Intelligent Sketch-based Recurrent Neural Networks Models to Handle Text-to-SQL Task

Youssef Mellah, Zakaria Kaddari, Toumi Bouchentouf, Jamal Berrich and Mohammed Ghaouth Belkasmi
Mohammed First University, LARSA/SmartICT Laboratory, ENSAO, Oujda, Morocco

Keywords: SQL, NLP, RNN, LSTM, GRU, WikiSQL.

Abstract: Databases store a large amount of current data and information, and to access them, users must know a query language like SQL. Therefore, using a system capable of converting a natural language into an equivalent SQL query will make this task much easier. In that direction, the making a system facilitating the interaction with the relational databases is a challenging problem in the field of Natural Language Processing (NLP), and remains a very important area of research. It has recently regained momentum due to the introduction of large-scale DataSets. We present, in this article, our approach based on Recurrent Neural Networks (RNNs), more specifically on Long-Short Term memory cells (LSTM) and Gated Recurrent Units (GRU). We also describe WikiSQL, the DataSet used for training, evaluation, and testing our models. Finally, we present our results of evaluations.

1 INTRODUCTION

Today, a large quantity of information is stored in a relational database and forms the basis of applications such as medical records (Hillestad, 2005), financial markets (Beck, 2000), and asset management customer relationships (Ngai, 2009). However, it is absolutely necessary to know a query language like SQL, to interact with relational databases, which is not obvious to everyone. Therefore, recent research appeared to address systems that map Natural Language (NL) to SQL queries. A long-standing goal has been to enable users to interact with the database through NL (Androutsopoulos, 1995; Popescu, 2003). We call this task Text-to-SQL. In this work, we present our approach based on classifications (Bakliwal, 2011) and recurrent neural networks (Mikolov, 2010), specifically on LSTM (Sundermeyer, 2012) and GRU (Dey, 2017) cells. The idea is inspired by the SQLNet approach (Xu, 2017); in particular, we use a sketch to generate an SQL query from natural language. The sketch aligns naturally with the syntactic structure of an SQL query.

We set up RNN, similar to the traditional sketch-based approaches of program synthesis (Alur, 2013; Solar-Lezama, 2006).

2 RELATED WORK

There is a range of representations for semantic analysis or the mapping of natural language to formal meaning, such as executable programs and logical forms (Zelle, 1996; Zettlemoyer, 2012; Wong 2007). As a subtask of semantic analysis, the Text-to-SQL problem has been studied for a long time (Li, 2006; Giordani 2012; Wang, 2017), and one of the primary works, PRECISE (Popescu, 2003), which translates questions into SQL queries and identifies questions about which it is unsure. (Iyer, 2017) use a Seq2Seq model with human feedback. The community of database has come up with methods that tend to involve engineering manual features and user interactions with systems. Recent work sees Deep Learning (DL) (Cai, 2017) like a primary technique, based on neural machine translation(Castano,1997).

Our work is similar to recent work using DL, precisely RNN with LSTM and/or GRU.

3 DATASET

We operate on WikiSQL (Zhong, 2017), a DataSet for Text-to-SQL task which contains a collection of

Mellah, Y., Kaddari, Z., Bouchentouf, T., Berrich, J. and Belkasmi, M.
Intelligent Sketch-based Recurrent Neural Networks Models to Handle Text-to-SQL Task.
DOI: 10.5220/0010731000003101
In *Proceedings of the 2nd International Conference on Big Data, Modelling and Machine Learning (BML 2021)*, pages 201-205
ISBN: 978-989-758-559-3
Copyright © 2022 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

questions, corresponding SQL queries, and SQL tables. WikiSQL, which is the largest hand-annotated semantic analysis dataset to date. This DataSet is more prominent than other datasets that handle the Text- to-SQL task, either in terms of number of tables or examples. Each table only exists in a single set, either the train, the dev or the test set.

Using WikiSQL, the model must be able to not only generalize to new queries, but to new table schema, due to the diversity and the large number of tables that contain. Finally, WikiSQL contains realistic data extracted from the web, with 87,673 examples of questions, queries, and database tables built from 26,521 tables. All SQL queries in WikiSQL respect the format illustrated in the sketch of Figure 1.

```
SELECT $AGG $COLUMN
WHERE $COLUMN $OP $VALUE
(AND $COLUMN $OP $VALUE)*
```

Figure 1: WikiSQL queries sketch.

4 APPROACH

Our approach can be seen as a neural network alternative to traditional sketch-based program synthesis approaches, so we also track location filling. The idea is to use a sketch to generate an SQL query from natural language. The sketch respects the syntactic structure of an SQL query; neural networks are set up each predicting a component of the request. As shown in Figure 1, the locations that will be predicted are tokens starting with "$". Our proposed pipeline can therefore be divided into six modules ($AGG, $SELCOL, $CONDCOUNT, $CONDCOL, $CONDOP and $CONDVALUE).

