Yugant Hadiyal, B.E.

Answering Software-Questions using Neural Networks

Literature Review

to complete the Research Seminar

Master of Science

Master's degree: Research in Computer and System Engineering

submitted to

Technical University Ilmenau

Supervisor

M. Sc. Mihaela Todorova Tomova
Prof. Dr.-Ing. Patrick Mäder (JP)

Softwaretechnik für sicherheitskritische Systeme
Head: Prof. Dr.-Ing. Patrick Mäder (JP)

Ilmenau, September 2020

This document is set in Palatino, compiled with pdfLaTeX2e and Biber.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date	Signature

Abstract

Interacting with databases requires knowledge of the query languages. A person without prior training in query languages might find it challenging to access databases or might not be able to access it at all. Natural Language Processing[NLP] can help to make databases available to an individual without having any information about the database and query language. NLP is a subfield of linguistics, computer science, information engineering, and AI concerned with the interactions between computers and human (natural) languages, precisely the way to program computers to process and analyze large amounts of linguistic communication data. This literature review focuses on the use case of natural language to SQL query for software questions. Here, software questions are about the software development process or software management systems. Project management systems provide information on the project's development process. This information helps to track the progress of the development. These project systems store the data in databases and need the training to use it. Applying NLP concepts, project management systems can be made available to individuals without any special knowledge about the database like schema, tables, architecture and domain of the information. Deep Neural Networks[DNNs] come handy to solve this kind of problems. This solution will be economically beneficial because tracking of the project management requires specialized human resources which is costly. Hiring technical human resource is about time and cost trade-off. In such a scenario, either company can hire people with experience or can train the available ones. Software to SQL models helps to optimize this trade-off. The overall purpose of the solution is to make human interaction with a database as simple as possible using the English language.

Abbreviations

NLP	Natural Language Processing
AI	Artificial Intelligence
RNN	Recurrent Neural Network
BERT	Bidirectional Encoder Representations from Transformers
SemQL	Semantic query language
TF-IDF	Term Frequency- Inverse Document Frequency
BOW	Bag of Words
IRNet	Intermediate Representation Network

List of Figures

Contents

1 Introduction

In recent years, because of the availability of substantial computational power, NLP has become accessible to the industries and individuals. Accessing databases with natural language is an open problem. In the field of NLP, there are many techniques to retrieve information from structured or unstructured data. Various use-cases can leverage these techniques; one of such use-case is retrieving data from a project management system. For planning, organizing and managing the resources for software development, project management systems are useful. Retrieving data from databases using human language is problematic because of the complexity of generating a SQL query from natural language. Top of the complexities is the ambiguity of the human language and synthesizing SQL-query from a question. The system may generate the same query for different questions which increases uncertainty and error of the system. With the help of the deep neural networks, retrieval of the information from the database from human language can be possible. Recurrent Neural Networks (RNNs) have proven to be useful when it comes to sequence to sequence generation. Question asked in human language is a sequence of words, and the desired output is a SQL query which is also a sequence. This literature review compares various state-of-the-art approaches/implementations like BERT, IRNet, TypeSQL and TREQS. To create such a model training dataset is one of the crucial factors, there are many datasets available. WikiSQL is a crowdsourced dataset which contains structured data, questions and SQL query for that. Spider is a similar dataset with cross-domain data. This dataset can be used to develop a model which can generate a SQL query based on questions asked in human language. Most of such datasets are crowdsourced, namely, WikiSQL, Spider, Academic, Advising, ATIS, Geography, Restaurants, Scholar, IMDB, and Yelp.

2 Fundamentals

Various approaches can be useful to create a model which can convert the natural language to SQL query. These approaches divide this problem into several subproblems. Some techniques use a sketch mechanism where they fill the missing part of the SQL query. The sketch is a template with some fixed keywords present for the SQL query like SELECT, FROM, and other keywords for different templates. Classifiers created using DNN fills the missing words in the SQL query. In another approach, a SQL query is generated as a sequence of words where input sequence is words from the question in English, and output is words which are a SQL query. Making a solution for cross-domain is a challenging problem. A suitable dataset is always essential to achieve high accuracy.

