Analysis for Natural Language to Structured Language

April 28, 2022

Registration number:	2106850
Project:	Seq2seq translation
Link to GitHub:	http://www.github.com/Virenferns/CE888

Executive summary (max. 250 words)	162
Introduction (max. 600 words)	371
Data (max. 300 words/dataset)	298
Methodology (max. 600 words)	431
Results and Discussion (max. 1000 words combined)	500
Conclusions (max. 500 words)	236
Total word count	2002

Contents

Abstract

I introduce spider which is a large-scale, complicated, cross-domain semantic parsing and text-to-SQL dataset that was annotated by 11 college students. It contains 10,181 questions and 5,693 complicated SQL queries on 200 databases with numerous tables, spanning 138 different domains. It differs from most previous semantic parsing projects in that it use a single database and the same programs in both the train and test sets .[6] To do so, I'll follow the concept of paraphrase generation, which aims to improve the readability of a sentence by concatenating that have related meanings. I propose a framework that combines the effectiveness of the sequence-to-sequence (seq2seq) model to enhance the quality of generated paraphrases. A two-layer architecture is designed. The model's initial layer is an encoder, which will learn to capture long-term dependencies as well as the input sentence's syntactic and semantic characteristics. The data will then be translated into the Decoder layer before being passed on to the model. A seq2seq model with a long short-term memory is used in the second layer (LSTM) .[1]

1 Introduction

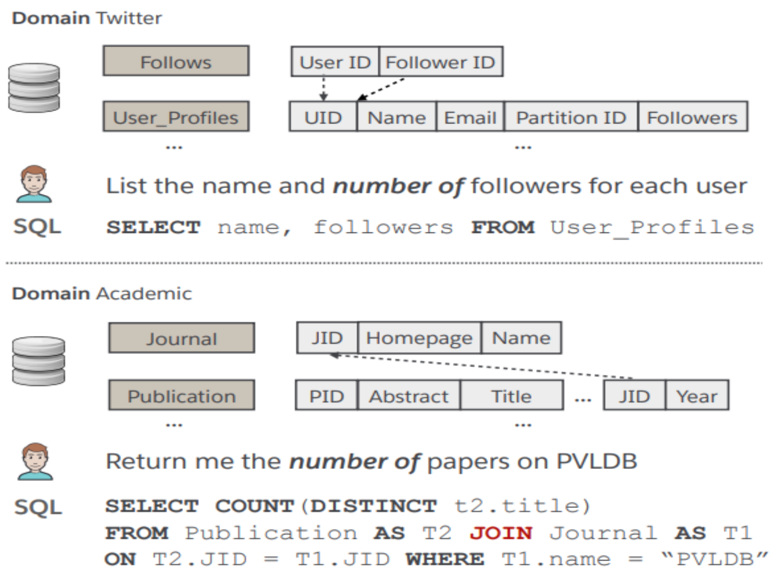
The SQL-to-text task's purpose is to generate human-like descriptions that comprehend the meaning of a given structured query language (SQL) query. One of the most significant tasks in natural language processing is semantic parsing. It necessitates both the comprehension of natural language statements and the translation of those utterances to meaningful executable queries such as logical forms, SQL queries, and Python code. [6]. In past years, there has been a lot of growing interest in Text-to-SQL, which is the process of creating a SQL query from a question. On public Text-to-SQL benchmarks, advanced neural techniques synthesis SQL queries end-to-end and reach more than eighty percent exact matching accuracy, outperforming state-of-the-art approaches on Spider, a newly announced cross-domain Text-to-SQL benchmark. [3] The scenario is difficult because the model must evaluate a query based on a relational database that was not encountered during training and appropriately represent the question intent using SQL logic. Consider the two statements in Figure 1. Both have the intent to count, but the SQL queries are significantly different due to the variations in the target DB architecture. As a result, cross-DB text-to-SQL semantic parsers must accurately model the natural language query, the target DB structure, and the contextualization of both, instead of recalling known SQL patterns. Second, due to various Spider's cross-domain settings, there are a lot of out-of-domain (OOD) terminology. In Spider, for example, 35percent of words in database schemas in the development set do not appear in the training set schemas. In WikiSQL, the percentage is only 22percent. Because OOD words generally lack adequate representations in neural models, the vast amount of OOD words provides another significant issue in predicting columns in SQL queries.Second, due to various Spider's cross-domain settings, there are a lot of out-of-domain (OOD) terminology. In Spider, for example, 35percent of words in database schemas in the development set do not appear in the training set schemas. In WikiSQL, the percentage is only 22percent. Because OOD words generally lack adequate representations in neural models, the vast amount of OOD words provides another significant issue in predicting columns in SQL queries. [4] We test multiple state-of-the-art semantic parsing models to determine the task difficulty. This is a difficult assignment for all of them. In the database split configuration, the best model achieves 12.4percent precise matching accuracy. This indicates that there is a lot of space for growth. [6]

