SEQ2SEQ VS SKETCH FILLING STRUCTURE FOR NATURAL LANGUAGE TO SQL TRANSLATION

K. Ahkouk [1]*, M. Machkour[2], K. Majhadi[3], R. Mama[4]

[1]Information Systems and Vision Laboratory, Ibn Zohr University, Agadir, Morocco – karam.ahkouk@gmail.com,
[2]Information Systems and Vision Laboratory, Ibn Zohr University, Agadir, Morocco – m.machkour@uiz.ac.ma
[3]Information Systems and Vision Laboratory, Ibn Zohr University, Agadir, Morocco – khadija.majhadi@gmail.com
[4]Information Systems and Vision Laboratory, Ibn Zohr University, Agadir, Morocco – mathmama15@gmail.com

Commission VI, WG VI/4

KEY WORDS: Language processing, Relational Databases, Natural language translation, Human language to SQL translation.

ABSTRACT:

Sequence to sequence models have been widely used in the recent years in the different tasks of Natural Language processing. In particular, the concept has been deeply adopted to treat the problem of translating human language questions to SQL. In this context, many studies suggest the use of sequence to sequence approaches for predicting the target SQL queries using the different available datasets. In this paper, we put the light on another way to resolve natural language processing tasks, especially the Natural Language to SQL one using the method of sketch-based decoding which is based on a sketch with holes that the model incrementally tries to fill. We present the pros and cons of each approach and how a sketch-based model can outperform the already existing solutions in order to predict the wanted SQL queries and to generate to unseen input pairs in different contexts and cross-domain datasets, and finally we discuss the test results of the already proposed models using the exact matching scores and the errors propagation and the time required for the training as metrics.

1. INTRODUCTION

Many ways to find solutions for Natural Language Processing (NLP) tasks have been deeply studied, among them, the sequential models that were the pillars for tasks like language translation, Text Summarization, etc. On one hand the sequence-to-sequence structure was able to give satisfying result for simple machine learning problems getting rid of the old linguistic techniques and the syntactic methods that lack of precision and suffer when they are exposed to complex inputs and data.

Another class of tasks is natural language translation to database languages like SQL, XQuery, Xpath and others (NL to Query). This kind of models was a big step to find a consistent solution for translation Natural language sentences to Structured Queries like SQL, unlike the traditional works that are based on syntactic parsing. The old models which followed several steps like the part of speech tagging, the creation of the syntax tree, and the use of some intermediate representation based on XML or even the usage of dictionaries of synonyms that helped to make elements substitution, were unable to provide satisfying result against popular datasets.

Unlike the previous cited tasks of NLP, the NL to Query is the task of translating the user question to a database language query using a predefined syntax for the aim to extract data from the database systems. In ordinary NLP tasks, there is a margin of errors that might be tolerated even if the output sentence is not completely correct. For example, if a user translates the sentence in the Ex1 to French using the available models, the output sentence can be straightforward understood even if there are errors in the destination sentence like in the Ex2:

Ex1: 'This is a beautiful house' => 'C'est une belle maison'

Ex2: 'This is a beautiful house' => 'C'est un bon maison'

The essential thing is that the output is understandable by the user even if there are errors. In the other side the predicted queries in the task of NL to Query should not include errors since this may affect the wanted result by the user or can trigger errors in the execution of it.

About 60.48%[+] of the used databases in the world are Relational databases systems. This gives us an idea about how much these kind of systems are spread and how are getting more and more momentum in these days. Lots of domain-experts and developers prefer to interact and use relational databases as they are easy to manage and simple to understand in term of the structure of the data. There are norms and standards that help structuring the data in the right tables and columns in order to keep things organized. Also, there are several languages that help applying those norms. For example, there are Data Definition, Data Manipulation, Data Control, Transaction Control, Data Query languages that are all sub languages of the parent-popular one called SQL. In the same time the extraction of specific data from these tables and columns is not simple for everyone, especially for people who don't know the behind-the-scene of these systems and how they work. Therefore helping to provide a simple way for non-expert

* Corresponding author
+ https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/

users to interact with these systems is the core of the Natural Language to SQL translation task.

