Deep Neural Parsing for Natural Language Database Queries based on Schema Matching

Anonymous EMNLP submission

Abstract

In this paper, we propose a schema-matching-based deep neural parser for natural language table/database queries and searches. We design a tagging algorithm based on semantic nearest neighbor and bloom filter to match each word in a query with corresponding schema, and train a sequence-to-sequence (seq2seq) model to translate tagged queries to SQL-like logical forms, which are defined to be used as intermediate states to access into the table. Our model are trained with the help of data augmentation and finally achieves a 59.7% test accuracy on the Wikitable dataset (Pasupat and Liang, 2015), a new benchmark.

1 Introduction

Question-Answering based on knowledge base has been continuously attracting the interests of the machine learning community, since establishing automatic query-and-response feedback is essential to facilitate human-computer interaction, and for machines to achieve higher level of artificial intelligence. Related technologies could be widely used in applications such as natural language user interface, chatting bot, smart housekeeper, etc.

Organizing the worlds facts and storing them in a structured database would be the most effective way to build large-scale knowledge source, since the structure of database is simple and the method for query and searching is scalable to massive knowledge base. Therefore, building a semantic parser for natural language database queries would be one step closer towards machine understanding natural language questions in real-world tasks.

In this paper, we are trying to build a deep neural parser to answer complex questions on structured tables. The general approach of our work is to tag each word in the query with matching schema field, define a set of SQL-like logical rules as intermediate states to the database, and finally train a sequence-to-sequence (seq2seq) model using ($tagged\_query$, $logical\_form$) pairs as supervision. For training and evaluation of our model we use an open source dataset Wikitable (Pasupat and Liang, 2015), which contains 22,033 queries on 2,108 tables. We augment the data for training and development, and use the untouched original set for testing. We report the accuracies of our model and compare with the previous works in section. 5.

2 Related Work

Recent works (Pasupat and Liang, 2015; Liang et al., 2011; Clarke et al., 2010) utilize traditional rule-based features and symbolic processing to construct semantic representations of natural language queries, which is a bottom-up approach attempting to grammatically parse the queries. This approach, however, is hindered by the large dimensions of search space due to the complexity and flexibility in natural language.

Some other previous works, *Neural Enquirer* (Yin et al., 2015) and *Neural Programmer* (Neelakantan et al., 2016), construct distributed representations of both the knowledge base tables and the queries, and executes queries through an **end-to-end memory-based neural network**. With step-by-step attention supervision, *Neural Enquirer*

I. DIRECT QUERIES	
argmax/min <field: 1> <field: 2>	which country has the most gold medals
where <field: 1> {comp} <field_val: 1> select <field: 2>	how many silver medals british won
where <field: 1> {comp} <field_val: 1> argmax/min <field: 2> <field: 3>	which country has the most silver medals among the top 5
where <field: 1> {comp: 1} <field_val: 1> where <field: 2> {comp: 2} <field_val: 2> select <field: 3>	what's the rank of nation who has 3 gold medals and 4 silver medals
where <field: 1> {comp: 1} <field_val: 1> where <field: 2> {comp: 2} <field_val: 2> argmax/min <field: 3> <field: 4>	which nation with 2 bronze and 0 silver has the most medals in total
where <field: 1> {comp: 1} <field_val: 1> select <field: 2> as A where <field: 2> {comp: 2} A argmax/min <field: 1> <field: 2>	which country ranks before japan
II. QUERIES BASED ON CALCULATION	
sum	what is the total number of nations
where <field> {comp} <field_val> sum	what is the total number of nations with no medals
sum/mean <field>	what is the average number of gold medals earned
where <field: 1> {comp} <field_val: 1> sum/mean <field: 2>	what is the average medals won by top 5 countries
where <field: 1> {comp: 1} <field_val: 1> select <field: 2> as A where <field: 1> {comp: 2} <field_val: 1> select <field: 2> as B sum/mean/diff A B	what is the difference in number of bronze medals between india and nepal

Figure 1: The SQL-like logical rules used in this work. Multiple lambda DCS operations (such as Union, Join, Intersection, and Reverse (Pasupat and Liang, 2015)) are simplified into more unified format; Aggregation and Arithmetic types of lambda DCS are also incorporated.

(Yin et al., 2015) achieves a 84% accuracy on their own synthetic dataset, with a small vocabulary (∼300) and limited complexity. While *Neural Programmer* (Neelakantan et al., 2016) with weak supervision of question-answer pairs, achieves a 37.7% accuracy on Wikitable, slightly better than (Pasupat and Liang, 2015). Although memory-based neural networks perform well on small and simple-structured dataset, it is difficult to generalize on large complex dataset with a growing vocabulary and rich natural language queries. Therefore, it would be more realistic to first transform the natural language query into some generic intermediate forms which could reliably access to database.