4.1 AGG Module

The role of this model is to predict the correct aggregation function, given the user question as input. This is a classification problem. Therefore, the model must select one of the six classes ["", "COUNT", "AVG", "MAX", "MIN", "SUM"], conditioned on user request as input only. Using word embedding, a sequence of tokens is taken by the pattern, which represents the natural language statement. The wrapper is then sent to an LSTM layer whose internal states are first passed to a dense layer with tanh as activation and finally to a dense layer with a softmax function which gives a probability distribution over all the classes. This can be posed as a classification problem in which we have six classes and we choose the one with the probability maximum. Figure 2 shows the conception of this module.

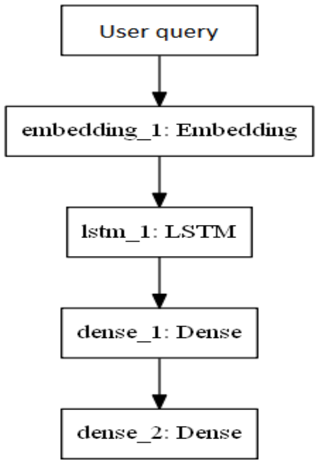

Figure 2: AGG, CONDCOUNT and CONDOP Modules architecture

4.2 SELCOL Model

The goal of this model is to get the appropriate selection COLUMN given the natural language utterance; it is also treated as a classification problem. Given this time the user question and the database schema as inputs, the model returns a column of the table schema. The inputs are converted to embedding, then they are processed by two GRUs according to the hidden states.

Then, we concatenate the outputs, and integrate them into two dense layers, with softmax as the activation function in order to return a probability (score) between 0 and 1 of each column. The column with the highest probability is returned at the end by the model. The architecture of the model is shown in figure 3.

4.3 CONDCOUNT Model

This model is for finding the number of conditions in the WHERE clause. We remark that the most complex query in WikiSQL contains tree conditions, so we treat this need also as a classification problem with four classes: [0, 1, 2, 3]; 0 for no condition, 1 for one condition, 2 for two conditions and 3 for tree conditions (the maximum). This module is considered like AGG module. Figure 2 shows the visualization of the module.