2.1 Word embedding + BERT

This approach is divided into three parts, as follows:

- Generating a feature vector from the cell values of the table with the same length of the question.
- Generating a header vector from the columns of the table with the same length of the question.
- Passing this combined vector to the BERT model.

BERT is a profound transformer-based model. It has been pre-trained on a vast corpus using the mask language model loss and the next-sentence loss. Later BERT is used for text classification, text matching and natural language inference. BERT takes the input as a combination of a word embedding consisting of table header and question header. In the paper

Algorithm 1 The construction for question mark vector

```
vector = [0]*question_length
for cell in table do
  if contains(question,cell) then
    start_index = get_index(question, cell)
    vector[start_index:start_index+len(cell)] = 2
    vector[start_index] = 1
    vector[start_index+len(cell)] = 3
    break
  end if
end for
for one in header do
  if contains(question,one) then
    index = get_index(question, one)
    vector[index] = 4
  end if
end for
```

Figure 2.1: Construction of Question mark vector (T. Guo and Gao, 2019)

"Content Enhanced BERT-based Text-to-SQL Generation", a sketch-based approach is described. Two algorithms are there for generating the table and question header(T. Guo and Gao, 2019). Below are the figure 1 and figure 2 for generating the question vector and table header vector.

BERT is used as a representation layer. Total of 6 embedding layers is applied to the BERT model to predict the missing entities of the sketch. These missing entities are mentioned in the paper (T. Guo and Gao, 2019) as below:

- SELECT column
- SELECT agg
- WHERE number
- WHERE column
- WHERE op
- WHERE value

As a final result a SQL query is generated from the provided input by filling the sketch/template.

Algorithm 2 The construction for table header mark vector

```
vector = [0]*header_length
index = 0
for one in header do
  if contains(question, one) then
    vector[index] = 1
  end if
  index = index + 1
end for
for cell in table do
  if contains(question,cell) then
    vector[get_column_index(table, cell)] = 2
  end if
end for
```

Figure 2.2: Construction of Table header mark vector (T. Guo and Gao, 2019)

2.2 IRNet

This approach is a sketch-based and cross-domain. Prediction of the column name from the natural language question is described as the mismatching problem in the paper "Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation"(J. Guo et al., 2019), to deal with challenges, authors have developed an intermediate representation language called SemQL. Instead of using end-to-end synthesizing, an intermediate representation named SemQL is used. SemQL is an intermediate tree representation of the query. Figure 3 shows an example of SemQL. Here the question is divided into two parts, select-part and filter-part(where clauses).

After generating the SemQL, schema linking takes place. The goal of schema linking in IRNet is to recognize the columns and the tables mentioned in a question and assign different types to the column based on how is it mentioned in the question.

Schema linking is an instantiation of entity linking in the context of Text-to-SQL, where these entities are columns, tables and cell values of the database.

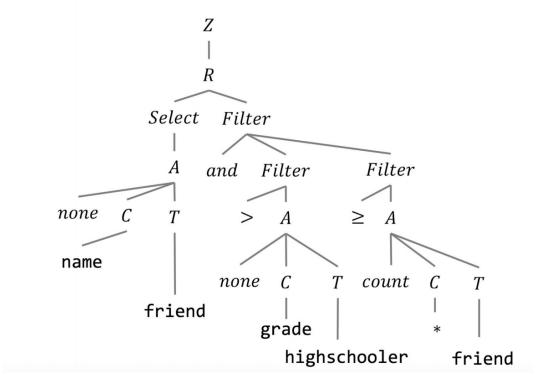

Figure 2.3: SemQL tree

Schema linking processes the question and classifies table column and value.

For classification, the question is divided into n-grams with a length of 1 to 6. N-grams helps to achieve better word matching.

Overall, this model synthesises the query from SemQL. Using a decoder, SemQl can be generated. This decoder interacts with three types of actions to generate a SemQL query, including APPLYRULE, SELECT COLUMN and SELECT-TABLE. APPLYRULE(r) applies a production rule r to the derivation tree of a SemQL query. SELECTCOLUMN(c) and SELECTABLE(t) select a column c and a table t from the schema, respectively.

