셋은 자연어 질의의 양, 데이터베이스

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No. 2021-0-00471, 모델링 & 최적화 기반 오류-free 정보인프라 자율제어 기술 개발).
† 연락저자 : 강필성 교수, 02841 서울시 성북구 안암로 145 고려대학교 산업경영공학부, Tel: 02-3290-3383, Fax: 02-929-5888,
E-mail: pilsung_kang@korea.ac.kr
2021년 10월 6일 접수; 2021년 11월 18일 수정본 접수; 2021년 12월 21일 게재 확정.

테이블 개수, SQL 쿼리 난이도 등에서 차이를 보이기 때문에 특정한 하나의 데이터 셋만 활용한 연구들이 수행되거나(Hwang *et al.*, 2019; Lyu *et al.*, 2020)보다 범용성이 높은 Text-to-SQL 모델 구축을 위해 두 가지 데이터 셋을 함께 활용한 연구들이 수행되어 왔다(Wang *et al.*, 2019; Lin *et al.*, 2020).

Text-to-SQL 모델은 의미 분석 기법에서 많이 사용하는 인코더-디코더 구조를 기반으로 초기 연구가 시작되었다(Xu *et al.*, 2017; Zhong *et al.*, 2017). 사전 훈련된 언어 모델이 자연어 처리 분야에서 두각을 보이기 전에는 인코더를 순환 신경망으로 사용했으며(Hochreiter and Schmidhuber, 1997; Chung *et al.*, 2014), 현재 높은 성능을 보이는 최신 모델들은 BERT, Roberta (Devlin *et al.*, 2018; Liu *et al.*, 2019)와 같은 대규모의 사전 학습된 언어 모델을 통해 은닉 표상을 구성한다. Text-to-SQL 초기 모델들은 단순히 자연어 질의와 데이터베이스 스키마를 이어서 입력을 구성하는 방식을 사용하였으나(He *et al.*, 2019; Hwang *et al.*, 2019), 최근에 발표된 연구들은 이 외 추가적인 정보를 인코딩 과정에서 포함하는 구조를 제안하였다. 예를 들어, RAT-SQL(Wang *et al.*, 2019)은 자연어 질의와 스키마의 토큰 간의 관계를 인코딩 절차에 반영하였으며, BRIDGE(Lin *et al.*, 2020)는 셀 값을 인코딩에 포함하는 방식을 활용하였다. 인코딩 된 표상들에 대하여 디코딩하는 방식 역시 다양한 방식이 시도되어 왔으며, 대표적인 방식으로는 각 구절에 포함될 컬럼 및 연산자를 각 Subtask에 맞게 분류하는 방식(He *et al.*, 2019; Hwang *et al.*, 2019; Lyu *et al.*, 2020), Sequence-to-Sequence (Seq2Seq)와 같이 한 토큰 씩 생성하는 방식(Lin *et al.*, 2020), 그리고 코드 생성 과업에서 많이 사용하는 Abstract Syntax Tree(AST) 방식(Yin and Neubig, 2017; Wang *et al.*, 2019) 등이 있다.

현재 Text-to-SQL의 대표적인 벤치마크 데이터 셋은 모두 영어로 구성되어 있기 때문에, Text-to-SQL을 위해 개발된 모델들 역시 대부분 영어로 제공되는 자연어 질의를 SQL 쿼리로 변환하는 구조로 설계되어 있다(Hwang *et al.*, 2019; Wang *et al.*, 2019; Lin *et al.*, 2020). 영어 이외의 언어로는 영어 벤치마크 데이터 셋을 중국어 또는 베트남어로 직접 수작업으로 번역한 뒤 이를 자신들의 언어로 구성된 자연어 질의에 대한 Text-to-SQL을 진행한 연구가 소수 존재한다(Min *et al.*, 2019; Nguyen *et al.*, 2020). 그러나 한국어에서 대해서는 아직까지 관련 연구가 이루어지지 않고 있다. 이에 본 연구에서는 대부분의 관계형 데이터베이스의 스키마는 영어로 구성되어 있으나 그에 관련한 사용자의 질의는 한국어로 입력되는 상황에서 다국어 BERT(Devlin *et al.*, 2018)의 장점을 활용한 한국어 자연어-영어 데이터베이스에 대한 Text-to-SQL 기법을 제안하였다. 제안한 방법론은 영어 Text-to-SQL 데이터 셋(WikiSQL)에 대한 네 가지 변환 기준을 적용한 한국어 Text-to-SQL 데이터 셋 생성을 통해 그 효과를 검증하였다. 자체적으로 변환한 한국어 질의 Text-to-SQL 데이터 셋에 대해 제안한 방법론을 적용한 결과, 비교적 준수한 성능을 나타내었으며, 이 결과를 통해 한국어에 대한 Text-to-SQL의 가능성을 확인했다는 데에 본 연구의 의의가 있다.

본 논문의 구성은 다음과 같다. 먼저 제2장에서는 영어로 구성된 Text-to-SQL의 대표적인 벤치마크 데이터 셋, 본 연구의 성능 평가로 활용된 Text-to-SQL 모델, 그리고 Execution Guided Decoding에 대해 소개한다. 제3장에서는 한국어 WikiSQL 데이터 셋을 구축하는 방식에 대하여 소개한다. 제4장에서는 한국어 Text-to-SQL에 대한 실험 결과를 설명하고 제5장에서는 본 연구의 의의와 함께 향후 연구 방향에 대해서 논의한다.

2. 대표적 Text-to-SQL 데이터셋 및 모델

2.1 Text-to-SQL 데이터셋

2.1.1 WikiSQL

WikiSQL은 Zhong *et al.*(2017)에 의해 소개된 최초의 대용량 Text-to-SQL 데이터로서, 영문 위키피디아에 수록된 테이블(table), 해당 테이블에 대한 자연어 질의(question), 그리고 자연어 질의를 변환한 SQL 쿼리(query)로 구성되어 있다. 자연어 질의는 대상으로 하는 테이블 내에서 사용자가 원하는 값에 대한 질문이며, SQL 쿼리는 해당 질문의 결과를 추출하기 위해 데이터베이스에 전송하는 쿼리다. SQL 쿼리는 SELECT/WHERE/FROM 절만으로 구성되어 있으며, 단일 테이블에 대한 쿼리이기 때문에 테이블 간의 관계를 고려할 필요가 없는 상대적으로 단순한 형태로 구성되어 있다. WikiSQL은 총 24,241개의 테이블과 80,654쌍의 자연어 질의-SQL 쿼리 집합으로 구성되어 있으며, 해당 데이터 셋 이전에 존재하던 가장 큰 규모의 Text-to-SQL 데이터셋들에 비해 테이블 수는 10배 이상(기존: 2,420개 - WebQuestions, Berant *et al.*, 2013), 자연어 질의-SQL 쿼리 쌍은 3배 이상(기존: 26,098쌍 - Overnight (Wang *et al.*, 2015)) 더 증가된 대규모 데이터셋이다.

WikiSQL은 크라우드 소싱을 기반으로 다음 절차를 통해 구축되었다. 1단계에서는 Bhagavatula *et al.*(2013)에서 사용된 영문 위키피디아 데이터 셋의 테이블 중에서 사전에 정의된 기준을 충족하지 못하는 작은 테이블들을 제외하였다. 2단계에서는 개별 테이블들에 대해서 총 여섯 가지 중 한 가지의 템플릿을 사용하여 SQL 쿼리가 생성된다. 3단계에서는 이렇게 생성된 쿼리에 대응하는 직역에 가까운 템플릿 기반의 질의가 생성된다. 4단계에서는 크라우드 소싱을 통해 3단계에서 생성된 템플릿 기반의 질의를 의역을 통해 보다 자연어에 가까운 질의로 변환하는 작업이 수행된다. 마지막 5단계에서는 이렇게 의역된 질의와 템플릿 기반의 질의를 다른 두 명의 작업자가 평가하여 충분히 의미가 유사하다고 판단되는 경우에만 최종적으로 해당 테이블-SQL 쿼리-자연어 질의 쌍을 데이터 셋에 포함시키게 된다. <Figure 1>에서 WikiSQL에 포함된 테이블-SQL 쿼리-자연어 질의와 SQL 쿼리에 대한 예시를 살펴볼 수 있다.