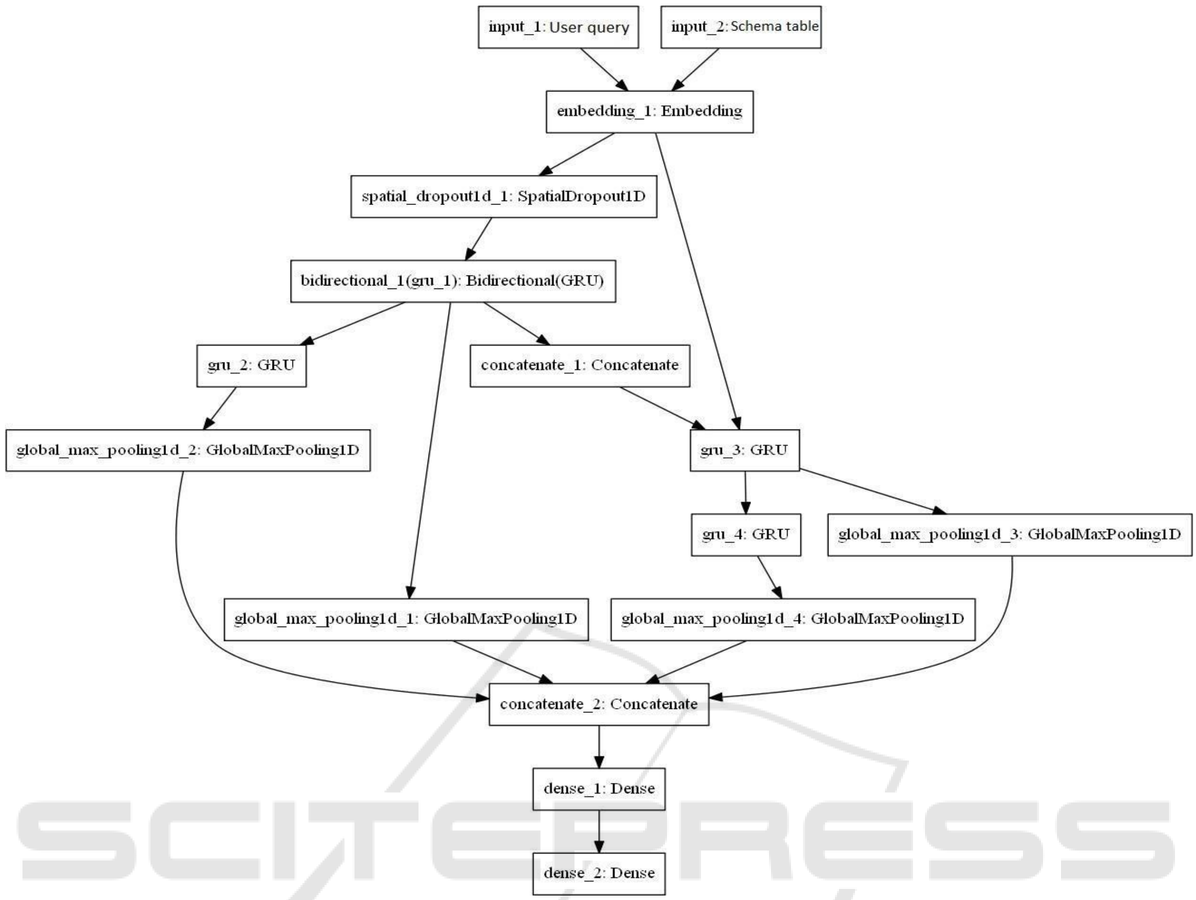

Figure 3: SELCOL and CONDCOL Modules architecture

4.4 CONDCOL Model

For this model, the goal is to find the appropriate column for the condition in the where clause, giving the question and the database schema as inputs. This model is identical to the SELCOL model. The architecture is the same in figure 3.

4.5 CONDOP Model

The function of this model is to predict the correct operation for the condition in the where clause. It is also considered as a problem of classification of three classes: [=,>, <]. This model is identical to the model of prediction of the aggregation function (AGG). For the architecture it is the same visualized in figure 2

4.6 CONDVALUE Model

The goal here is to generate the value of the condition in the where clause. The model takes the user question as input, and returns two outputs: the first concerns the number of words to be taken from the question, and the second concerns the probability of each word to appear as a condition value. The entry tokens are converted to embedding and then pass a bidirectional GRU. The hidden state of the latter is subsequently passed to another GRU. Then the whole is passed to two dense layers, one with relu as an activation function, and the other with softmax. The first dense layer returns the probability of each token in the issue that it appears in the value, and the second dense layer returns the number of tokens to build the final value (maximum 4 according to the DataSet). The architecture of this model is presented in figure 4.

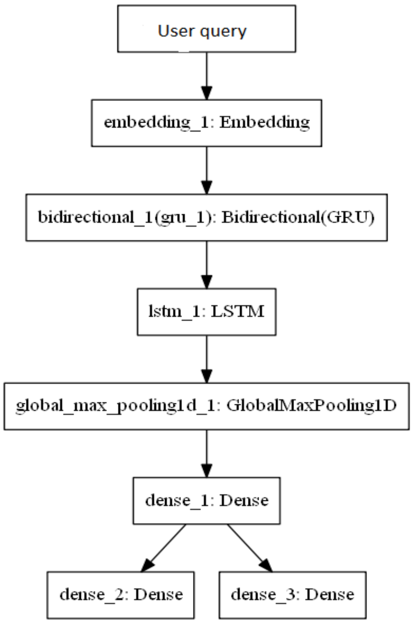

Figure 4: CONDVALUE Module architecture.

5 TECHNICAL DETAILS

This part is devoted to present certain parameters involving RNNs. All models are implemented with Python and the Keras framework. The two imputes used (user question and database schema) are tokenized with Keras's tokenizer, and are represented as a sequence of tokens. These sequences are converted into representative vectors (embedding) using GloVe (Pennington, 2014). Each token is converted into a vector of dimensions 50. The CONDOP, CONDCOUNT and AGG models had two hidden layers, and the others have only one hidden layer. The learning rate and the dimensions of all these hidden layers are respectively 0.2 and 50. The Adam optimizer (Kingma, 2014) is used to optimize the cross-entropy, keeping these hyper-parameters by default. We trained the different models with a batch size of 64 and that over 100 epochs.

6 RESULTS AND DISCUSSION

Table 1 shows results of train and test execution accuracy of each module, evaluated on WikiSQL DataSet.

Module	Train Accuracy	Test Accuracy
AGG	92%	90%
SELCOL	96.5%	95.3%
CONDCOUNT	94.8%	93.3%
CONDCOL	85%	86.8%
CONDOP	91.2%	92.8%
CONDVALUE	45.6%	41.4%
All SQL Query	**42.8%**	**40.3%**

We remark that the smallest accuracies are usually for CONDVALUE, and CONDCOL. In fact, on text-to-SQL task, there is always a problem with column prediction because it is unrealistic that users always formulate their questions with exact column names and string entries.