There are related works (Zettlemoyer and Collins, 2005, 2007; Yih et al., 2015) parsing natural language into lambda-calculus expressions, or (Dong and Lapata, 2016; Jia and Liang, 2016) trying to transform natural language queries into database-specific logical forms. Two of the works (Dong and Lapata, 2016; Jia and Liang, 2016) uses the sequence-to-sequence (seq2seq) model to learn the sequential and hierarchical structure of the logical forms, and (Jia and Liang, 2016; Zhang et al., 2015) further uses data augmentation to increase the size of their training data. However, the parsing processes of these works are largely decoupled from the knowledge databases, which are also limited to small closed-domain tables (e.g. Jobs640, Geo880, and ATIS), thereby giving possibly overfitted results, and hard to generalized to other databases.

3 Parsing to intermediate SQL-like logical forms

3.1 Logical Forms of a Natural Language Query

In general, questions based on a given database table, which are usually natural utterances from human, could be connected to the answer of the query through intermediate logical forms. Previous works invested efforts on parsing to lambda-calculus expressions (Pasupat and Liang, 2015; Liang et al., 2011; Zettlemoyer and Collins, 2005, 2007; Yih et al., 2015); we are more favor of the SQL-like-logical-form rules used to generate the synthetic dataset from *Neural Enquirer* (Yin et al., 2015), which are closer to standard and generic SQL database operations, and could be much easier to apply a wide range of different database after fully trained. For example, one possible logical form of the question "`which nation obtained the most gold medals`" is "`argmax <Nation> <Gold>`". However, there are only 4 types of SQL-like logical forms used in *Neural Enquirer* (Yin et al., 2015), which makes it far from covering the whole complexity of natural language.

Therefore, we extend to a total of 16 different SQL-like logical forms to represent basic database queries, some of which are summarized in Fig. 1. The SQL-like logical forms we defined are the basic types and building blocks, which are easily be captured by seq2seq model. In fact, the trained model is able to generalize the sequential and hierarchical structure of natural language queries, with more complex logical forms. Learning from the first five logical forms in I.DRIECT QUERIES (Fig. 1), the model is able to infer to questions with more than two conditionings stacking in sequence; learning from the nested logical forms at the bottom of both I and II, enables the model to recognize hierarchical patterns and other unseen composite structures.

3.2 Data Augmentation

Since the Wikitable dataset itself does not contain SQL-like logical forms, we need to generate more data without decreasing the language complexity of the original dataset. We widely select nearly 200 Wikitable queries (Pasupat and Liang, 2015) and annotate them with the corresponding SQL-like logical forms. Then we use three methods to augment each query, based on field recombination and synonym replacement, which give us a 15 $\sim$ 20 times increment of the selected data:

(1) For each (*query*, *logical_form*) pair, based on the given logical form, we can recombine the schema field names. For example, the logical form `argmax <Nation> <Gold>` could be augmented to `argmax <Nation> <Silver>` and `argmax <Nation> <Bronze>`.

(2) For each schema field appearing, we could use different possible values that field could take. For example, if we have the query `how many gold did romania won` with corresponding logical form `where <Nation> equal <Nation_val> select <Gold>`, we can alter the value with respect to `<Nation_val>`, such as `how many gold did india won`.

(3) For each query, if there are words that could be replaced by its synonyms (Zhang et al., 2015) and semantic nearest neighbor (Wang and Yang, 2015), it could generate several new queries with the same logical forms. For example, the query `which nation won the most gold medals` could be augmented with another query `which country won the most gold medals`, and both correspond to the same logical form.

We generate an augmented dataset with nearly 4000 (*query*, *logical_form*) pair. The augmented data is then randomly shuffled and divided into training set and development set.

4 Seq2Seq Model based on Schema Matching

4.1 Schema field matching

One problem of previous works (Dong and Lapata, 2016; Jia and Liang, 2016) is that the training is decoupled from the information of schema $S = \{f_1, f_2, ..., f_n\}$. We represent each field f_n with a triple $(id, type, range)$, where $id \in \{1, 2, ..., n\}$, and $type \in \{\text{int}, \text{double}, \text{string}, \text{ordinal}...\}$.

In order to incorporate schema information into the model, we try to tag each word in the original query with a specific token indicating the related schema field. If a word implicates a schema field f_i, then the tagging token will represent the field in the schema `<field:`i`>`; if a word is numerical, ordinal, or string entity, it possibly belongs to the value range of a field $f_j.range$, so the tagging token is `<field_val:`j`>`; if a word does not relate to any fields provided, then tagged with `<nan>`. For example, if we have the query `how many gold medals the country ranked 14 won` with schema containing fields [`Nation, Rank, Gold, Silver, Bronze, Total`], the tagging of the query would be `<nan> <nan> <field:3> <nan> <nan> <field:1> <field:2> <field_val:2> <nan>`, with each token aligned with the word in the query.