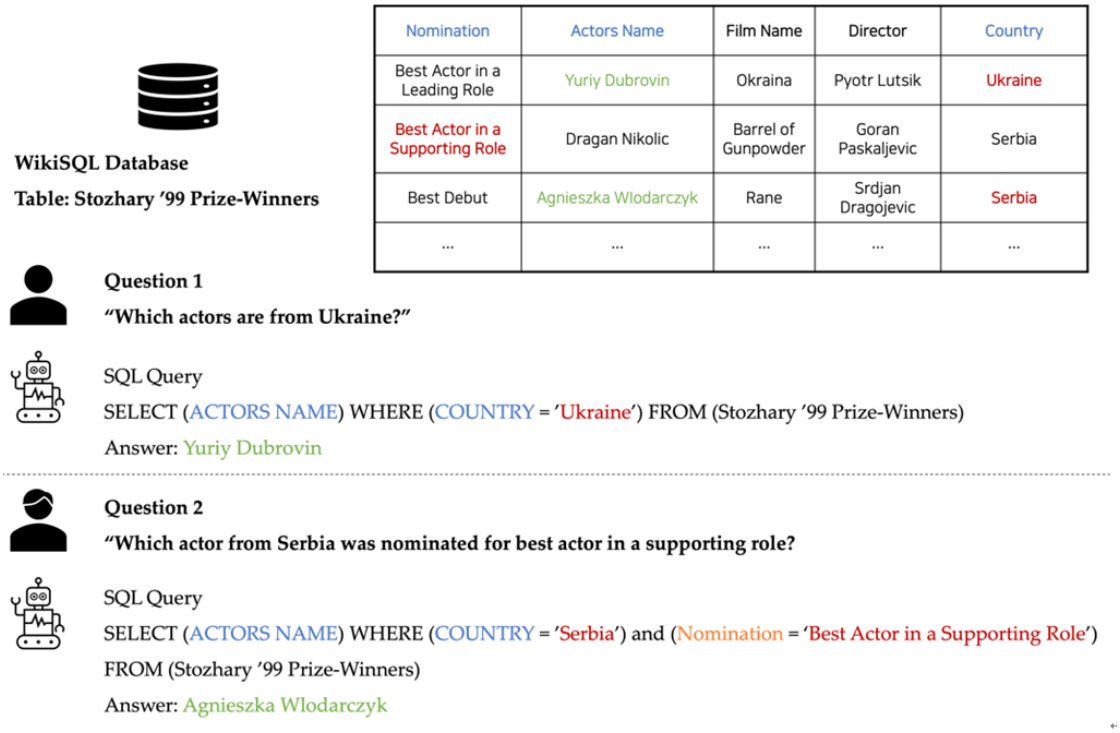


Figure 1. WikiSQL Examples

첫 번째 질의는 WikiSQL 내에 포함되어 있는 테이블 중 Stozhary라는 영화제의 수상 기록을 담은 테이블에 대하여, 사용자가 우크라이나의 영화배우가 누구인지에 대한 질문이다. 해당 질문에 대하여, 정해진 답을 도출하기 위해 테이블에 전송해야 하는 SQL 쿼리와 함께 해당 쿼리를 실행시켰을 때 반환되는 정답이 함께 제시되어 있다. 동일한 테이블에 대하여 두 번째 질의는 첫 번째 질의와 다른 국가인 세르비아 출신 영화 배우인 동시에 조연상 후보로 지명된 사람을 묻고 있다. 즉, 자연어 질의들은 테이블 내에 존재하는 다양한 값들에 접근하기 위한 사용자의 요청이라고 할 수 있다.

2.1.2 SPIDER

SPIDER는 Yu *et al.*(2018)에 의해 공개되었으며 200개의 데이터베이스 내에 다중 테이블과 1만 개의 자연어 질문과 쿼리 쌍을 포함하고 있다. SPIDER는 각 질의와 쿼리 쌍에 대해 4단계의 난이도가 할당되어 있으며, SPIDER의 SQL 쿼리는 SELECT / WHERE / FROM 이외에 HAVING, GROUP BY 등과 같은 절이 많이 포함되어 SQL 구성에 대한 숙련도를 갖춰야 작성할 수 있는 쿼리를 갖고 있다.

2.2 Text-to-SQL 모델

최신의 높은 성능을 달성하는 Text-to-SQL 모델들은 공통적인 구조가 존재한다. 이는 자연어 질문에서 정답을 도출해내는 Question-Answering 기법과 대화 내에서 사용자의 목적을 추출해내는 Dialogue State Tracking과 같은 의미 분석 기법의 기본 구조와 유사하기도 하다(McCann *et al.*, 2018; Kim *et al.*, 2019; Zeng and Nie, 2020). 먼저, 자연어 질의와 데이터베이스의 스키마(테이블 이름, 컬럼, 값 등)를 함께 입력 값으로 사용해 같은 문맥에 존재하게 하여 상호작용할 수 있도록 한다. 이후, 자연어 질의를 인코딩하는 과정에서 LSTM, GRU와 같은 순환 신경망을 사용하기도 하며(Hochreiter and Schmidhuber, 1997; Chung *et al.*, 2014) 최근에는 사전 훈련된 다양한 언어 모델(Devlin *et al.*, 2018; Liu *et al.*, 2019)을 사용한다. 인코딩된 값을 각 모델이 정의한 문제 상황에 어울리는 디코더를 통해 해석하게 되면 각 자연어 질의에 대응하는 SQL 쿼리를 생성할 수 있게 된다. <Figure 2>에서 질문과 스키마 요소들을 입력 값으로 사용해 Text-to-SQL 모델을 통과시켜 출력 값으로 SQL 쿼리를 도출하는 과정을 볼 수 있다.

2.2.1 SQLova

SQLova(Hwang *et al.*, 2019)는 Text-to-SQL 분야에 BERT 언어 모델을 통한 인코딩을 처음으로 적용한 시도로 의미가 크며, 이후에도 다양한 언어 모델을 실험한 모델들의 이론적 토대가 되었다. SQLova는 <Figure 3>과 같이 Table-aware encoding layer와 Natural Language to SQL (NLSQL) layer의 두 층으로 구성된다.

Table-aware encoding layer에서는 자연어 질의와 테이블 컬럼들의 헤더 토큰들을 결합(concatenation)하여 하나의 입력 벡터를 구성한다. 사전 학습된 대용량의 BERT 언어 모델을 활용

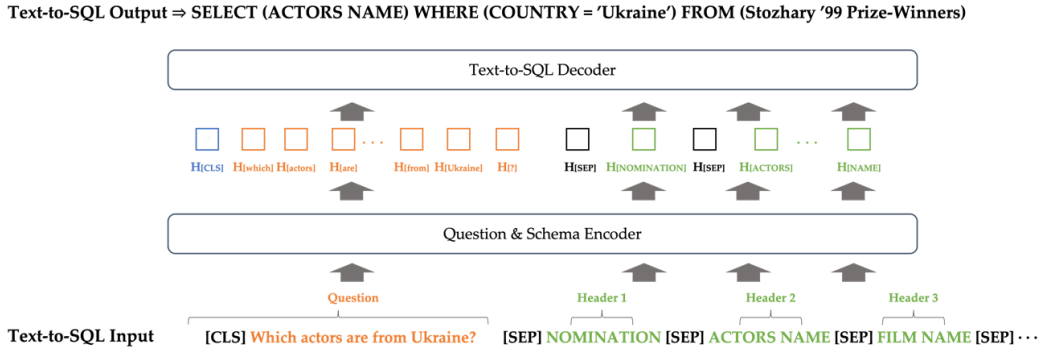


Figure 2. Text-to-SQL Model Overview

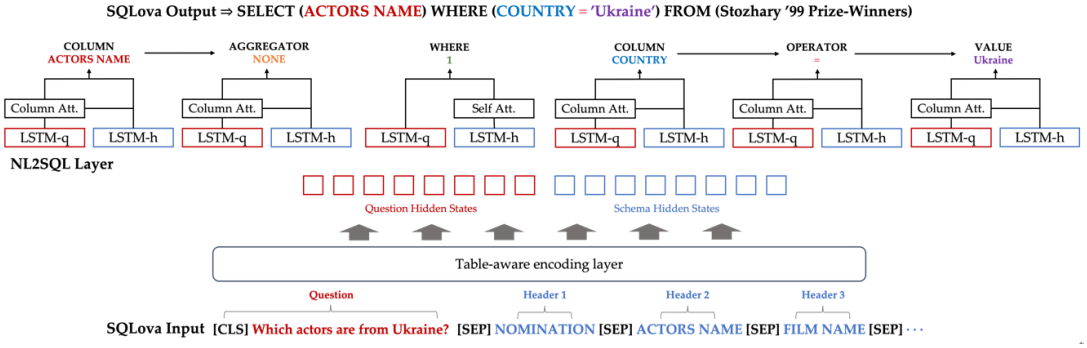


Figure 3. SQLova Model Overview

하기 위하여 입력 벡터의 첫 토큰은 [CLS] 토큰을 사용하며, 자연어 질의와 각 테이블 컬럼 헤더들 사이에는 [SEP] 토큰을 삽입하여 구분자 역할을 수행한다. 이에 더하여 세그먼트 아이디 또한 입력으로 사용하는데 자연어 질의 토큰들에는 0을 할당하고, 테이블 헤더 토큰들에는 1을 할당한다. 이렇게 구성된 입력 벡터는 BERT 언어 모델을 통해 인코딩을 진행하게 되며, BERT 인코더의 마지막 두 층의 결과물이 결합되어 이후 NL2SQL 층의 입력으로 사용된다.

NL2SQL 층에서는 질의와 컬럼 헤더 부분을 각각 나누어 LSTM을 적용시킴으로써 문맥 정보가 더욱 강화된 질의 토큰의 은닉 표상 LSTM-q와 컬럼 헤더 토큰의 은닉 표상 LSTM-h를 학습한다. 이후 하나의 LSTM-h와 전체 LSTM-q와의 컬럼 어텐션(Column Attention; Xu *et al.*, 2017)을 적용해 각 헤더와 질의 간의 관계를 학습한 표상, 또는 LSTM-h 내에서 셀프 어텐션을 적용해 만든 표상을 통해 디코딩을 진행한다. WikiSQL의 여섯 가지 SQL 쿼리 요소를 채우기 위하여 각기 다른 디코딩 절차를 따른다.

- SELECT COLUMN: 컬럼 어텐션을 통해 LSTM-q의 모든 토큰과 단일한 컬럼의 관계를 계산하여 LSTM-h의 컬럼 표상을 재구성한 뒤, 모든 컬럼에 대한 소프트맥스를 통해 SELECT에 사용할 컬럼을 결정한다.
- SELECT AGGREGATOR: SELECT COLUMN으로 정해진 컬럼에 대하여 선형 변환을 거쳐, 집합자(Aggregator)의 수와 동일한 차원으로 변환한 뒤, 소프트맥스를 통해 집합자(NONE, MAX, MIN, COUNT, SUM, and AVG)를 선택한다.
- WHERE NUMBER: LSTM-h내의 셀프 어텐션을 진행하여 모든 컬럼들의 문맥을 통해 LSTM-h의 임베딩을 새로 계산하고, 해당 컬럼들을 활용해 질문의 임베딩을 새로 계산 후 소프트맥스를 통해 벡터 최대값의 인덱스를 WHERE 절의 개수로 선택한다.
- WHERE COLUMN: SELECT COLUMN과 동일한 절차를 거치지만, 소프트맥스 대신 개별 컬럼에 대한 시그모이드 활성함수를 적용하여, WHERE NUMBER에서 구한 조건의 수만큼 컬럼을 선택한다.
- WHERE OPERATOR: SELECT AGGREGATOR과 동일한 절차를 거치지만, 소프트맥스를 통해 연산자(NONE, =, >, <)를 선택한다.
- WHERE VALUE: 컬럼과 연산자가 구해졌을 때, 컬럼 어텐션을 통해 질의의 토큰과 컬럼 간의 관계를 계산해 새로운 임베딩을 구성한 다음 구간 예측을 통해 값의 시작과 끝 인덱스를 구한다.