Also, the VALUES of the conditions are not always mentioned in the question users. Suddenly the model must be able to access the data of the databases (which is out of the scope of our models), this explains the inadequate precision of the value, and which influences negatively on the total precision of the whole SQL query.

We believe that by improving the prediction of VALUES, the precision of all SQL queries will be significantly improved.

7 CONCLUSIONS

We presented our approach, to handle the Text-to-SQL task. We employed a sketch based on Classifications. We used in particular RNN with LSTM and GRU cells. Finally, we showed the results and accuracies of our models.

In future work, we plan to use the Transformer architecture or test the Seq2Seq architecture based on Encoder- Decoder, to improve the precisions, generate more complete and complex SQL queries and evaluate the model on more complex Datasets such as Spider, and see where we can be in term of accuracy.

REFERENCES

Hillestad, R., Bigelow, J., Bower, A., Girosi, F., Meili, R., Scoville, R., & Taylor, R. (2005). Can electronic medical record systems transform health care? Potential health benefits, savings, and costs. Health affairs, 24(5), 1103- 1117.

Beck, T., Demirgüç-Kunt, A., & Levine, R. (2000). A new database on the structure and development of the financial sector. The World Bank Economic Review, 14(3), 597-605.

Ngai, E. W., Xiu, L., & Chau, D. C. (2009). Application of data mining techniques in customer relationship management: A literature review and classification. Expert systems with applications, 36(2), 2592-2602.

Androutsopoulos, I., Ritchie, G. D., & Thanisch, P. (1995). Natural language interfaces to databases-an introduction. arXiv preprint cmp-lg/9503016.

Popescu, A. M., Etzioni, O., & Kautz, H. (2003, January). Towards a theory of natural language interfaces to databases. In Proceedings of the 8th international conference on Intelligent user interfaces (pp. 149-157). Bakliwal, A., Arora, P., Patil, A., & Varma, V. (2011, November). Towards Enhanced Opinion Classification using NLP Techniques. In Proceedings of the Workshop on Sentiment Analysis where AI meets Psychology (SAAIP 2011) (pp. 101-107).

Mikolov, T., Karafiát, M., Burget, L., Černocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In Eleventh annual conference of the international speech communication association.

Sundermeyer, M., Schlüter, R., & Ney, H. (2012). LSTM neural networks for language modeling. In Thirteenth annual conference of the international speech communication association.

Dey, R., & Salem, F. M. (2017, August). Gate-variants of gated recurrent unit (GRU) neural networks. In 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS) (pp. 1597-1600). IEEE.

Xu, X., Liu, C., & Song, D. (2017). Sqlnet: Generating structured queries from natural language without reinforcement learning. arXiv preprint arXiv:1711.04436.

Alur, R., Bodik, R., Juniwal, G., Martin, M. M., Raghothaman, M., Seshia, S. A., & Udupa, A. (2013). Syntax-guided synthesis (pp. 1-8). IEEE.

Solar-Lezama, A., Tancau, L., Bodik, R., Seshia, S., & Saraswat, V. (2006, October). Combinatorial sketching for finite programs. In Proceedings of the 12th international conference on Architectural support for programming languages and operating systems (pp. 404- 415).

Zelle, J. M., & Mooney, R. J. (1996, August). Learning to parse database queries using inductive logic programming. In Proceedings of the national conference on artificial intelligence (pp. 1050-1055).

Zettlemoyer, L. S., & Collins, M. (2012). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. arXiv preprint arXiv:1207.1420.

Wong, Y. W., & Mooney, R. (2007, June). Learning synchronous grammars for semantic parsing with lambda calculus. In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (pp. 960-967).

Giordani, A., & Moschitti, A. (2012, December). Translating questions to SQL queries with generative parsers discriminatively reranked. In Proceedings of COLING 2012: Posters (pp. 401-410).

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

Popescu, A. M., Etzioni, O., & Kautz, H. (2003, January). Towards a theory of natural language interfaces to databases. In Proceedings of the 8th international conference on Intelligent user interfaces (pp. 149-157).

Da San Martino, G., Romeo, S., Barroón-Cedeño, A., Joty, S., Maàrquez, L., Moschitti, A., & Nakov, P. (2017, August). Cross-language question re-ranking. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 1145-1148).

Iyer, S., Konstas, I., Cheung, A., Krishnamurthy, J., & Zettlemoyer, L. (2017). Learning a neural semantic parser from user feedback. arXiv preprint arXiv:1704.08760.

Cai, R., Xu, B., Yang, X., Zhang, Z., Li, Z., & Liang, Z. (2017). An encoder-decoder framework translating natural language to database queries. arXiv preprint arXiv:1711.06061.

Castano, A., & Casacuberta, F. (1997). A connectionist approach to machine translation. In Fifth European Conference on Speech Communication and Technology.

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

Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412