This SemQL will be the input for the model. During the training of the model, SemSQL is converted into word vectors. This encoder is made of LSTM networks. After encoding the words, the model encodes the schema. Schema encoder takes column names as input, and as an output, it gets a context vector. This vector adds a context from the natural language to the column name. On finding all the missing parts of the sketch, it fits it in the sketch and query is generated.

2.3 TypeSQL

TypeSQL is a sketch-based approach. In this approach, the natural language question is preprocessed before passing it to the model. The whole model can be split into three sub-problems.

TypeSQL is a sketch based approach. In this approach, the natural language question is preprocessed before passing it to the model. The whole model can be splitted into three sub-problems.

- Preprocessing
- Input encodding
- Slot filling

Preprocessing: The natural language questions are converted to n-grams with the length of 2-6. These n-grams are classified in various categories like person, column, float, integer, date and year. Figure 2.4 describes the architecture of the TypeSQL model.

Input encodding: The array of n-grams with its type/category gets encoded using two bi-directional LSTM networks. This encoding makes the tokens order-free from selecting the relevant column name.

Slot filling: Six different models to fill the slots in the sketch is made for a different entity like $SELECT_COL, $COND#, $COND_COL, $AGG, $OP and $COND_VAL. This model has also been tested on WikiSQL.

2.4 One-Shot learning

Most of the approaches work with WikiSQL dataset for Text-to-SQL generation. These approaches can generate simple SQL queries and can not generate a complex query like a SQL query with joins or nested queries. Template-based models and sequence-to-sequence models are proposed to support complex queries. This limitation is dealt with a new approach mentioned in the paper ”One-Shot Learning for Text-to-SQL Generation”(Lee et al., 2019).