In this paper, we present a logical comparison between each method and how the sketch may outperform the already existing techniques in NL to Query

2. BACKGROUND

There have been lots of studies treating the problem of NL to Query. The majority of these works tackle the problem as a semantic parsing task of NLP using a variety of annotated Datasets. In particular, there is WIKISQL (Victor et al., 2017) which is a mono table dataset that contains more than 80000 pairs of natural language sentences and SQL queries. This dataset can't be used for evaluating models since it includes simple structures of queries with only one column in the SELECT clause and one table in the FROM clause with no joins and nested queries. Spider (Yu et al., 2018) is another dataset that covers multiple domains in one corpus. This dataset contains about 10000 Question/SQL query pairs that can be used as a starting point for training semantic parsing-based models. The number of pairs in that corpus is relatively small for a task of NL to Query.

(Victor et al., 2017), (Wang et al., 2017), (Xu et al., 2017), (Yu et al., 2018), (Dong, Lapata, 2018), (Shi et al., 2018), (Hwang et al., 2019), (He et al., 2019), (Liu et al., 2019), (Lee, 2019), (Yu et al., 2018), (Lin et al., 2019), (Bogin et al., 2019), (Yao et al., 2019), (Catherine et al.2018) and (Lyu et al., 2020) as it is shown in (Ahkouk, Machkour, 2019) have all proposed models based on semantic parsing for the generation process. A few of them include linguistic techniques or some Heuristics to enhance the quality of the generated queries or to reduce the output space. For instance the work of (Victor et al., 2017) which is evaluated against the WIKISQL dataset was among the earliest works that introduce the use of sequence to sequence structure to the problem of text translation to SQL. Using that approach combined with the reinforcement learning layer has yielded a 59.4% of execution accuracy. While the majority of the works adopt a sequence to sequence structure for the decoding part, The SQLNet by (Xu et al., 2017) is a different solution based on a sketch filling approach, which provides a simpler way to make the model focus on the necessary parts of the query. The SQLNet was evaluated using the WIKISQL dataset, and based on the Column Attention mechanism which helps to highlight the relevant parts of the user question regarding the columns of the dataset. The model is composed of several sub-modules; each module is in charge to predict one token not a sequence of ones.

3. SEQ2SEQ DECODING AND SKETCH FILLING

3.1 Overview

The differences between the sequential generation and the sketch filling solution can be seen clearly in the decoder part of the model, in contrast with the encoder section where there are often the same structures. In the Figure. 1, an example of the common Encoders structures:

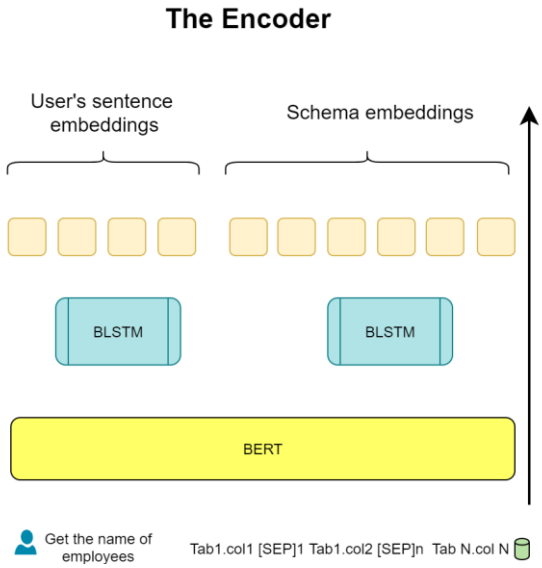

Figure 1. The general structure of an Encoder.

The user question is feed to the model after being tokenized in order to get the words embeddings. This is done using a layer of Bidirectional Encoder Representations from Transformers (BERT) (Jacob et al., 2018). Unlike Word2Vec (Mikolov et al., 2013) which contains static vectors only, BERT is used since it allows the extraction of contextualized word embeddings depending on the whole input. Further operations on the output of BERT are done using Bidirectional Long Short-Term Memory (BLSTM components). This is helpful to better understand and get the adequate representations of each token either in the user natural language sentence or in the schema of the database as it has been done by (Hwang et al., 2019). The user's question and schema use a shared layer of BERT to have closer and inner contextualized word embeddings that will be feed in their turn to the bidirectional LSTMS.