이후 대상 테이블을 나타내는 FROM 절과 각 Subtask로 구해낸 SELECT / WHERE절을 모아 하나의 SQL 쿼리를 구성한다.

2.2.2 HydraNet

HydraNet(Lyu *et al.*, 2020)은 Text-to-SQL을 포함한 보편적인 의미분석 기법들이 입력값으로 자연어 질문과 모든 문맥(데이터베이스 테이블 이름, 컬럼)을 함께 사용하는 것(He *et al.*, 2019; Hwang *et al.*, 2019)과 다르게, 자연어 질문과 1개의 컬럼을 하나의 쌍으로 입력값을 구성하는 특징이 있다. HydraNet은 <Figure 4>와 같이 트랜스포머 기반 인코더와 Hybrid Ranking 디코더로 구성되어 있다.

먼저 입력 값은 [CLS] 토큰 뒤에 단일한 컬럼의 정보를 구성하기 위해 컬럼 타입('string', 'real', etc.), 테이블 이름, 컬럼을 결합하고 해당하는 토큰들을 나열 한 뒤, [SEP] 토큰과 함께 자연어 질의 토큰들을 나열하여 구성한다. 이와 같이 구성한 뒤 트랜스포머 기반 언어 모델을 인코더로서 활용하게 된다면, [CLS] 토큰이 단일 컬럼과 자연어 질의 간의 관계를 포함하고 있는 '컬럼 벡터'로서의 역할을 하게 될 것이며 해당 벡터는 풀링과 같은 추가적인 처리 없이 바로 디코더의 입력 값으로 활용할 수 있다(Lyu *et al.*, 2020).

Hybrid Ranking 디코더는 단일한 컬럼 별로 확률 계산을 하여 모든 SQL 요소에 대하여 순위를 매기는 방식으로 각 SQL 요소를 예측해낸다. 전달 받은 하나의 컬럼에 대한 [CLS] 토큰 표상을 사용하여 SELECT, WHERE 컬럼으로 선택될 확률을 계산한다. 해당 과정을 모든 컬럼에 대하여 진행한 뒤 순위를 매겨 (Ranking) 가장 높은 순위의 컬럼을 SELECT, WHERE 컬럼으로 결정하는 것이다. 각 컬럼에 대한 집합자와 연산자 또한 [CLS] 토큰을 활용한 분류로써 결정하며, WHERE Value는 [CLS] 토큰을 활용하지 않고 자연어 질의 토큰을 활용한 구간 예측을 통해 얻어낸다.

- SELECT & WHERE COLUMN: 질의와 하나의 컬럼의 쌍으로 구해낸 [CLS] 토큰에 대하여 선형 변환 후, 시그모이드 활성함수를 적용하여 컬럼의 확률을 구한다. 모든 컬럼에 대하여 적용하여 가장 확률이 큰 값을 선택한다.
- SELECT & WHERE NUMBER: 모든 컬럼에서 산출된 [CLS] 토큰 임베딩에 대해, 개수를 나타내는 벡터들에 곱하여 합이 최대가 되는 벡터를 선택한다. 사전 정의된 개수는 0~3개이다.
- SELECT AGGREGATOR & WHERE OPERATOR: 선택된 컬럼에 대한 [CLS] 토큰 임베딩을 각 집합자와 연산자를 나타내는 벡터들에 곱한 뒤, 소프트맥스로 최대 확률을 나타내는 요소를 선택한다.
- WHERE VALUE: [CLS] 토큰 임베딩을 유일하게 사용하지 않는 절차로, 질의에 포함되어 있는 토큰들의 표상을 문장 내 위치 인덱스의 의미를 담고 있는 벡터와 내적하여 최대의 값을 갖는 위치를 구간 예측의 시작과 끝으로 계산하여 값을 구한다.

HydraNet에서는 SQLova와 마찬가지로 대상 테이블을 나타내는 FROM 절과 각 Subtask(Ranking)로 구해낸 SELECT / WHERE절을 모아 하나의 SQL 쿼리를 구성한다.

2.2.3 Bridge

Bridge(Lin *et al.*, 2020)는 번역 과업에서 많이 사용하는 Seq2Seq 기법을 통해 SQL 쿼리를 한 토큰 씩 생성하며(Bahdanau *et al.*, 2014; Sutskever *et al.*, 2014), 트랜스포머 기반 인코더와 포인터 제너레이터 네트워크(Pointer-Generator Network)를 활용하는 디코더로 구성된다(Vinyals *et al.*, 2015; See *et al.*, 2017)

인코더에서 사용하는 입력 값은 전체적으로 일반적인 Text-to-SQL과 비슷하여, [CLS] 토큰, 자연어 질의 토큰, [SEP] 토큰 그리고 데이터베이스 테이블 이름과 컬럼 토큰의 결합으로 구성되어 있다. Bridge는 이에 더해 각 컬럼의 셀 값을 추가적으로 결합하며 모든 셀 값을 추가하는 것이 아닌, 자연어 질의 토큰과 테이블 컬럼 내의 값이 일치할 경우 해당하는 컬럼 뒤에 값을 결합한다. <Figure 5>에서 입력 값 구성에 대한 예시를 볼 수 있으며, 자연어 질의 내의 'Ukraine'이라는 단어가 Country 컬럼의 값에 포함이 되어 있을 경우 Country 컬럼 뒤에 해당 값을 추가해준다. 셀 값을 입력 값에 추가해주는 것은 BERT가 질의와 테이블 값 간의 관계를 파악하기 용이하게 해준다(Lin *et al.*, 2020).

디코딩 과정에서는 기본적인 Seq2Seq가 어휘 집합 내에서

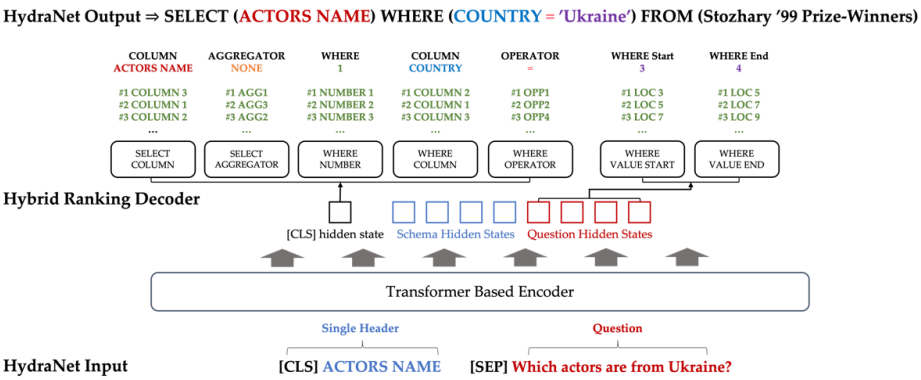


Figure 4. HydraNet Model Overview

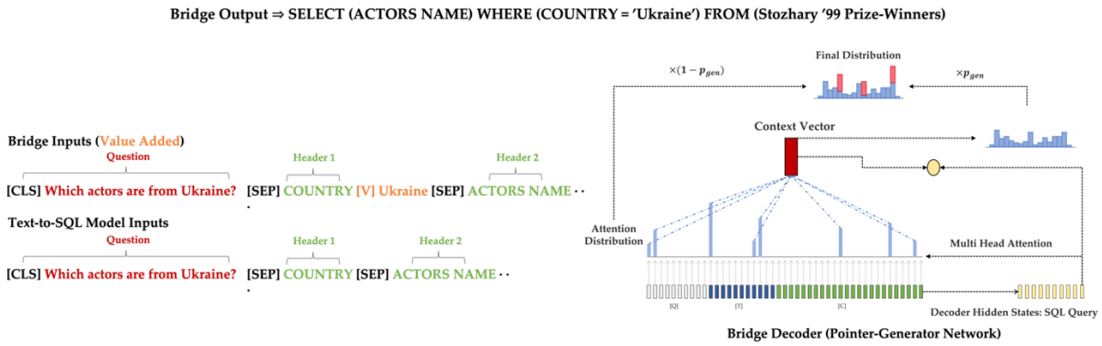


Figure 5. Bridge Model Input & Decoder

한 단어 씩 선택하여 생성하는 절차를 따르며, 이 때 각 토큰 생성 시점마다 디코더 은닉 표상과 인코더가 구성한 입력 값과의 멀티 헤드 어텐션을 계산한다(Vaswani *et al.*, 2017). 어텐션을 더한 Seq2Seq을 통해 단어를 생성해 낼 때, 어텐션은 단어 생성 과정에서 입력 값의 어떤 부분에 집중할 지에 대한 정보로서의 역할만을 하지만(Bahdanau *et al.*, 2014), 포인터 제너레이터 네트워크는 어텐션 분포를 입력 값(자연어 질의 토큰, 테이블 이름, 컬럼, 값)에서 토큰을 복사해오는 방식(Copy Mechanism)에 추가적으로 활용한다(See *et al.*, 2017). 즉, 포인터 제너레이터 네트워크는 <Figure 5>에서 볼 수 있듯, 각 디코딩 시점에서 어휘 집합의 확률 분포를 생성함과 동시에 어텐션 분포를 저장 해둔다.

이어 디코더 은닉 표상으로 0~1의 범위를 갖는 p_{gen} 확률을 계산해 p_{gen}만큼 어휘 집합 확률 분포에, (1-p_{gen})만큼 어텐션 분포에 가중치를 적용하여 최종적인 단어 생성 분포를 구성한다. 해당 방식을 통해 디코더는 어휘 집합에서 단어를 생성해 낼 수 있을 뿐 아니라, 입력 값으로 사용된 자연어 질의, 데이터베이스 테이블 이름, 컬럼 그리고 값까지 자유롭게 복사해 올 수 있다. Bridge 모델은 한 토큰 씩 순서에 맞게 생성하는 방식이기 때문에 Subtask 모델과 달리 독립적인 분류를 통해 결과물들을 하나의 SQL 쿼리로 취합하는 과정이 필요 없다. 본 모델은 SPIDER 데이터셋에서 큰 효용성이 있으나 WikiSQL에도 적용 가능하기에 본 연구의 평가 모델에 포함하였다.