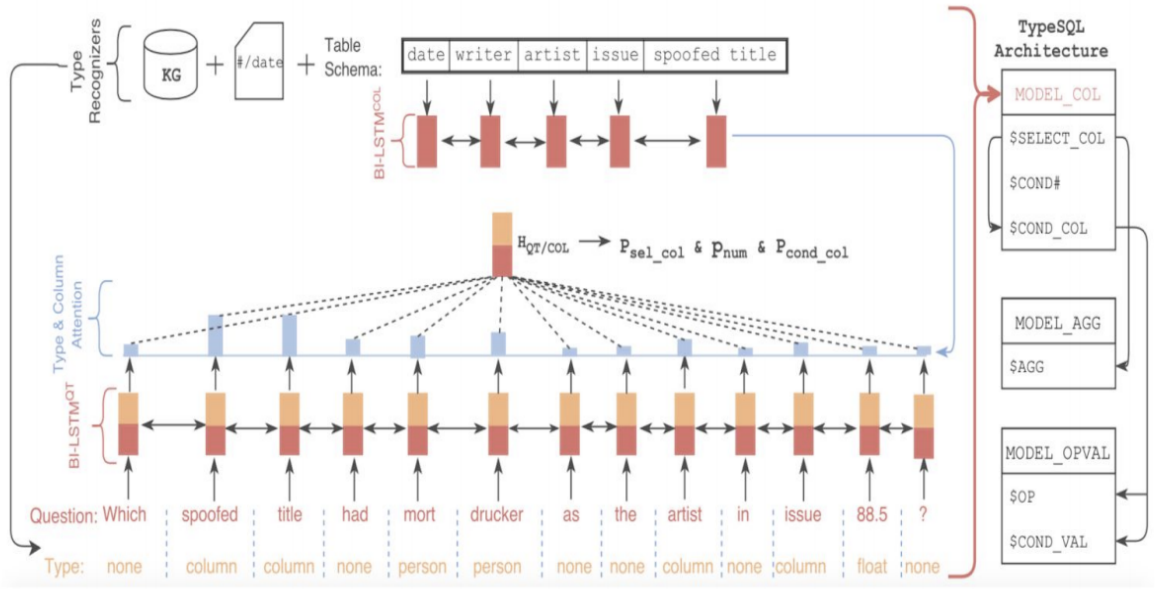

Figure 2.4: TypeSQL ArchitectureYu et al., 2018

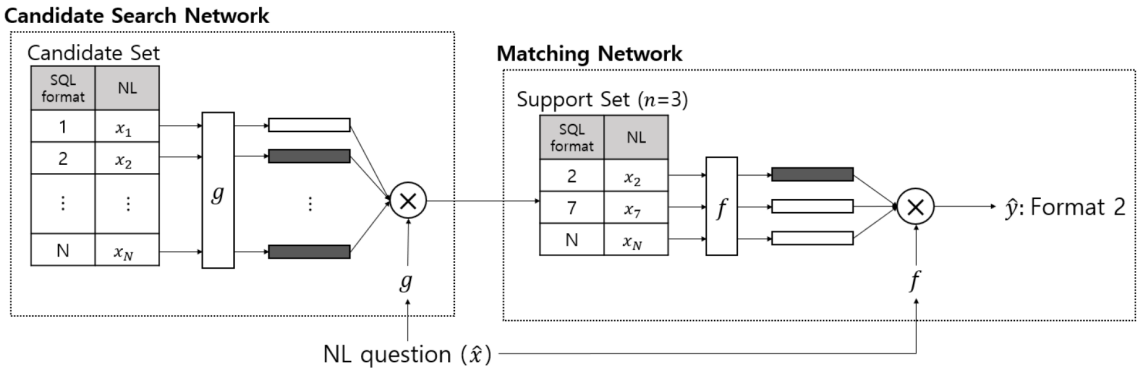

Figure 2.5: Template matching Lee et al., 2019

The structure of the model consists of two main parts, as below:

- Template Matching
- Slot filling

First, it classifies an SQL template for a given natural language question and then, fills the variable slots of the predicted template. This architecture is based on an idea similar to the template-based model of Finegan-Dollak (Finegan-Dollak et al., 2018). However, the previous model requires a number of examples for each template and needs retraining to support new templates of SQL. It applies one-shot learning so that the model can learn a template with just a single example. Moreover, this model does not require any additional training to support new SQL templates (Lee et al., 2019). This model is the most versatile one so far.

In the template matching phase: The first step is Candidate Search Network(de Oliveira, da Silva, and de Moura, 2015). It has a candidate set which comprises sample pairs of natural language questions and their labels (SQL templates), by sampling one example pair from each of the whole classes. Candidate search provides pointers which help to classify the template. It can generate top-n templates for the given English question.

Figure 2.5 describes the template matching phase.

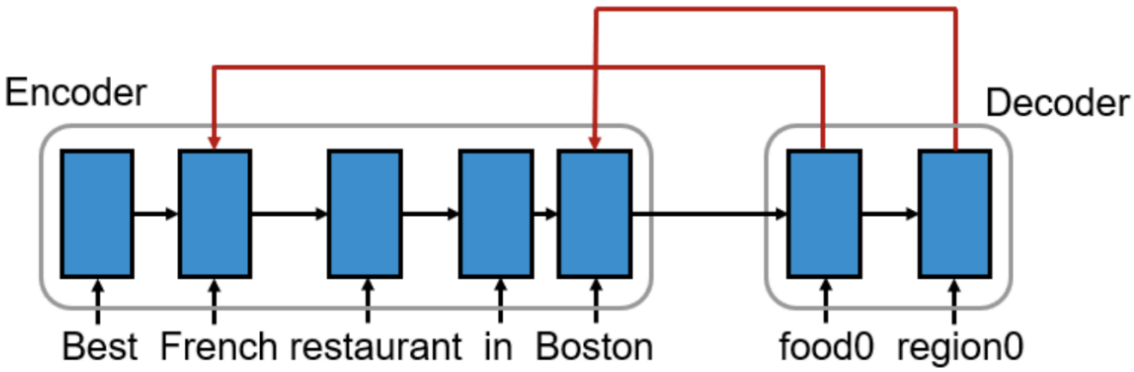

Figure 2.6: Encoder & Decoder Lee et al., 2019

In the slot-filling phase: A bi-directional LSTM as an input encoder and a uni-directional LSTM as an output decoder is used. This helps to get the tokens which are suitable for a slot and relevant to the natural language question.

Figure 2.6, roughly describes the encoder and decoder, which is executing the slot-filling part.