3.2 Seq2Seq

The generation process in the sequential approach is made by predicting one token each time. This includes the reserved tokens of the SQL syntax like: SELECT, FROM, WHERE, etc. Each generated token is also given as input to the next prediction operation and so on until the token <END> is generated as it is shown in the Figure. 2. The output of the encoder, which is the representation of the user's question and the schema of the database, is used through the BLSMs in order to predict the appropriate elements either from the SQL syntax vocabulary, which includes the SQL commands like SELECT, WHERE, GROUP BY, JOIN, etc, or from the set of columns of the database (previously fed from the encoder) or even from the values of the user question using a copying mechanism.
This method incorporates high risk in the generation process. When one token is faulty, and since each token is feed to the next generation operation, the rest of the process is affected, and hence, all the next generated tokens are more likely to be faulty. For example, in third step of the generation, the model should output the token '*FROM'*, if the generated token is different of '*FROM'* then the whole prediction steps should be started from the beginning again until the third token corresponds to the one on the ground truth.

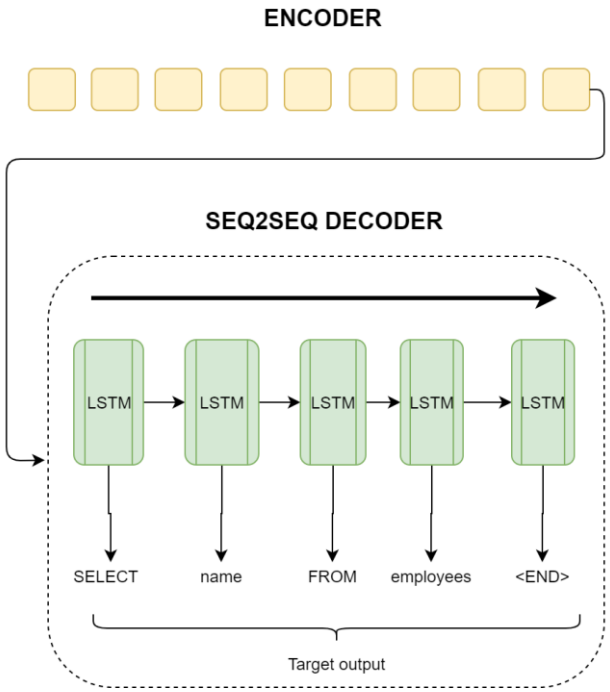

Figure.2. An example of a sequential decoder.

In the case of the mono-columns datasets like WIKISQL it's obvious that the token generated after a column is the reserved word *'FROM'*. Thus, including such reserved words in the generation process impact negatively the model, therefore it reduces the quality of outputs.

The model is seen as one block, thus for the training and if there are errors, the optimization should be done to the whole model, which means more time for the training.

The sequential approach can be helpful in the WHERE section if it is used appropriately, since each element on the WHERE clause depends on the previous one. For example, a solution that adopts sequence to sequence structure for predicting the 3 elements, the column, the operator and the value, can yield satisfying result; or even to use sequence to sequence structure for copying values from the user natural language question, since some queries might include values with more than one token for the same condition.

3.3 Sketch with holes

The first thing that might be noticed is that the generation process in sequential models includes reserved SQL tokens. A better solution is to get rid of these tokens and to focus only on the variable parts in the target query. The static tokens can be added then to the final query regarding the structure of the query.

The same encoder can be used which provides closed representations of the input so the decoder can decode those elements and generate the adequate outputs.

In general, the sketch is a composition of several sub-modules; each one is in focus for predicting the related output. The query can be seen as a sentence with holes, when the holes are the missing parts that the sub-modules should fill.