2 Data

The Training dataset was collected to try to predict the clarity of obtaining a structured query by passing in a simple english sentence pertaining to that query. It has 7000 records and is approximately balanced. Prior to preprocessing, i segregated the data into train, validation and testing of which each contained 4200, 1400, 1400 features respectively. The number of columns used after these preprocessing steps was query and question.

2.1 Segregation of Database

I came across columns like query, DB id, question, and tokenized query and question when exploring the dataset. In order to complete this work, I considered columns such as query and question.



Domain Twitter
Follows
User_Profiles
User ID
Follower ID
UID
Name
Email
Partition ID
Followers
List the name and *number of* followers for each user
SQL
SELECT name, followers FROM User_Profiles
Domain Academic
Journal
Publication
JID
Homepage
Name
PID
Abstract
Title
JID
Year
Return me the *number of* papers on PVLDB
SQL
SELECT COUNT(DISTINCT t2.title) FROM Publication AS T2 JOIN Journal AS T1 ON T2.JID = T1.JID WHERE T1.name = "PVLDB"

Figure 1: Two similar-intent queries from the Spider dataset resulted in radically different SQL logical forms on two separate databases. The interpretation of a natural language inquiry in cross-DB text-to-SQL semantic parsing is strongly based in the underlying relational DB schema.

2.2 Pre-processing of the dataset

Taking into consideration the dataset (i.e training) and what our main task is, i have decided to apply the following preprocessing to the question and query list.

- Firstly saved the dataset in a pandas dataframe with an initial shape being (7000, 7000).
- To avoid redundancy in the processed data, we cleaned all whitespaces in the text and decreased the case of every word.
- It is recommended that all special case characters be eliminated. Special characters are removed from the text so that the model can see it in its purest form.
- The result obtained after cleaning the data is divided into different tokens using the word-tokenizer from the NLTK package.
- Performed lemmetization than stemming as lemmatizing a sentence looks at the surrounding text and provides more information.
- Divided the question and query suitable to feed in the model. For that i assigned a start and end tag to the start of the decoder input i.e. ¡start¿ and end of the decoder output ¡end¿

3 Methodology

The seq2seq model is built on a multiple LSTM network. The training of models that can convert sequences from one domain to sequences from another domain, for example natural language language to structured query language, is known as sequence-to-sequence (Seq2Seq) modelling. This model is made up of two primary parts: an encoder and a decoder. The main concept behind this system is to use an encoder to obtain a semantic representation of the source text, which is represented as a series of hidden representations h = [h1, h2,..., hL]. Then, based on these hidden representations, the decoder generates key points. [7]. The data input into the model is compressed into a vector of the desired length, and this compressed vector is stored in the barrier, where it is used by the decoder to forecast the output.