2.5 HydraNet

HydraNet breaks down the problem into column-wise ranking, decoding and assembling the column-wise outputs into a SQL query by inflexible rules. In this approach, the encoder is given a natural language question, and one individual column, which aligns with the original tasks BERT/RoBERTa are trained on, and hence it avoids any ad-hoc pooling or additional encoding layers which are necessary for prior approaches. This approach

looks for the ways to leverage the BERT model to achieve text to SQL-query generation

The approach can be divided into three phases as below:

- Input Representation
- SQL Query Representation
- Column Ranking

Input Representation: The input vector is created by paring the question with each candidate columns. This also contains datatype of the respective column. This vector is a sequence of tokens which later on fed into the network.

SQL Query Representation: The input gets split into two categories, objects related to a specific column and object(s) which is not related to any column.

Column Ranking: Columns are ranked to decide the place of the column in the SQL-query. Three types of ranks are generated Select_Rank - column for select clause, Where_Rank - column for where clause and Relevence_Rank - indicates the relevance of the column to the SQL-query.

3 Discussion

3.1 CNN VS RNN

It is vital to select the best architecture for the new approach. Recurrent Neural Network(RNN) has an advantage over Convolutional Neural Network(CNN) because RNN is good with sequential data and is fast in comparison to CNN. RNNs are ideal for text and speech analysis. CNN works with fixed size input-output. RNN is flexible when it comes to input/output size. Most approaches use LSTM with RNN and Bi-LSTM is also being used for encoding and classification purposes.

3.2 Sketch vs Sequence

All of the approaches can be divided into two categories. One is sketch-based, and second is sequence generation or classification. Sketch-based approaches go with template structure, in which the model tries to fill the slots. Generating SQL query as a sequence of words for the given question is more appropriate for the problem of generating SQL from natural language instead of slot-filling.

Sketch-based approaches make it somewhat fixed and domain-specific. On the contrary, the sequence to sequence models is more flexible when it comes to it being domain independent. IRNet, TypeSQL and One-shot learning is sketch-based approaches.

One-shot learning is a good option among the other sketch-based approaches as it classifies and selects the template for the given question, and a new template can be added without training the model again.

Both methods have their pros and cons. Sketch-based approaches are neat and clear to understand and implement. Where sequence-to-sequence approaches are complex to implement and needs more time to implement with compare to sketch-based approaches.

3.3 Word embedding

In different approaches mentioned above, uses different types of word-embedding wherever necessary. Methods of word embedding also have different effects on the overall function of the models.

BERT based approach has a word embedding algorithm. It generates the vector with the help of the question and table header, which is being passed to the LSTM layer to get encoded. Word embedding + BERT approach has execution accuracy of 89.2%, and it highly relies on the word embedding.

TypeSQl generates n-grams and classifies among different categories. Later on, this vector gets encoded using the LSTM network and fed to the model.

HydraNet creates a vector out of the natural language question where each column name is paired with the given question. This vector also contains the data type of the column, which makes the vector useful for SQL-query generation.

IRNet does not use word embedding; this approach uses the method of the intermediate representation of the query called SemQL. The SemQL is a tree presentation of the natural language question split into to sections one is a select part, and other is filter part. Both parts contain a sub-tree which represents the question as an intermediate SQL keyword's tree. This representation is being used for the synthesis of the SQL-query.

Similar to IRNet, One-shot-learning approach also does not rely on the word embedding.

3.4 Number of sub-models

The number of sub-models can have an impact on the overall efficiency and time/cost of the approach. The more number of models means more training or more complexity. Sub-models are necessary for sketching mechanism as a different model is generating a specific word to fill in the slots.

BERT based approach uses six sub-models for the embedding of the natural language question. The generated output from these models is used to fill-in the sketch's slots. TypeSQL also uses six models with 2 Bidirectional LSTM networks each to classify different entities for the SQL query. The number of neural networks is one of the parameters by which one can judge the approach.

IRNet and One-shot learning approaches use two sub-models for slot-filling. HydraNet uses three sub-models for sequence generation and column ranking.