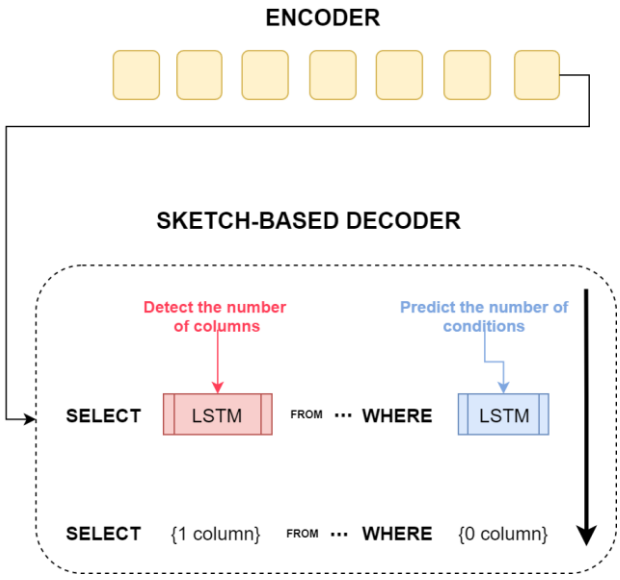

Figure.3. A Sketch-based decoder. The decoding is performed for each independent sub-module.

Before the generation process, the number of elements in each slot should be inferred. The Figure.3 displays the overview of the process of getting the number of columns and the number of conditions in the where clause of a potential query. Each element is treated as a classification problem. For example the number of conditions in the where clause is an N-way classification one, with N as the maximum possible number of conditions in that slot.

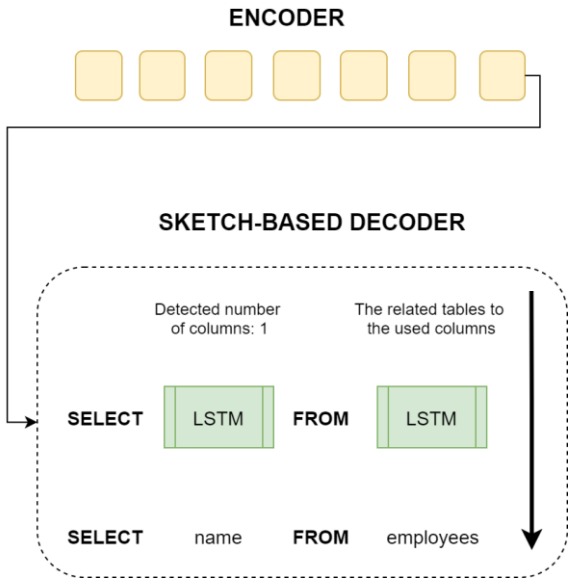

Figure.4. An example of a Sketch-based decoder.

Once the number of elements is detected, we proceed with the elements prediction as it is shown in the Figure.4. Only the important parts are generated in the query and not the whole query in contrast with the sequential approach. For example, in the slot of SELECT, a sub-module is there to focus only on the columns to be generated and the same is done for the FROM

clause. The Figure.5 shows the generation process in the where clause which includes the column, the operator and the value. The advantages of the Sketch-filling solution is that each sub-module is independent from the others, therefore the errors propagation is limited within each sub-module, thing that helps also to reduce the time for the training. If there is an error in one sub-module, only the parameters of this one are optimized not the ones of the whole model.

In this paper we don't go deeper on how each sub-module works since our goal is to highlight the differences between the sequential approach and the sketch one.

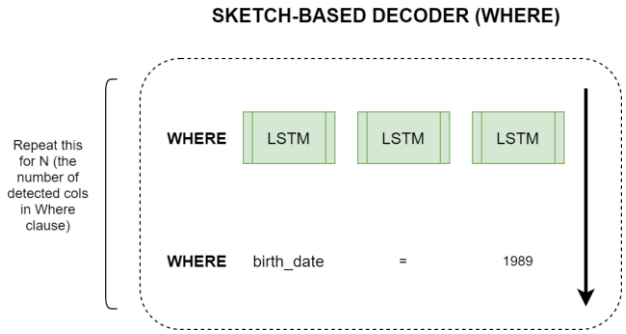

Figure.5. The Sketch-based decoder – where clause.