2.2.4 Execution Guided Decoding (EG)

Text-to-SQL 모델들은 인코딩된 표상에 대하여 각 모델이 정의하는 문제 상황에 맞게 디코딩을 진행하는 것에 더하여, EG(Wang *et al.*, 2018)를 추가하여 성능을 향상시키고 있다(Hwang *et al.*, 2019; Wang *et al.*, 2019; Lin *et al.*, 2020). 이는 생성해낸 SQL 쿼리가 대상 데이터베이스에 대하여 실행이 가능한지에 대한 점검을 하는 과정으로, 실행이 불가능한 쿼리를 애초에 정답 후보에서 제외함으로써 정확한 SQL 쿼리를 얻어내고자 한다. Text-to-SQL 모델이 SQL 쿼리의 앞부분부터 생성해 나갈 때 SQL 쿼리의 부분만으로도 실행이 가능할 때, 데이터베이스에 적용하여 실행 가능 여부를 파악한다. 이 때, 하이퍼 파라미터로 몇 개의 쿼리 요소를 생성할 지 결정하는 빔 사이즈와 실행 가능한 쿼리 중 몇 개의 쿼리를 후보로 남겨둘 지 결정하는 Top-k를 설정한다. EG는 다음과 같은 경우에 생성 SQL 후보에서 제외한다(Wang *et al.*, 2018).

1. 런타임 에러(Runtime error): 생성해낸 쿼리 내에서 컬럼의 타입에 적용할 수 없는 집합자나 연산자가 적용될 경우이다. 예를 들어, 문자 타입 컬럼을 SELECT 컬럼으로 생성한 뒤, MAX, MIN과 같이 사용할 수 없는 집합자를 생성하게 되면 런타임 에러가 발생한다. WHERE 절의 컬럼과 연산자의 관계에서도 같은 현상이 발생할 수 있다.
2. 결과 없음(Empty returns): 생성한 SQL 쿼리를 데이터베이스에 대하여 실행했을 때, 결과 값이 없을 때이다.

3. 한국어 WikiSQL 데이터셋 구축

현재 Text-to-SQL은 영어, 중국어 등 다양한 언어로 연구가 되고 있으나(Min *et al.*, 2019; Nguyen *et al.*, 2020), 한국어에 대한 연구는 부족한 실정이며 이는 적합한 한국어 전용 Text-to-SQL 데이터 셋의 부재가 가장 큰 원인이다. 따라서 본 연구는 한국어에 적합한 Text-to-SQL 데이터 셋을 구축하기 위하여 영문 WikiSQL을 다양한 전략을 통해 번역한 데이터 셋을 구성한 뒤 제2장에서 소개된 대표적인 Text-to-SQL 모델들의 성능을 도출하여 한국어Text-to-SQL 연구의 토대를 마련하고자 한다. 이를 위해 대표적인 두 가지 영어 Text-to-SQL 데이터 셋(WikiSQL, SPIDER) 중 WikiSQL을 선택한 이유는 SPIDER 데이터 셋의 자연어 질의와 SQL 쿼리의 난이도, 복잡도가 WikiSQL에 비해 높기 때문에 이를 번역하기 위해서는 전문가의 참여가 필요하여 긴 시간과 높은 비용이 소요되기 때문이다. 반면, 상대적으로 복잡도가 낮은 WikiSQL의 경우 본 연구에서 제안하는 기계 번역(machine translation) 전략을 통해 사람의 번역과 유사한 수준의 결과물은 빠른 시간 내에 비용을 들이지 않고 생성을 할 수 있다.

본 연구에서 제안하는 번역 전략을 실행하기에 앞서 사전 실험으로 영문 WikiSQL의 질의 문장을 구글 번역기[1]를

Example 1 (WikiSQL- Table ID: "1-26041144-16″)
WikiSQL English Question: What is the catches maximum number?

WikiSQL Google Translate (Simple ENG → KOR): 최대 어획량은 얼마입니까?

WikiSQL Google Translate (Fixed ENG → KOR): catches 최대 수는 얼마입니까?

Example 2 (WikiSQL- Table ID: "2-16894271-9″)
WikiSQL English Question : What is Date, when Championship is Mackay?

WikiSQL Google Translate (Simple ENG → KOR): 챔피언십이 맥케이라면 데이트란?

WikiSQL Google Translate (Fixed ENG → KOR): Championship이 Mackay인데 Date은 무엇입니까?

Figure 6. Examples of Problems Encountered in the Translation Process

이용하여 한글로 번역한 데이터 셋에 대한 성능을 측정한 결과, 2장에서 소개된 대표적인 Text-to-SQL 모델들은 Logical Form Accuracy가 30% 수준의 낮은 성능을 나타내었다. 해당 결과는 <Table 4>의 단순 한국어 WikiSQL 결과로 확인할 수 있다. 이와 같이 구글 번역기를 이용한 단순 번역 기반 데이터 셋의 낮은 성능을 극복하기 위하여, 본 연구에서는 영어 자연어 질의 내에서 해당하는 SQL 쿼리 정답의WHERE 조건 또는 해당하는 데이터베이스의 컬럼, 값과 일치하는 단어들을 한국어로 번역이 되지 않도록 영어로 고정한 뒤 영어 자연어 질의를 한국어로 번역하는 방법을 사용하였다. 본 방식을 통해 단순 한국어 번역 데이터 셋이 갖는 두 가지 문제인 구글 번역기의 오역, 그리고 자연어 질의 내 단어와 데이터베이스 값과의 불일치 문제를 해결하여 보다 적합한 데이터 셋 구축 및 Text-to-SQL 모델 성능 향상을 도모하였다. 단순 한국어 번역 데이터셋에서 발생하는 구글 번역의 오역은 <Figure 6>의 예시를 통해 설명할 수 있다.

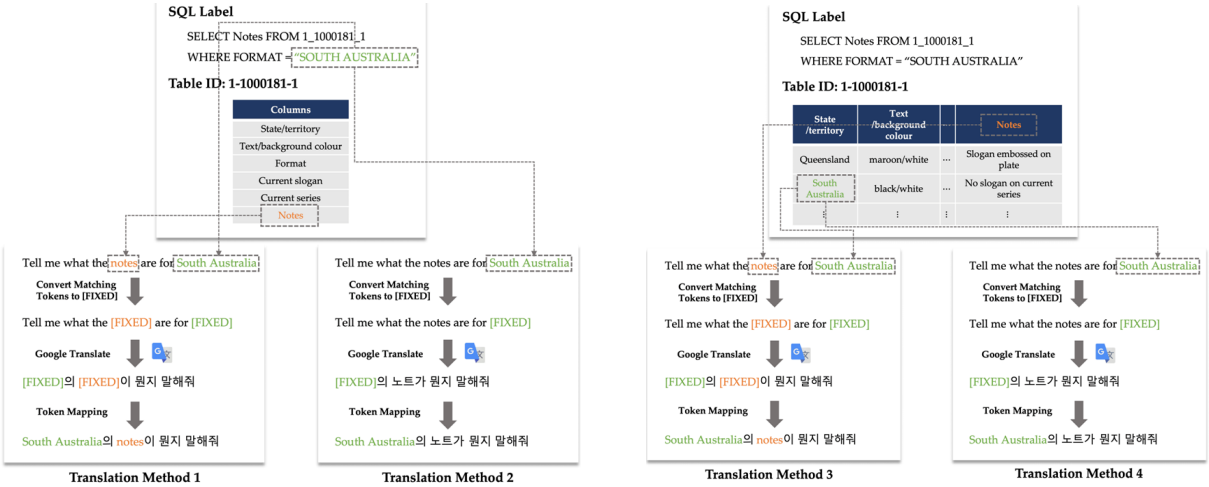
영어 WikiSQL의 자연어 질의 'What is the catches maximum number?'에 대해 본 연구의 방식을 사용하지 않고 영어 단어를 고정하지 않은 채 번역하게 된다면, '최대 어획량은 얼마입니까?'라는 문장으로 번역된다. 하지만 예시 1의 자연어 질의는 스포츠에 관련된 데이터베이스에 대한 질의이므로 영어 자연어 질의의 의중이 잘못 반영된 문장이다. 이는 'catches'가 '잡은 양'이라는 의미를 갖기에 맥락에 따라 '어획량' 또는 '포구'라는 의미도 가질 수 있는 단어의 다의성에 의해 발생하였다. WikiSQL의 자연어 질의들이 자세한 맥락을 포함하지 않는 간결한 질의로 이루어졌기 때문에 구글 번역이 이러한 다의성을 반영한 번역을 해내지 못한 경우가 많았다. 따라서, 본 연구에서는 구글 번역 기능 사용과 동시에 일부 영어 단어는 번역하지 않고 고정함으로써 다의성으로 인한 오역을 최소화했다. 이를 통해 <Figure 6>의 예시는 'catches 최대 수는 얼마입니까?'와 같이 번역되어 영어 자연어 질의의 의미가 유지되면서 한국어의 어순을 갖는 문장으로 번역할 수 있게 되었다.

단순 한국어 번역 데이터셋에서 발생하는 자연어 질의 내 단어와 데이터베이스 내 단어와의 불일치 문제는 <Figure 6>의 예시 2를 통해 살펴볼 수 있다. 영어 자연어 질의가 'What is Date, when Championship is Mackay?'일 때, 구글 번역 만을 사용할 경우 '챔피언십이 맥케이라면 데이트란?'으로 번역된다. 이는 날짜의 의미인 'Date'를 '데이트'라고 번역한 오역의 문제도 있지만, 해당 질의를 입력 값으로 Text-to-SQL 모델을 통과시켜 SQL 쿼리를 생성했을 때 WHERE Value의 평가에서 문제가 발생한다. Text-to-SQL 모델이 한국어 질의의 의미를 잘 파악하여 해당하는 SQL 쿼리를 생성했다 하더라도 WHERE 절을 'WHERE=맥케이'라고 생성하게 되는데 이는 Text-to-SQL 모델이 WHERE value를 예측 또는 생성함에 있어 자연어 질의 문장 내에서 구간을 예측하는 방법(Hwang *et al.*, 2019; Lyu *et al.*, 2020) 또는 어휘 집합에서 단어 또는 토큰을 선택하는 방식을 사용하기 때문이다(Lin *et al.*, 2020). 하지만 WikiSQL 정답 SQL 쿼리의 WHERE 절은 'WHERE=Mackay'로 구성되었기에 오답으로 평가되게 된다. 따라서 WHERE절의 평가를 정확하게 하여 평가 지표의 수치를 보완하기 위해 데이터베이스 테이블 컬럼 명, 값과 자연어 질의의 단어가 일치하는 값에 대해서 영어로 고정하는 번역 방법을 사용하였고 이를 통해 'Championship이 Mackay인데 Date은 무엇입니까?' 라는 문장으로 번역되게 된다. 이러한 방식을 통하여 Text-to-SQL 모델의 예측 결과에 대한 평가가 더욱 정확하게 나타나게 된다.

