Project Proposal
Semantic Parsing Natural Language into Relational Algebra

Ruiyang Xu, Ayush Singh and Xun Peng
CS6120 NLP Fall 2017
xu.r@husky.neu.edu
singh.ay@husky.neu.edu
peng.xu@husky.neu.edu

1 Introduction

The past decade has witnessed the emerging of big data and data science. Meanwhile, people have created tens of thousands of data. Relational databases still serve as a major storage and management solution to those data, which also means SQL remains as the most commonly used query language to access those data. However, on the other hand, people, who want to manipulate a relational database, still need to invest a vast amount of time to learn SQL. But SQL as a language itself, which, given the flexibility and complexity of a programming language, is not so easy to master. Fortunately, recent progress in Natural Language Processing (NLP) opens a new door to solve this trouble. NLP gives the machine the ability to understand human language and semantically parsing the language into the SQL query. Via this new technology, people who want to use relational database would not need to learn SQL anymore, instead, they can just tell the machine what they want and let the machine generate the SQL code for them.

We shall see later that direct translating of an English description to a SQL query would not be so effective (Zhong et al., 2017). A better approach is to translate it to a well-defined intermediate logic form (usually a formal language of relational algebra) and generate SQL from that logic form. However, generating a full-fledged runnable and correct SQL from an incomplete/suspicious logic form might still be a challenge as a course project. Since the English description would not be so accurate, there would be two major issues in this task: 1. figuring out the correspondence between English phrases and SQL keywords (such as 'How many' could be an implication of COUNT aggregation). And 2. determining the correspondence between words in English description and SQL schema (for example, the word 'paper' is used in English description while we only have the word 'publication' in the schema). We would like to do a further investigation on these two problems in this research.

2 Related Work

Researches in SQL synthesis can be divided into three methodologies. The first one is from the program synthesis community, which tries to tackle this problem via inductive inferences like Scythe (Wang et al., 2017). This method usually needs input-output examples to help to generate counterexamples. Nevertheless, pure inductive program synthesis cares less about the English description of the query problem itself. The second one is from the deep learning community, which tries to leverage the power of the deep neural network to somehow figure out the hidden probabilistic structural relationship between the input English descriptions and the output SQL queries. Since this method often treats the synthesis problem as a sequence to sequence translation problem, it cares less about both of the semantics of the input English description as well as the output SQL query (and hence usually gets a low accuracy). The third one is from the NLP community. It utilizes the NLP technology to retrieve semantic information from the input description, meanwhile, it uses those information to assist the inference process during query synthesis. Since this method both take semantic of descriptions and SQL queries into consideration, it usually got higher accuracy than the pure sequence to sequence deep learning.

SQLizer (Yaghmazadeh et al., 2017) is our primary inspiration that starts by using semantic parsing from the English description into an intermediate logic form, a query sketch (which is essentially a relational algebra with holes). This sketching technique allows them to utilize technologies from program synthesis community (a type theory guided sketch completion). Yet the authors haven't further discussed the shortcoming and regression performance of SQLizer on medium to large datasets with complex queries rather than tested it on a handcrafted small dataset. Therefore the scalability of their method is still questionable.

Several rules based semantic parser has been surveyed by Kate et. al. in (Kate et al., 2005), in which they have used learned transformation rule to parse English U.S.-geography questions into a database query language (Prolog). The theory of semantic tractability has been proposed by Popescu et. al. in (Popescu et al., 2003), in which they defined a measurement of the complexity of the English description so that their system can recognize those complex descriptions and ask for further details about them.

A more sophisticated way provided by Giordani et. al. (Giordani and Moschitti, 2012) is to generate candidate queries via lexical dependencies in the question and the database schema and build a set of plausible clauses enriched with meaningful joins. Those clauses can later be combined into different SQL queries and then ranked for evaluation. One feature of their work is that they didn't use SQL query to train their model, but only English description plus database schema.

Leveraging the recent progress in deep learning, Seq2SQL (Zhong et al., 2017) introduced another way by generating queries directly from input natural language descriptions via a sequence to sequence generative model. They also performed a reinforcement learning with synthetic gradient injection (Schulman et al., 2015). However, since lacking contextual learning of the syntax in a SQL query and the semantics of the English description, Seq2SQL performs not very well (with accuracy 57.6%). And even with the reinforcement learning injection, the accuracy only be pushed up by 2.7%.