The Sketch-based decoder can also support advanced SQL structures using multiple sub-modules for the GROUP BY, HAVING, ORDER BY, etc, as it is mentioned in the work of (Ahkouk, Machkour, 2020). In the same time the sketch can be used recursively for predicting the nested queries in the where clause by inferring the structure of the whole query before triggering each sub-module in the model.

4. DISCUSSION

4.1 Models performance

There are many metrics that can be used to understand the differences between the two approaches. To evaluate the performance of each structure we propose to use 2 metrics. The most important metric is the quality of the output. The table bellow shows the scores of only two previously proposed models. We compare these models since they use the same dataset and the two discussed approaches.

Model	RL	Sets (WIKISQL)		Approach
		Dev(EM)	Test(EM)	
Seq2SQL	No	52.2%	50.7%	Sequential
SQLNet + RL	Yes	53.5%	51.6%	Sequential
SQLNet	No	63.2%	**61.3%**	Sketch

Table 1. Performence of Seq2SQL vs SQLNet

The Table 1. Performence of shows a comparison of the Seq2SQL by (Victor et al., 2017) and SQLNet by (Xu et al., 2017). From the table it can be clearly seen that SQLNet, which is based on a sketch method, outperforms the sequence to sequence model. The model predicted 61.3% of queries in the test set of the WIKISQL dataset correctly, while the Seq2SQL model generated only 51.6% with the use of reinforcement learning. In the table we only mention the exact matching (EM) scores not the execution ones since we believe that we might have wrong queries that return the same result as the correct one. For example, the queries below can have the same result if the table employees have no employee with a salary equal to 0:

*Query1: SELECT * FROM employees*

*Query2: SELECT * FROM employees WHERE salary != 0*

Another metric that can be used to evaluate the two approaches is the time for the training and errors propagation.
In the partial test set result from the Table 2, we notice that the Sketch-based model outperforms the other one on all prediction slots. It is obvious that the sequence to sequence suffer for precision especially on the Where clause. This is due to the concept of independence between each slot in the sketch. When the column sub-module makes a wrong prediction, the where sub-modules are kept safe and the error is not propagated to their parameters, thing that requires shorter time for the training for that one sub-module only, in contrast to the sequential models which will try to re-generate the whole query from the beginning, hence the back-propagation is triggered and the parameters are updated for the whole model not only for the faulty parts.

Model	RL	Slots of SQL Query			Approach
		Aggregation	Select	Where	
Seq2SQL	YES	90.1%	88.9%	60.2%	Sequential
SQLNet	No	90.3%	90.9%	71.9%	Sketch

Table 2. Partial comparison between a sequential model and the sketch-based one

The time for training is also impacted by the number of parameters of the model that should be optimized, bigger the model is, longer the time for the training.

4.2 Analysis

When analysing the structure of SQL queries inside the proposed training corpuses like WikiSQL and SPIDER, we can reformulate the problem of text to SQL as the prediction of two main categories of objects. The first one might include the global objects that are predicted to define the structure and the form of the SQL query, while the second group of objects is related to the content of the query.

Global objects: These types of objects include the output that will be helpful to infer 'how our target query is' in term of complexity, grouping, query inclusion, etc. For example we might have a sub-module that predicts the empty structure of the wanted query like:

SELECT $ FROM $ WHERE $ (SELECT $ FROM $) $ (SELECT $ FROM $ WHERE $)

In the above example we have a query with one column or more, one table or more, but we are sure that we have two conditions with nested queries ($ are things to predict). Using this approach of decoupling global objects from the content ones allowed us to get 50% of the final query. Before the training phase, the dataset should include annotations related to

the empty structure of the query; this can be added to the corpus manually or automatically via a script.

Content objects are elements that have relation with the wanted data from the database like columns, operators, aggregation function and values. The prediction of these elements can be done efficiently using separated sub-modules; each module is in charge to predict one component. Note that there is no relation between the two categories of objects as the first class defines the shape of the target query and the second one fill the already constructed query with the adequate elements in a total independency between the components and the clauses.

The concept of using a sketch filling solution allows us as well to apply the NO answer technique for question answer tasks like (Jacob et al., 2018) which is very beneficial for the training process in term of the training time and the quality of the trained model.