본 연구에서 제시하는 한국어 WikiSQL 구축을 위해 적용한 번역 방법은 네 가지이며, 각 방식들은 위에서 설명한 데이터 셋 내의 자연어 질의와 데이터베이스 단어들의 불일치와 오역의 비율을 줄여주기 위해 고안되었다. 해당 방식들은 공통적으로 영어 자연어 질의 내에서 해당하는 SQL 쿼리 정답의 WHERE 조건 또는 해당하는 데이터베이스의 컬럼, 값과 일치하는 단어들은 한국어로 번역을 하지 않고 영어로 고정한 뒤 영어 자연어 질의를 한국어로 번역하는 방법을 사용한다. 각 번역 방법들에 대한 설명은 다음과 같으며, 대표적인 예시는 <Figure 7>에 나타나 있다.

- 방법 1: 자연어 질의 문장의 단어 중, 해당하는 자연어 질

1) https://translate.google.co.kr/.



(a) Translation Method 1 & 2 (b) Translation method 3 & 4

Figure 7. Four Translation Methods

의와 쌍을 이루고 있는 정답(SQL Label)의 WHERE Value와 대상으로 하는 테이블의 컬럼 명과 일치하는 단어를 영어로 고정하고 번역하였다. <Figure 7> (a)의 좌측 예시에서 'notes'는 테이블의 컬럼 명에 속해 있어 고정하였고, 'South Australia'는 정답의 WHERE Value와 일치하기에 고정하였다.

- 방법 2: 자연어 질의 문장의 단어 중, 해당하는 자연어 질의와 쌍을 이루고 있는 정답(SQL Label)의 WHERE Value와 일치하는 단어를 영어로 유지하고 번역하였다. 이는 번역 방법 1에서 컬럼 명의 일치를 제외한 경우이며, <Figure 7> (a)의 우측 예시에서 'South Australia'만이 정답의 WHERE Value에 해당하여 고정한 것을 볼 수 있다.
- 방법 3: 자연어 질의 문장의 단어 중, 대상으로 하는 테이블의 컬럼 명, 값(Value)과 일치하는 단어를 영어로 유지하고 번역하였다. <Figure 7> (b)의 좌측 예시에서 'notes'는 테이블의 컬럼 명에 속해 있어 고정하였고, 'South Australia'는 테이블의 값과 일치하기에 고정하였다.
- 방법 4: 자연어 질의 문장의 단어 중, 대상으로 하는 테이블의 값(Value)과 일치하는 단어를 영어로 유지하고 번역하였다. <Figure 7> (b)의 우측 예시에서 'South Australia'만이 테이블의 값에 해당하여 고정한 것을 볼 수 있다.

본 번역 방식은 한국어 질문 속에 영어 데이터베이스 요소들이 포함되어있어 완벽한 한국어 번역이라고 할 수는 없지만 데이터베이스를 영어로 작성하는 관습을 따랐기에 자연스러운 번역이라고 할 수 있다(Min *et al.*, 2019). 또한 해당 방식은 Bridge 모델에서 자연어 질의와 테이블 값과 일치하는 경우 입력 값에 추가하여 언어 모델이 질의와 테이블 간의 관계를 더 잘 파악할 수 있게 하는 의도와 유사하다(Lin *et al.*, 2020).

4. Experiments

4.1 실험 설계

4.1.1 다국어 언어 모델

제3장에서 제시한 네 가지 데이터 셋 번역 방법을 적용해 한국어 WikiSQL을 구성하게 되면, 자연어 질의들은 한국어의 어순에 따라 번역되지만 데이터베이스 테이블의 컬럼 또는 값과 일치하는 단어들은 영어로 고정된다. 따라서, 영어 WikiSQL을 사용하여 학습하는 모델들(Hwang *et al.*, 2019; Lin *et al.*, 2020; Lyu *et al.*, 2020) 같이 인코더로 BERT, RoBERTa(Devlin *et al.*, 2018; Liu *et al.*, 2019) 등의 대용량 영어 데이터 셋으로 사전 학습된 언어 모델을 한글 질의에 바로 사용할 수 없다. 또한, 한국어 WikiSQL은 완전히 한국어로 구성된 데이터 셋이 아니기 때문에 KoBERT[2], KR-BERT(Lee *et al.*, 2020) 등과 같이 대용량의 한국어 데이터셋으로 사전 학습된 언어 모델을 인코더로 활용하기도 어렵다. 따라서, 본 연구에서 사용하는 데이터 셋을 처리하기 위해서는 한국어와 영어 두 가지 언어에 대해 모두 사전 학습이 되어 토크나이징 및 토큰들의 인코딩이 두 언어 모두에 적용 가능한 언어 모델을 사용해야 한다.

상기 조건을 만족하는 언어 모델은 한국어, 영어 등을 포함한 104개의 언어로 사전 학습된 다국어 BERT(Multilingual BERT(Devlin *et al.*, 2018))이며, 본 연구에서는 이를 활용하여 두 언어가 혼재되어 있는 한국어 WikiSQL에 대한 인코딩을 진행하였다. 즉, 본 연구에서 구성한 한국어 WikiSQL에 대한 성능 평가를 위해 SQLova, HydraNet 그리고 Bridge(Hwang *et al.*, 2019; Lin *et al.*, 2020; Lyu *et al.*, 2020)와 같이 BERT를 인코더로 활용하는 모델들에 대하여, 인코더를 다국어 BERT로 대체하여 모델을 구성하였다. 다국어 BERT는 huggingface에서 공개한 사전 학습된 모델을 활용하였으며,[3] 이에 따라 다국어

2) https://github.com/SKTBrain/KoBERT.
3) https://huggingface.co/bert-base-multilingual-cased.

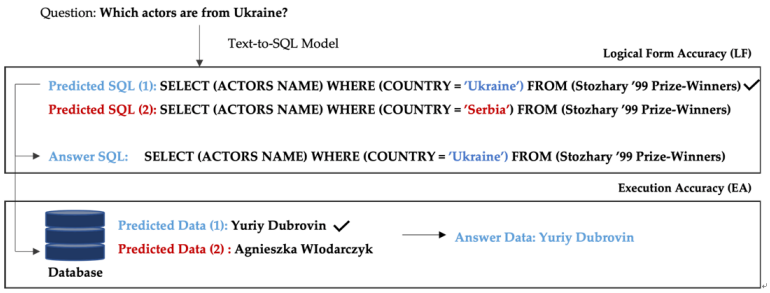


Figure 8. Text-to-SQL Metric

BERT의 토크나이저를 사용해 입력 값인 자연어 질의와 테이블 컬럼에 대해 토크나이징을 진행했다.

4.1.2 Text-to-SQL 평가 지표

Text-to-SQL 모델을 평가하기 위해서 Logical Form Accuracy (LF)와 Execution Accuracy (EA) 두 가지의 성능 평가 지표를 사용하였다.

- Logical Form Accuracy(LF)
  LF는 모델을 통해 생성한 SQL 쿼리와 정답 SQL 쿼리가 일치하는 비율이다. WikiSQL이 포함하는 SQL 쿼리는 SELECT, WHERE, FROM 절을 포함하므로 Text-to-SQL 모델이 생성한 SQL 쿼리와 정답 SQL 쿼리가 SELECT, WHERE 절에 포함된 컬럼, 집합자 그리고 연산자가 모두 일치하면 정답으로 판정한다. <Figure 8>의 예시에서 Ukraine 출신 연기자를 묻는 자연어 질의에 대하여 WHERE절의 Country 컬럼을 'Ukraine'과 'Serbia'라고 각기 예측한 SQL 쿼리가 있다. 첫 번째 쿼리는 정답과 모든 요소가 완벽히 일치하여 정답으로 판정되지만, 두 번째 쿼리는 정답과 비교했을 때 WHERE절이 달라 오답으로 판정된다.
- Execution Accuracy(EA)
  EA는 모델을 통해 생성한 SQL 쿼리를 데이터베이스에 적용했을 때 얻게 되는 값이 정답 값과 일치하는 비율이다. <Figure 8>에서 LF를 측정하기 위한 SQL 쿼리가 데이터베이스에 적용되어 값을 도출하는 것을 볼 수 있다. 이 때, 첫 번째 쿼리를 적용해 얻어낸 값은 정답 값과 일치하여 정답이라고 판정하지만, 두 번째 쿼리를 적용하여 얻어낸 값은 정답 값과 일치하지 않으므로 오답이라고 판정한다. EA는 일반적으로 LF보다 성능이 더 높게 나타나는 경향이 있으며, 그 이유는 SQL의 특성상 하나의 출력 값을 반환하는 복수의 SQL 쿼리 작성이 가능하기 때문에, LF 관점에서 오답인 쿼리를 실행하면 EA 관점에서는 정답인 출력이 생성될 수 있기 때문이다.

Table 1. WikiSQL Train / Dev / Test Dataset Description

	WikiSQL	
	Question	Table
Train	56,355	18,585
Dev	8,421	2,716
Test	15,878	5,230

4.1.3 데이터 셋 설명

한국어 WikiSQL은 영어 WikiSQL에 대해 구글 번역을 적용한 것이므로 영어 WikiSQL과 동일한 개수의 자연어 질의, SQL 쿼리 쌍 그리고 대상 데이터베이스 테이블을 갖는다. 2.1.1에서 설명했던 바와 같이 총 자연어 질의는 80,654쌍이며 대상 데이터베이스 테이블은 24,241개이다. 본 연구에서 사용한 한국어 WikiSQL의 훈련, 검증, 평가 데이터 셋의 자연어 질의, SQL 쿼리 쌍과 테이블의 개수는 <Table 1>과 같다.