3 Overview

In the first phase of this project, we would like to do some reading and gain some background knowledge on semantic parse as well as some state of the art implementation of the semantic parse. After we have acquired enough knowledge, we will move into phase 2 and start to implement our own semantic parser. Regarding our semantic parser, we have 4 main ideas want to experiment:

1. Unlike the SQLizer (Yaghmazadeh et al., 2017) in which they train their parser without the schema information, we think schema should also be provided during the training phase (Giordani and Moschitti, 2012) so that the machine will have more contexts on the problem being asked. Specifically, we try to figure out correspondence mapping between the words in the English description and database schema and replace the words in English description with the ones in the schema (using word2vec, SUMO, or syn/word/frame nets). And we would like to see if this preprocessing helps to improve the parsing accuracy.

2. Considering some conditional statements constrained on specific values (such as 'OOPSLA' and '2010'), we would like to see if it is possible to find the correlation between a given value (like '2010') and it's attribution (a table column in the schema, like 'year'). If such correlation can be found, it will be helpful to reduce the problem space during parsing.

3. We want to introduce an incremental parsing/synthesis process so that we can handle quires with a complex structure. Specifically, we try to find the boundary of each clause in the English description and parse each clause locally. Then we will use database schema to decide how to combine those local relational algebras into one whole query.

4. We would like to know if there is a possibility to improve the performance of a deep learning based parser via giving it more semantic information. We believe that more specific features will lead to better learning result. So instead of pouring the whole original English description into a neural network, we would like to extract useful information from it and use that information to train the neural network. So unlike Seq2SQL, we would first like to use encoder-decoder network to learn syntactical structure of SQL and finally use it as a heuristic in our system

After implementing the semantic parser, we will use some existing tool to parse our logic forms (relational algebra) into concrete SQL queries so that we can run it and evaluate the result. We also notice some difficulties in our project which we hope can be resolved in the future research. One is the ambiguity of values, as mentioned in (Yaghmazadeh et al., 2017): for instance, 'OOPSLA 2010' can be regarded as a single word and refer to a specific venue; but it can also be interpreted as two words 'OOPSLA' and '2010', which refers to a venue and the year of the venue respectively. We hope this trouble can be solved in a more sophisticated way (i.e. sampling from the column of a database, and predict which column the given value belongs to) instead of the enumeration. The other one is the creation of new columns which even not exist in the schema. For the time being, we still don't have a good answer to this one, and we hope future research will lead us to some solution.

4 Dataset and Evaluation

We have two benchmarks to evaluate our result. The first one is from the SQLizer team, which is a relatively handcrafted small dataset (only hundreds of queries). The other one is from Salesforce's WikiSQL (Zhong et al., 2017), which has a relatively large dataset. Both of them have well-formed English description and ground-truth SQL query answer.

Apart from defining accuracy based on the number of the correct end result of a generated query, we also want to compare whether the generated query is similar to ground-truth query or different but still generating the same result. Performance evaluation should also be considered when generating queries to evaluate the application domains.

References

Giordani, A. and Moschitti, A. (2012). Translating questions to sql queries with generative parsers discriminatively reranked. In *COLING (Posters)*, pages 401–410.

Kate, R., Wong, Y. W., and Mooney, R. (2005). Learning to transform natural to formal languages. *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05).*

Popescu, A.-M., Etzioni, O., and Kautz, H. (2003). Towards a theory of natural language interfaces to databases. *roceedings of the 8th international conference on Intelligent user interfaces.*

Schulman, J., Heess, N., Weber, T., and Abbeel, P. (2015). Gradient estimation using stochastic computation graphs. *Advances in Neural Information Processing Systems 28 (NIPS 2015).*

Wang, C., Cheung, A., and Bodik, R. (2017). Synthesizing highly expressive sql queries from input-output examples. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 452–466. ACM.

Yaghmazadeh, N., Wang, Y., Dillig, I., and Dillig, T. (2017). Sqlizer: Query synthesis from natural language.

Zhong, V., Xiong, C., and Socher, R. (2017). Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103.*