We shall use deep learning in the Sequence-to-Sequence (Seq2Seq) modelling for language translation in this paper. This technique will be used to translate brief English sentences into equivalent structured query language also called as SQL. In this assignment, the sequence to sequence modelling is processed using the LSTM encoder and decoder. Following are the steps replicated in order to achieve this task

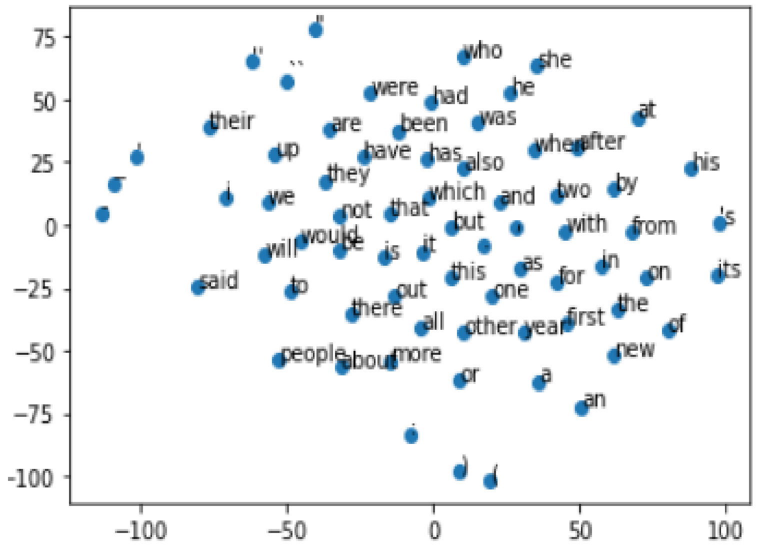


Figure 2: t-distributed stochastic neighbor embedding

- Hyper parameters: The hyper parameters used for this task is batch size of 64 for training, latent dimension of 256 which will be used to match the dimensionality of the encoder and decoder LSTM layer, number of samples to be processed which will be 20000 and setting a maximum sentence length of 50
- The next step would be performing data vectorization which will convert the text into feature vectors. Later tokenize all the words present in their respective ids.
- After the features are set, we define the encoders input and decoder input values which is a numpy array. Thereafter iterating through the matrix we define the decoders target value. In Addition a padding is being set for both the input and output sequence of shape (7000, 38) and (7000, 119)
- next step is to create the word to vector representation for which i made use of glove pre trained word vectors downloaded from 'http://nlp.stanford.edu/data/glove.6B.zip'. The main reason for closing glove embedding is because GloVe has the benefit that, unlike Word2vec, it does not rely solely on local statistics (local context information of words) to generate word vectors, but also incorporates worldwide statistics (word co-occurrence) in our case would be vectorising words like SELECT, ORDER BY etc.
- To show the vectors, I'll use a technique called t-distributed stochastic neighbour embedding, or t-SNE for short. It enables us to reduce the data's 50 dimensions (in my example) to only two. After that, plotting this with a matplotlib scatter plot is just as incredibly easy as shown in 2

4 Results

Using Dense and LSTM layers, the keras model for this problem was the best fit. Two portions of the model layers are dedicated to the encoder and decoder, respectively. The encoder is made up of three layers: input, embedding, LSTM, and hidden and cell state. We can easily utilise a softmax activation for the dense layer's output and train the model with a categorical cross-entropy loss, which is a common classification problem solution, in this manner. In addition, I'll specify a metric accuracy that will inform me of the training's progress at each epoch. I iterated the training over 20 epochs after fitting the data to the model, resulting in an accuracy of 97 percent and a loss of 11 percent. The decoder, on the other hand, has two input layers, both of shape 256, the decoder hidden and cell state values, as well as a shape 1 input layer coupled with a decoder embedding layer with its initial state pointed to the decoders input value. The LSTM and dense layer decoders will come next. In order to translate back the predicted sequence, the word to index method is being called and the item are bifercated into input and target. start and ending of the sentence can be identified by using the tags mentioned earlier ¡start¿ ¡end¿. In order to obtain a desired output every word is passed through a loop and the corresponding word of that index value is captured and saved.