4.2 번역 방식에 따른 성능 평가

네 가지의 다른 전략에 의해 번역된 한국어 WikiSQL들에 대해 본 연구에서 평가 모델로 정한 세 가지 Text-to-SQL 모델의 성능 평가 실험을 진행하였다. 각 실험에서는 데이터 셋 별로 Text-to-SQL 모델의 성능을 비교하여 의도와 맞는 데이터 셋 구축 또는 모델링이 되었는지 판단하였다.

실험 1에서는 Subtask(SQLova, HydraNet), Seq2Seq(Bridge) 기반 모델에서 모두 EG를 사용했기 때문에 각 모델의 디코딩 과정에서 사용하는 하이퍼 파라미터인 빔 사이즈를 탐색했다 (Hwang *et al.*, 2019; Lin *et al.*, 2020; Lyu *et al.*, 2020). 빔 사이즈를 4, 8, 16, 64로 조절해가며 성능을 평가한 결과는 <Table 2>와 같다.

WikiSQL의 SQL 쿼리의 길이는 SPIDER의 SQL 쿼리의 길이에 비해 상대적으로 짧기 때문에 EG를 통한 디코딩 대상 길이 역시 짧다는 특징을 갖는다. 따라서, 빔 사이즈를 8 이상으로 사용해도 얻을 수 있는 성능 향상이 미미하여 각 Text-to-SQL 모델의 성능을 나타내는 <Table 2>의 빔 사이즈를 8 까지만 표시하였다. <Table 2>에서 각 모델들의 빔 사이즈가 4에서 8로 증가할 때 성능이 LF 기준 최대 0.3, EA 기준 최대 0.3의 차이로 소폭 향상하는 것을 볼 수 있으며, SQLova 같은 경우는 빔 사이즈 증가에 따른 변화가 거의 없다. 따라서 이후의 모든 실험에서 EG를 사용할 때 빔 사이즈 8을 기준으로 성능을 비교하였다.

Table 2. Performance of Korean WikiSQL Dataset

	Translation 1		Translation 2		Translation 3		Translation 4		Difference
Beam Size	4 LF / EA (%)	8 LF / EA (%)	4 LF / EA (%)	8 LF / EA (%)	4 LF / EA (%)	8 LF / EA (%)	4 LF / EA (%)	8 LF / EA (%)	Method 1-3 (LF, EA) / Method 2-4 (LF, EA)
SQLova	70.9 / 79.3	70.9 / 79.3	66.6 / 75.6	66.6 / 75.6	70.6 / 78.8	70.6 / 78.8	67.1 / 75.8	67.1 / 75.8	(-0.3, -0.5) / (+0.5, +0.2)
HydraNet	49.4 / 59.0	49.7 / 59.3	49.3 / 59.0	49.6 / 59.3	45.7 / 55.2	45.8 / 55.4	49.2 / 59.1	49.4 / 59.3	(-3.9, -3.9) / (-0.2, 0.0)
Bridge	57.8 / 64.6	57.8 / 64.7	53.1 / 59.5	53.2 / 59.5	54.1 / 61.2	54.2 / 61.2	56.3 / 64.8	56.4 / 65.0	(-3.6, -3.5) / (+3.2, +5.5)

실험 2에서는 정답 SQL 정보를 이용한 번역 방법 1,2와 데이터베이스 테이블 정보를 이용한 번역 방법 3,4의 성능을 비교하였다. 두 번역 방법들 간의 성능 차이가 적다면 데이터베이스 테이블과 일치하는 값을 영어로 보존해도 높은 성능을 달성할 것이라 가정했으며, 실제 예측 과정에서는 정답을 통해 번역하는 것이 불가할 것이므로 번역 방법 1,2를 한국어 Text-to-SQL 성능의 상한선으로 간주하였다. 즉, 실제로 수중에 가지고 있는 자연어 질의와 데이터베이스를 통해 구성할 수 있는 번역 방법 3,4가 성능의 상한선과 얼마나 밀접한 지를 평가하는 것이다.

<Table 2>에 나타난 번역 방법 1,2와 번역 방법 3,4의 빔 사이즈 8에 대한 성능 비교 결과, 모델 별로 상이한 패턴의 결과가 나타났다. 가장 성능이 높은 SQLova는 번역 방법 1과 3, 번역 방법 2와 4가 적은 차이를 나타내고 있으며 후자의 경우에는 오히려 번역 방법 4가 LF 기준 0.5, EA 기준 0.2 차이로 성능이 약간 더 우수하다. HydraNet의 번역 방법 1과 3의 차이는 LF, EA 기준 모두 3.9 차이를 나타나는 반면 번역 방법 2와 4의 차이는 적은 차이를 나타낸다. Bridge는 번역 방법 1과 3의 차이는 3.6, EA 기준 3.5로 나타나며 번역 방법 2와 4의 비교 시에는 번역 방법 4의 성능이 LF 기준 3.2, EA 기준 5.5 차이로 더 우수하게 나타났다. 이를 통해 정답의 정보를 사용하지 않고 수중의 자연어 질의와 데이터베이스로 구성할 수 있는 번역 방법 3,4의 성능이 성능의 상한선인 번역 방법 1,2의 성능과 종합적으로 유사한 성능을 낼 수 있다고 할 수 있어, Text-to-SQL 데이터 셋을 번역할 때, 데이터베이스 테이블의 컬럼 명, 값과 일치하는 단어들을 고정시키는 방법을 사용하는 것이 합리적인 대안이 될 수 있음을 확인하였다.

실험 3에서는 성능 비교를 통해 파악한 최적의 하이퍼 파라미터인 빔 사이즈 8과, 성능의 상한선을 나타내는 번역 방법 1, 2와 종합적으로 유사한 성능이 나타났던 번역 방법 3, 4를 기준으로 한국어 Text-to-SQL의 성능을 정리하였다. 결과로서 SQLova는 번역 방법 3, 그리고 HydraNet과 Bridge는 번역 방법 4를 나타내었다. <Table 2>에서 SQLova(번역 방법 3)의 LF와 EA는 각각 70.6과 78.8을 나타냈으며, HydraNet(번역 방법 4)의 LF와 EA는 각각 49.4와 59.3, 그리고 Bridge(번역 방법 4)는 56.4와 65.0으로 나타났다.

실험 3의 결과를 통해, SQLova, Bridge, HydraNet 순으로 한국어 WikiSQL에 대한 성능이 높게 나타난 것을 확인할 수 있다. SQLova와 HydraNet은 같은Subtask 기반의 모델이고 HydraNet이 SQLova에 비해 영어 WikiSQL 리더보드[4]에서 더 높은 성능을 나타내지만 한국어 WikiSQL에 대해서는 성능이 상대적으로 저조하게 나타났다. 이런 이유로 HydraNet은 한국어 WikiSQL을 평가하기에 적합하지 않다고 판단하여 이후의 Top-k 파라미터를 변화시키는 실험과 영어 WikiSQL에 대하여 역번역(Sennrich *et al.*, 2015)을 통해 새로운 데이터를 구성하는 실험의 평가 모델로 활용하지 않았다. 따라서 이후 실험들은 Subtask 기반 모델인 SQLova와 Seq2Seq 기반 모델인 Bridge에 대하여 성능을 평가한다.

4.3 Top-k 성능 평가

EG는 2.2.4에서 설명한 바와 같이 Beam Search에 기반한 기법이기 때문에 하이퍼 파라미터로 빔 사이즈와 Top-k를 설정한다. 4.2의 실험 1 결과는 모두 빔 사이즈 8, Top-1의 결과이

Table 3. Top-k Performance of Text-to-SQL Models

	Top-1 LF / EA (%)	Top-2 LF / EA (%)	Top-3 LF / EA (%)	Top-5 LF / EA (%)	Top-10 LF / EA (%)
SQLova	70.6 / 78.8	77.4 / 86.0	80.5 / 88.3	83.7 / 90.4	85.6 / 92.0
Bridge	56.4 / 65.0	69.3 / 78.0	75.7 / 83.6	81.7 / 88.0	85.1 / 90.7

4) https://github.com/salesforce/WikiSQL.

며, 본 실험에서는 Top-k 파라미터를 변화시키면서 한국어 WikiSQL에 대한 Text-to-SQL 모델 성능이 얼마나 변할 수 있을 지 확인하고자 했다. <Table 3>에서 Top-k에 따른 번역 방법 3의 SQLova와 번역 방법 4의 Bridge 모델 성능의 변화를 살펴볼 수 있으며, 빔 사이즈는 8로 고정했다.

실험 결과, SQLova는 Top-1에 비해 Top-10의 성능이 LF 기준 15%, EA 기준 13.2%가 향상하였으며, Bridge는 이보다 더 높은 LF 기준 28.7%, EA 기준 25.7%가 향상하였다. 즉, Text-to-SQL 모델에 EG를 적용할 때 Top-k 파라미터를 증가시키면 더 높은 성능을 달성할 수 있으며, Subtask 기반 모델보다 Seq2Seq 기반 모델이 Top-k의 변화에 따라 성능이 향상되는 폭이 더 크다고 할 수 있다.