3.5 Domain independence

Domain independence is the most crucial part of the approach. As this literature review evaluates different models to decide the approach for Text-to-SQL generation for software questions, the approach should be domain-independent as there is no specific dataset for such scenario.

Among all the mentioned approaches, IRNet and One-shot learning have the potential of being useful in the task of Text-to-SQL generation for software questions domain.a

Other approaches have used only WikiSQL as the dataset with compare to IRNet, which uses Spider dataset for evaluation. In the next section, datasets are discussed for more understanding.

One-shot learning approach uses four different datasets which indicate its flexibility to operate on a different domain.

3.6 Dataset

Most approaches have achieved the benchmark performance on WikiSQL dataset. The limitation of the WikiSQL dataset is that it does not contain nested queries, queries with joins and other complex queries, which makes WikiSQL dataset less effective to achieve a robust model for Text-to-SQL generation.

On the contrary, Spider dataset has more diversity when it comes to the complexity of queries and domains. No dataset is available for the specific use-case of software questions.

4 Conclusion

4.1 Efficiency

BERT based model achieves the improvement in finding out the operator and value for the SQL query. It works well with WikiSQL dataset. TypeSQL and HydraNet follow the same path in the context of the dataset and have similar results. This efficiency is of no use for the topic of this literature review as it does not match the criteria of the domain independence.

IRNet has achieved good efficiency with Spider dataset, which makes it a suitable approach or a reference for the desired solution.

As the number of submodels increases, the errors increases when used in the slot filling mechanism as these submodels are not interconnected. So a model is not aware of the other models and can generate the same results as the other model, and this situation can fail the slot based mechanisms.

4.2 Versatility

Sequence to sequence models is more versatile instead of slot filling approach sequence to sequence-based approach should be proposed for the desired solution.

So far, Spider dataset is more compatible for cross-domain approach and should be considered for a cross-domain approach.

4.3 Approaches and parameters

The approaches are assigned the parameters which can be helpful to evaluate and compare them. These parameters are the number of sub-models, type of approach, use of word-embedding and domain independence. These parameters are broadly described in the discussion.

The table below shows the different parameters of the above mentioned approach.

Approach	No. of Sub-models	Type of Approach	Word-embedding	Domain independence
BERT	6	Sketch	Yes	No
IRNet	2	Sequence	No	Yes
TypeSQL	6	Sketch	Yes	No
One-Shot-Learning	3	Sketch	No	Moderate
HydraNet	2	Sequence	Yes	No

The table below shows the different datasets used by the above mentioned approach.

Approach	Dataset
BERT	WikiSQL
IRNet	Spider
TypeSQL	WikiSQL
One-Shot-Learning	Advising, ATIS, GeoQuery, Scholar
HydraNet	WikiSQL

4.4 Notes

- IRNet is the most suitable approach for text-to-SQL generation of software questions.
- Sequence-to-sequence approach should be chosen over sketch based.
- Spider dataset is the most versetile and can help to train the models which can achieve text-to-SQL generation of software questions.

Bibliography

de Oliveira, P., A. da Silva, and E. de Moura (2015). "Ranking Candidate Networks of relations to improve keyword search over relational databases." In: *2015 IEEE 31st International Conference on Data Engineering*, pp. 399–410 (cit. on p. 9).

Finegan-Dollak, Catherine et al. (2018). "Improving Text-to-SQL Evaluation Methodology." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. DOI: 10.18653/v1/p18-1033. URL: http://dx.doi.org/10.18653/v1/P18-1033 (cit. on p. 9).

Guo, Jiaqi et al. (2019). *Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation*. arXiv: 1905.08205 [cs.CL] (cit. on p. 5).

Guo, Tong and Huilin Gao (2019). *Content Enhanced BERT-based Text-to-SQL Generation*. arXiv: 1910.07179 [cs.CL] (cit. on pp. 4, 5).

Lee, Dongjun et al. (2019). *One-Shot Learning for Text-to-SQL Generation*. arXiv: 1905.11499 [cs.CL] (cit. on pp. 7, 9, 10).

Yu, Tao et al. (2018). *TypeSQL: Knowledge-based Type-Aware Neural Text-to-SQL Generation*. arXiv: 1804.09769 [cs.CL] (cit. on p. 8).