Input Sentence	Query	Predicted Query
count the number of climbers	select count(*) from climber	select count(*) from climber
show the name and location for all tracks location for all tracks	select name , location from track	show the name and location for all tracks

Table 1: Prediction Table

The above table 1 depicts the predicted queries obtained after training the model on the training data. To my observation some queries needed some more training as there are missing keywords that are not seen in the predicted query.

5 Discussion

Based on the output obtained after training the model, I concluded that more in-depth work on the model was required, thus I added the following: Matching to the letter: Letter-to-letter correspondence The expected query is compared to the gold query as a whole to see if it is equivalent. As mentioned in the preceding section, the SQL clauses are examined first. The anticipated question is correct only if all of the components are valid. Because each sentence performs a set comparison, this precise matching metric can solve the "ordering issue." [6]

With Attention: Seq2Seq When it came to text-to-SQL jobs, the author used a very standard sequenceto-sequence paradigm. The target SQL query is directly predicted token by token using a bidirectional RNN with LSTM (Hochreiter and Schmidhuber 1997) cells, while the natural language input is encoded using a bidirectional RNN with LSTM (Hochreiter and Schmidhuber 1997) cells. The encoding phase uses pre-trained word embeddings from word2vec, which are concatenated to the embeddings learned for tokens from the training data, to manage the short amount of data points in some datasets.

Semantic Parsing (Semantic Parsing) is a technique Natural language processing researchers have spent a lot of time on semantic parsing. It converts natural language into a logical representation of its meaning, which is then utilised to answer questions. [5]

6 Conclusions

In this study, we provide Spider, a big, sophisticated, and cross-domain semantic parsing and text-to-SQL dataset that serves both the NLP and DB communities. We define a new demanding and realistic semantic parsing job based on Spider. Experiments on numerous state-of-the-art models on this problem show that there is still room for improvement. [6]

The following two research directions will require us to expand in the future: To solve the OOV problem, we employ a copy method. This method is an extraction method that can help the model provide more precise details. Experiments have shown that this strategy produces excellent outcomes. Humans, too, use a combination of extraction and abstraction to summarise significant points from the source material. By examining the pattern of human summarization habit, we will be able to discover new techniques of extraction and abstraction cooperation in the future. [7]

Experimental results on the challenging Spider benchmark demonstrate the effectiveness of IRNet [3] The concept of execution guidance could be applied to other jobs, such as converting natural language to logical form, which we want to investigate further in the future. [5]

Our analysis has clear implications for future work. Evaluating on multiple datasets is necessary to ensure coverage of the types of questions humans generate. Developers of future large-scale datasets should incorporate joins and nesting to create more human-like data. And new systems should be evaluated on both question- and querybased splits, guiding the development of truly general systems for mapping natural language to structured database queries [2]

References

[1] E. Egonmwan and Y. Chali. Transformer and seq2seq model for paraphrase generation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 249–255, Hong Kong, Nov. 2019. Association for Computational Linguistics.

[2] C. Finegan-Dollak, J. K. Kummerfeld, L. Zhang, K. Ramanathan, S. Sadasivam, R. Zhang, and D. R. Radev. Improving text-to-sql evaluation methodology. *CoRR*, abs/1806.09029, 2018.

[3] J. Guo, Z. Zhan, Y. Gao, Y. Xiao, J. Lou, T. Liu, and D. Zhang. Towards complex text-to-sql in cross-domain database with intermediate representation. *CoRR*, abs/1905.08205, 2019.

[4] X. V. Lin, R. Socher, and C. Xiong. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. *CoRR*, abs/2012.12627, 2020.

[5] C. Wang, P. Huang, A. Polozov, M. Brockschmidt, and R. Singh. Execution-guided neural program decoding. *CoRR*, abs/1807.03100, 2018.

[6] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*, 2018.

[7] Y. Zhang and W. Xiao. Keyphrase generation based on deep seq2seq model. *IEEE Access*, 6:46047–46057, 2018.