4.4 역번역 WikiSQL 데이터 셋

제1장에서 본 연구와 유사하게 중국과 베트남에서 영어 Text-to-SQL 데이터 셋 대신 자국어로 데이터 셋을 구성한 뒤, Text-to-SQL 모델을 통해 데이터 셋을 평가하는 연구를 소개하였다(Min *et al.*, 2019; Nguyen *et al.*, 2020). 두 연구는 본 연구와 달리 전문가의 번역을 통한 데이터 셋 구축과 실험을 통한 평가라는 의의가 있지만, 큰 비용을 들여 자국어 데이터 셋을 구성해야 하는 필요성을 구체적으로 서술하지 않고 있다. 즉, 자국어 질의를 영어로 번역하여 기존의 영어로 학습된 Text-to-SQL 모델의 입력으로 사용하는 것보다, 자국어 데이터 셋을 구축하여 해당 국가의 언어로 사전 학습된 언어 모델 또는 다국어 언어 모델을 통해 인코딩 하는 과정을 포함하는 Text-to-SQL 모델을 활용하는 것이 더 높은 성능을 낸다는 것에 대한 평가를 진행하지 않고 있다. 해당 비교를 진행해 후자가 Text-to-SQL 성능이 더 우수함을 보여야 자국어로 구성한 Text-to-SQL 데이터 셋의 필요성이 부각될 것이다.

두 방식의 성능을 비교하기 위해선 한국어 자연어 질의를 영어로 번역한 데이터 셋과 본 연구에서 구축한 한국어 Text-to-SQL 데이터 셋의 성능을 비교해야 한다. 하지만 전자의 데이터 셋의 경우 기존의 Text-to-SQL 한국어 데이터 셋이 존재하지 않아 '한국어 ⇒ 영어'의 방향으로 번역할 대상이 없으므로, 역번역(Back Translation)을 사용하여 한국어로만 구성되어 있는 데이터 셋을 구성하였다(Sennrich *et al.*, 2015). 역번역은 번역 과업에서 사용하는 기법으로, A언어에서 B언어로 번역한 다음, 다시 A언어로 번역을 수행하여 본래의 문장과 유사한 의미를 갖는 다른 문장을 생성하는 방식이다. 역번역의 방식대로 영어 WikiSQL의 자연어 질의를 본 연구에서 사용한 번역 방식(일부 영어 단어 고정)을 사용하지 않은 채 구글 번역을 이용해 한국어로 번역하였다. 이후 해당 한국어 자연어 질의를 다시 영어로 번역하는 과정을 거쳐 '한국어 ⇒ 영어 번역 데이터 셋(역번역 WikiSQL)'을 구축하였으며, 이를 통해 앞서 언급한 자국어를 영어로 번역하여 영어 Text-to-SQL 모델의 입력 값으로 사용할 데이터를 구성하였다.

역번역 WikiSQL과 한국어 WikiSQL 대하여 Text-to-SQL 성능을 비교하기 위하여, 본 연구에서 사용한 SQLova와 Bridge 모델을 사용하였다(Hwang *et al.*, 2019 Lin *et al.*, 2020). 또한, 기존 영어 WikiSQL과 제3장에서 소개한 번역 방법을 사용하지 않고 구글 번역을 통해 영어를 한국어로 번역한 WikiSQL (단순 한국어 WikiSQL)을 비교 대상으로 추가하였다. <Figure 9>에서 동일한 자연어 질의가 각 데이터 셋에서 어떻게 표현이 되는지 살펴볼 수 있으며, 영어 WikiSQL과 단순 한국어 WikiSQL이 역번역 WikiSQL 구성에 어떻게 활용이 되는지에 대한 도식을 볼 수 있다.

영어 WikiSQL과 역번역 WikiSQL의 자연어 질의는 모두 영

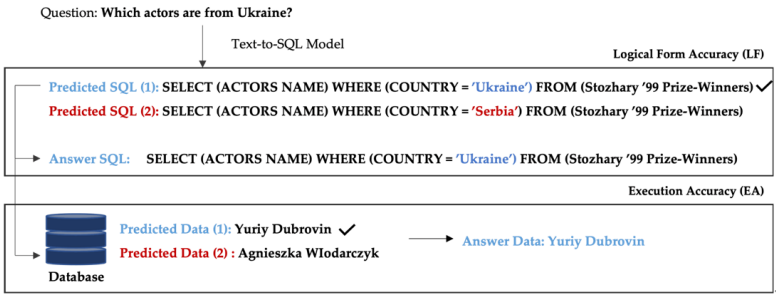


Figure 9. Comparison of Questions according to WikiSQL Translation Methods

Table 4. Text-to-SQL Model Performance for each WikiSQL

	English WikiSQL	Korean WikiSQL	Back translation WikiSQL	Simple Korean WikiSQL
Language Model	BERT-base LF / EA (%)	BERT-Multi LF / EA (%)	BERT-base LF / EA (%)	BERT-Multi LF / EA (%)
SQLova	77.1 / 84.5	70.6 / 78.8	38.7 / 46.5	30.9 / 36.1
Bridge	86.2 / 91.5	56.4 / 65.0	54.8 / 61.8	32.0 / 38.2

Table 5. SQLova Subtask Performance for Each WikiSQL

	SELECT Column (%)	SELECT Aggregator (%)	WHERE Number (%)	WHERE Column (%)	WHERE Operator (%)	WHERE Value (%)
English WikiSQL	94.4	89.8	94.9	92.0	93.0	92.0
Korean WikiSQL	89.0	88.9	93.8	88.9	90.3	90.0
Back Translation WikiSQL	63.3	66.0	62.8	57.0	59.0	53.4
Simple Korean WikiSQL	44.2	46.2	48.2	42.9	45.2	44.6

어로 구성되어 있기에 BERT 언어 모델을 각 Text-to-SQL 모델의 인코더로 설정했고, 한국어 WikiSQL과 단순 한국어 WikiSQL은 한국어와 영어가 혼재하거나 한국어만 존재하는 자연어 질의를 포함하기에 다국어 BERT를 인코더로 설정했다. 한국어 WikiSQL은 SQLova는 번역 방법 3, Bridge는 번역 방법 4를 활용해 구축한 데이터 셋을 사용하여 성능을 나타냈으며, 두 Text-to-SQL 모델의 EG에 사용되는 하이퍼 파라미터를 빔 사이즈 8과 Top-1로 설정하였다.

<Table 4>는 영어, 한국어, 역번역 그리고 단순 한국어 WikiSQL에 대한 실험 결과를 나타내고 있으며, 언급된 순서대로 높은 성능을 보이는 것을 확인하였다. 한국어 WikiSQL의 성능이 SQLova, Bridge 두 모델에 대하여 모두 역번역 WikiSQL보다 높게 나타나 자국어 자연어 질의가 구성된 데이터 셋을 구성하여 Text-to-SQL 모델을 구축하는 것이 자연어 질의를 영어로 번역하여 영어로 학습된 Text-to-SQL에 적용하는 것보다 높은 성능을 낼 수 있는 방법이라는 것을 보였다. 또한 한국어, 역번역 WikiSQL에 대한 Subtask(SQLova)와 Seq2Seq(Bridge) 기반 모델의 성능 하락 폭을 비교했을 때, Seq2Seq(Bridge)에 비해 Subtask(SQLova)가 더 많은 폭으로 성능이 저하되는 것을 확인하였다.

<Table 5>에서는 각 WikiSQL 별 SQLova의 Subtask(SELECT Column, WHERE Number 등) 성능을 살펴볼 수 있으며, <Table 4>의 결과와 같이 영어, 한국어, 역번역, 단순 한국어 WikiSQL 순으로 각 Subtask의 성능이 높은 것을 알 수 있다. 특히, 역번역을 사용할 경우, 집합자, 연산자 그리고 WHERE 절의 개수에 대한 성능 저하보다 컬럼과 값에 대한 성능의 하락 폭이 더 큰 것을 볼 수 있다. 이는 역번역으로 인하여 영어 자연어 질의의 단어들이 같은 의미를 가지지만 다른 표현으로 나타나는 경우와, 두 번의 번역 절차로 인하여 자연어 질의의 본래의 의중이 변하게 되어 단어의 의미가 아예 달라지는 문제점을 내포하기 때문이다.

본 장의 실험으로 자국어로 구성된 자연어 질의를 영어로 번역하여 영어로 훈련된 Text-to-SQL 모델의 입력 값으로 사용하는 것보다, 자국어 자연어 질의 자체를 사용해 Text-to-SQL 모델을 학습한 뒤 해당 모델을 통해 SQL 쿼리 문을 생성하는 것이 더 좋은 방법임을 나타냈다. 따라서 본 장의 실험은 Text-to-SQL의 발전과 함께 진행되는 비영어권 국가들의 자국어로 구성된 Text-to-SQL 데이터 셋을 구축하고자 하는 여러 연구들(Min *et al.*, 2019; Nguyen *et al.*, 2020)에 대한 정당성과 필요성을 부각해주는 시도이다. 또한, 본 연구에서는 영어 Text-to-SQL 데이터 셋을 자국어 번역하기 위한 적절한 방식을 제안하여 시간과 비용을 효율적으로 사용하는 Text-to-SQL 데이터 셋 구축이 가능함을 보였다.

5. 결 론

본 연구에서는 사용자가 관계형 데이터베이스에서 SQL 쿼리 문법에 대한 사전 지식 없이 원하는 정보를 얻어낼 수 있도록 자연어로 구성된 질의를 SQL 쿼리로 변환해주는 Text-to-SQL의 한국어 데이터 셋을 제안하였다. 현재 Text-to-SQL의 발전은 영어 벤치마크 데이터 셋인 WikiSQL과 SPIDER를 중심으로 이뤄지고 있으며 중국, 베트남 등에서 자국어로 구성한 데이터를 제안하였지만, 한국어로 구성된 Text-to-SQL 데이터 셋은 존재하지 않다. 이런 배경에서 기존 영어 WikiSQL 데이터 셋을 구글 번역을 활용하여 한국어 WikiSQL을 구축하였으며, 정답 SQL 쿼리의 정보와 데이터 베이스 테이블의 컬럼 명, 값이 자연어 질의 내에 포함되어 있을 경우 영어로 고정시키는 전처리를 적용하여 4가지 방식으로 번역하였다. 일부 영어 단어를 고정시키는 방법은 한국어 어순에 맞는 자연어 질의로 번역할 수 있게 했으며, 데이터베이스 요소들과의 일치를 원활하게 하여 성능 평가가 용이해졌다. 구성한 한국어 WikiSQL의 성능을 평가하기 위하여 Subtask(SQLova, HydraNet), Seq2Seq(Bridge) 기반의 세 가지 Text-to-SQL 모델을 사용하였고, 한국어와 영어가 혼재되어 있는 한국어 WikiSQL에 대한 인코딩을 가능하게 하기 위해 각 모델의 인코더를 BERT 대신 다국어 BERT로 교체하였다. 데이터베이스 정보만을 활용한 번역 방법 3,4로 구성한 데이터 셋과 모델들로 EG(빔 사이즈 8, Top-1)를 적용하여 성능 평가를 실시한 결과 LF 기준 45.7~70.6%, EA 기준 55.2%~78.8%의 성능을 도출하였다. 첫 번째 추가 실험에서는 EG의 하이퍼 파라미터인 Top-k를 10까지 증가시켜 모델들의 성능을 평가했다. 최대 LF 기준 85.1%~85.6%, EA 기준 90.7%~92.0%를 달성하여 영어 WikiSQL의 성능과 준하는 결과를 나타내었다. 두 번째 추가 실험에서는 자국어로 구성된 Text-to-SQL 데이터 셋을 통해 모델을 구성하는 방법과 자연어 질의를 영어로 번역하여 영어 Text-to-SQL 데이터 셋으로 훈련된 모델의 입력으로 사용하는

방법과의 비교를 진행하였다. 이를 위해 영어 데이터 셋을 역번역하여 한국어 데이터 셋을 영어로 번역하는 과정을 구성하였고 성능 비교 결과 자국어로 구성된 데이터 셋의 성능이 더 높게 나타났다. 해당 실험을 통해 자국어 데이터 셋 구축에 대한 당위성과 필요성을 부각시켰다.