5. CONCLUSION

In this paper we presented a logical comparison between the Seq2Seq approach and the Sketch filling one and how the latter one can outperform the sequential models when facing complex and rich SQL queries. To sum up, the Sketch-based models can be good replacements for the previous solutions that adopt seq2seq approaches especially for the task of Natural language translation to SQL. Our Goal is to build a system that can translate natural languages to SQL automatically especially for right to left languages like Arabic using a rich scale of SQL structures.

REFERENCES

Ahkouk, K. and Machkour, M., 2019. Human Language Question To SQL Query Using Deep Learning," Third International Conference on Intelligent Computing in Data Sciences (ICDS), Marrakech, Morocco, pp. 1-6, doi: 10.1109/ICDS47004.2019.8942342.

Ahkouk, K. and Machkour, M., 2020. Towards an interface for translating Natural Language questions to SQL: A conceptual framework from a systematic review', Int. J. Reasoning-based Intelligent Systems.

Bogin, B., Matt, G., Jonathan, B., 2019. Representing Schema Structure with Graph Neural Networks for Text-to-SQL Parsing. arXiv:1905.06241

Catherine, F. D., Jonathan, K., Zhang, L., Ramanathan, K., Sadasivam, S., Zhang, R., and Radev, D., 2018. Improving Text-to-SQL Evaluation Methodology, ACL

Dong, L., Lapata, M., 2018. Coarse-to-Fine Decoding for Neural Semantic Parsing. arXiv:1805.04793

He, P., Mao, Y., Chakrabarti, K., Chen, W., 2019. X-SQL: reinforce context into schema representation. MSR-TR-2019-6 | March 2019. Published by Microsoft Dynamics 365 AI

Hwang, W., Yim, J., Park, S., Seo, M., 2019. A Comprehensive Exploration on WikiSQL with Table-Aware Word Contextualization.. arXiv:1902.01069v1

Jacob, D., Chang, M. W., Lee, K., Toutanova, K., 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805

Lee, D., 2019. Clause-Wise and Recursive Decoding for Complex and Cross-Domain Text-to-SQL Generation. arXiv:1904.08835v1

Lin, K., Bogin, B., Mark, N., Jonathan, B., Matt, G., 2019. Grammar-based Neural Text-to-SQL Generation. arXiv:1905.13326

Liu, X., He, P.g., Chen, W., Gao, J., 2019. Multi-Task Deep Neural Networks for Natural Language Understanding. arXiv:1901.11504

Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781

Shi, T., Tatwawadi, K., Chakrabarti, K., Mao, Y., Polozov, O., Chen, W., 2018. IncSQL: training incremental text-to-sql parsers with non-deterministic oracles. arxiv:1809.05054

Victor, Z., Xiong, C. and Richard, S., 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. arXiv preprint arXiv:1709.00103,

Wang, C., Marc, B., Rishabh, S., 2017. Pointing Out SQLQueries From Text. MSR-TR-2017-45 | November

Xu, X., Liu, C., Song, D., 2017. SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning. arXiv:1711.04436v1

Yao, Z., Su, Y., Sun, H., Yih, W. T., 2019. Model-based Interactive Semantic Parsing: A Unified Framework and A Text-to-SQL Case Study. OSU & Facebook AI Research

Yu, T., Li, Z., Zhang, Z., Zhang, R., Radev, D., 2018. TypeSQL: Knowledge-based Type-Aware Neural Text-to-SQL Generation. The 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics, New Orleans,

Yu, T., Yasunaga, M., Yang, K., Zhang, R., Wang, D., Li, Z., Radev, D., 2018. SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-DomainText-to-SQL Task. arXiv:1810.05237

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., James, M., Li, I., Yao, Q., Roman, S., Zhang, Z. and Radev, D., 2018. A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain. arXiv:1809.08887

Qin Lyu, Kaushik Chakrabarti, Shobhit Hathi, Souvik Kundu, Jianwen Zhang, Zheng Chen. Hybrid Ranking Network for Text-to-SQL. arXiv:2008.04759