본 연구를 기반으로 한국어 Text-to-SQL의 성능 발전을 위한 몇 가지 방안이 존재한다. 첫 번째로, 구글 번역[5]과 네이버의 파파고[6]와 같은 기계번역 대신 전문가의 번역을 통해 Text-to-SQL 데이터 셋의 자연어 질의를 번역하는 것이다. 구글 번역은 원 자연어 질의의 의중을 흐리는 오역을 다소 생성해 냈기에 한국어 WikiSQL이 영어 WikiSQL보다 낮은 성능을 달성했을 가능성이 있으며 이를 올바른 번역이 적용된 데이터 셋으로 극복할 수 있을 것이다. 또한 본 연구에서 제시한 번역 방법 대신 전문가를 통한 번역을 거친다면 영어와 한국어가 혼재하지 않은 완전한 한국어 자연어 질의를 구성할 수 있어 진정한 의미의 한국어 Text-to-SQL이 달성될 것이다. 한국어만으로 구성된 데이터 셋은 한국어로 사전 훈련된 모델을 인코더로 활용할 수 있도록 하며, 이는 다국어 BERT로 인한 과업 성능 하락을 방지할 수 있을 것이다(Rust *et al.*, 2020). 두 번째로, SQLova, HydraNet, Bridge(Hwang *et al.*, 2019; Lin *et al.*, 2020; Lyu *et al.*, 2020)에 더해 다양한 모델로 성능 평가를 진행하는 것이다. 제한된 개수의 모델 성능은 한국어 WikiSQL의 최대 성능을 측정하기에 부족하며 특히 Subtask, Seq2Seq 기반 모델 이외에 그래프 또는 AST 등을 사용하는 모델(Wang *et al.*, 2019)에 대하여 추가적으로 평가해야 할 것이다. 마지막으로, WikiSQL을 기반으로 한국어 데이터 셋을 구축하는 대신 SPIDER(Yu *et al.*, 2018)에 대한 한국어 데이터 셋 구축 및 평가를 하는 것이다. WikiSQL에 비해 SPIDER 데이터 셋은 더 높은 난이도, 현업에서 사용할 만한 복잡한 SQL 쿼리를 포함하고 있으므로 Text-to-SQL의 효용성이 더 높은 데이터 셋이다. 한국어 SPIDER 데이터 셋과 모델을 구축하는 것은 SQL 쿼리 구성에 어려움을 느껴 데이터베이스에 접근하지 못하는 사용자들에게 더 큰 편리함을 제공할 것이다.

참고문헌

Bahdanau, D., Cho, K., and Bengio, Y. (2014), Neural Machine Translation by Jointly Learning to Align and Translate, arXiv preprint arXiv:1409.0473.

Berant, J., Chou, A., Frostig, R., and Liang, P. (2013), Semantic Parsing on Freebase from Question-Answer Pairs, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Bhagavatula, C. S., Noraset, T., and Downey, D. (2013), Methods for Exploring and Mining Tables on Wikipedia, *Proceedings of the ACM SIGKDD workshop on interactive data exploration and analytics*.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014), Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, arXiv preprint arXiv:1412.3555.

Devlin, J., Chang, M-W., Lee, K., and Toutanova, K. (2018), Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding, arXiv preprint arXiv:1810.04805.

He, P., Mao, Y., Chakrabarti, K., and Chen, W. (2019), X-SQL: Reinforce Schema Representation with Context, arXiv preprint arXiv:1908.08113.

Hochreiter, S., and Schmidhuber, J. (1997), Long Short-term Memory, *Neural Computation*, **9**(8), 1735-1780.

Hwang, W., Yim, J., Park, S., and Seo, M. (2019), A Comprehensive Exploration on Wikisql with Table-aware Word Contextualization, arXiv preprint arXiv:1902.01069.

Kim, S., Yang, S., Kim, G., and Lee, S. W. (2019), Efficient Dialogue State Tracking by Selectively Overwriting Memory, arXiv preprint arXiv:1911.03906.

Lee, S., Jang, H., Baik, Y., Park, S., and Shin, H. (2020), Kr-bert: A Small-scale Korean-specific Language Model, arXiv preprint arXiv:2008.03979.

Lin, X. V., Socher, R., and Xiong, C. (2020), Bridging Textual and Tabular Data for Cross-domain Text-to-sql Semantic Parsing, arXiv preprint arXiv:2012.12627.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019), Roberta: A Robustly Optimized Bert Pretraining Approach, arXiv preprint arXiv:1907.11692.

Lyu, Q., Chakrabarti, K., Hathi, S., Kundu, S., Zhang, J., and Chen, Z. (2020), Hybrid Ranking Network for Text-to-sql, arXiv preprint arXiv:2008.04759.

McCann, B., Keskar, N. S., Xiong, C., and Socher, R. (2018), The Natural Language Decathlon: Multitask Learning as Question Answering, arXiv preprint arXiv:1806.08730.

Min, Q., Shi, Y., and Zhang, Y. (2019), A Pilot Study for Chinese sql Semantic Parsing, arXiv preprint arXiv:1909.13293.

Nguyen, A. T., Dao, M. H., and Nguyen, D. Q. (2020), A Pilot Study of Text-to-SQL Semantic Parsing for Vietnamese, arXiv preprint arXiv:2010.01891.

Rust, P., Pfeiffer, J., Vulić, I., Ruder, S., and Gurevych, I. (2020), How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models, arXiv preprint arXiv:2012.15613.

See, A., Liu, P. J., and Manning, C. D. (2017), Get to the Point: Summarization with Pointer-generator Networks, arXiv preprint arXiv:1704.04368.

Sennrich, R., Haddow, B., and Birch, A. (2015), Improving Neural Machine Translation Models with Monolingual Data, arXiv preprint arXiv:1511.06709.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014), Sequence to Sequence Learning with Neural Networks, *Advances in Neural Information Processing Systems*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017), Attention is All You Need, *Advances in Neural Information Processing Systems*.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015), Pointer Networks, arXiv preprint arXiv:1506.03134.

5) https://translate.google.co.kr/.

6) https://papago.naver.com/.

Wang, B., Shin, R., Liu, X., Polozov, O., and Richardson, M. (2019), Rat-sql: Relation-aware Schema Encoding and Linking for Text-to-sql Parsers, arXiv preprint arXiv:1911.04942.

Wang, C., Tatwawadi, K., Brockschmidt, M., Huang, P. S., Mao, Y., Polozov, O., and Singh, R. (2018), Robust Text-to-sql Generation with Execution-guided Decoding, arXiv preprint arXiv:1807.03100.

Wang, Y., Berant, J., and Liang, P. (2015), Building a Semantic Parser Overnight, *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (Volume 1: Long Papers).

Xu, X., Liu, C., and Song, D. (2017), Sqlnet: Generating Structured Queries from Natural Language without Reinforcement Learning, arXiv preprint arXiv:1711.04436.

Yin, P., and Neubig, G. (2017), A Syntactic Neural Model for General-purpose Code Generation, arXiv preprint arXiv:1704.01696.

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., and Roman, S. (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.

Zeng, Y., and Nie, J. Y. (2020), Jointly Optimizing State Operation Prediction and Value Generation for Dialogue State Tracking, arXiv preprint arXiv:2010.14061.

Zhong, V., Xiong, C., and Socher, R. (2017), Seq2sql: Generating Structured Queries from Natural Language Using Reinforcement Learning, arXiv preprint arXiv:1709.00103.

저자소개

윤훈상: 성균관대학교 심리학과에서 2019년 학사학위를 취득하였다. 현재는 고려대학교 산업경영공학부에서 석사과정으로 재학 중이다. 연구 분야는 비정형 데이터를 활용한 데이터마이닝과 대화 상태 추적이다.

허재혁: 수원대학교 응용통계학과에서 2019년 학사학위를 취득하였다. 현재는 고려대학교 산업경영공학부에서 석사과정으로 재학 중이다. 연구 분야는 이미지 모델 해석과 견고성 분석 그리고 이상치 탐지이다.

김정섭: 한국외국어대학교 경영학부에서 2018년 학사학위를 취득하였다. 현재는 고려대학교 산업경영공학부에서 석박통합과정으로 재학 중이다. 연구 분야는 시계열 데이터를 활용한 이상치 탐지이다.

강필성: 서울대학교 산업공학과에서 2003년 학사, 2010년 박사학위를 취득하였다. 이후 현대카드 과장, 서울과학기술대학교 조교수로 근무하였으며, 현재는 고려대학교 산업경영공학부 부교수로 재직 중이다. 연구 분야는 정형 및 비정형 데이터를 활용한 데이터마이닝 및 기계학습 알고리즘 개발 및 제조/IT/공공분야 응용이다.