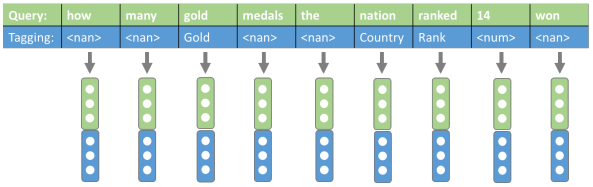

Figure 2: Tagging each input query with schema matching. Each word vector is concatenated with its tagged token when feeding into the model.

To tag the given query with corresponding schema, it involves the following steps: (1) ini-

tialize with all the tagging as <nan>; (2) identify all the words which could be tagged with <field:i>, based on semantic nearest neighbors and edit distance; (3) identify numerical values and name entities, and according to the value type, search for possible fields in the corresponding value range, and tag for all of potential <field_val:j>; if there are more than one, then we take average when feeding into the model.

(a new Figure show the tagging process)

4.2 Attention-based Encoder-Decoder Translation Model

We propose a seq2seq model to parse tagged query to SQL-like logical forms, which is an attention-based encoder-decoder model typically used for neural translation machine (NTM). Our model consists of an encoder which reads encoder inputs $q = x_1, x_2, ..., x_{|q|}$ into a vector representation $h_{|q|}$ and a decoder following behind to generate $y_1, y_2, ...y_{|a|}$ conditioned on the encoding vector $h_{|q|}$. We are using a **greedy decoder**, which also incorporates an **attention mechanism** for better encoder input conditioning (REF). The conditional probability $p(a|q)$ is decomposed as:

$$p(a|q) = \Pi_{t=1}^{|a|} p(y_t|y_{<t}, q) \quad (1)$$

Both encoder and decoder are recurrent neural networks (RNN) with gate-regulated unit (GRU) for each cell. The transition equation for hidden states within multiple layers is

$$h_t^l = GRU(h_{t-1}^l, h_t^{l-1}) \quad (2)$$

where h_t^l is the hidden state for time t at layer l, and $h_t^0 = x_t$. the hidden layer of the topmost GRU h_t^L in the decoder is used to predict the t-th output token:

$$p(y_t|y_{<t}, q) = \text{softmax}(W_o h_t^L)^T e(y_t) \quad (3)$$

where W_o is the output projection weight and $e(y_t) \in \{0, 1\}^{|V_a|}$ is a one-hot vector for computing y_t's probability from the predicted distribution.

Our aim is to train the seq2seq model with $(tagged\_query, logical\_form)$ as supervision.

5 Experiments

Several models Our implementation and improvements are based upon the basic seq2seq model provided by Tensorflow framework. We run our code on Microsoft Azure platform, which is equipped with Tesla M60 GPU capability.

We use Adaptive Momentum Estimation (Adam)

We compare our model against the previous system on the Wikitable dataset (Pasupat and Liang, 2015) we studied.

5.1 Model Comparison

After extensive comparisons with different hyperparameters on the model, we obtain the best model based on query tagging of **a 2-layer 1024-hidden-size network**, with training accuracy of 74.7% and development accuracy of 71.0% on the augmented dataset based on Wikitable, significantly higher than the previous studies (Pasupat and Liang, 2015; Neelakantan et al., 2016).

5.2 Error Analysis

To better evaluate the accuracy of the above four models, we further show the accuracies for the train data and the dev data in Table. 1

Table 1 summarizes the training accuracy and the dev accuracy for the four models for all combinations listed in Table 1. The vanilla model, which does not include extra information about the table, is the least accurate. In contrast, the models using field as the prefix, which strongly enhances the connection between the query and the table, has a persistent high accuracy for almost all hyper-parameter combinations.

5.3 Last hidden state output

We output the last hidden state of encoder for each query (concatenate the states if there are multiple layers) and then use *t-sne* algorithm (REF) to plot in 3-dimensional space. All the vectors are clustering to 16 centroids, which represent 16 different logical forms we have, indicating that the model does a great job categorizing logical forms in semantic space.

6 Conclusions & Future work

In this paper, we present a schema-matching-based seq2seq model to parse natural language database query into SQL-like logical form, which could generically be used to access into

Table 1: Table for accuracies of the train data and the dev data

combination		1	2	3	4	5	6	7	8	9	10	11
vanilla	train	0.390	0.512	—	0.516	—	0.556	—	0.524	—	—	—
	dev	0.361	0.464	—	0.459	—	0.323	—	0.310	—	—	—
entity prefixed	train	0.491	0.501	0.491	0.493	0.566	0.689	0.503	—	0.491	—	—
	dev	0.430	0.460	0.419	0.430	0.512	0.652	0.423	—	0.412	—	—
field prefixed	train	0.586	0.637	0.694	0.694	0.699	—	0.688	0.694	0.690	—	0.612
	dev	0.520	0.580	0.668	0.672	0.688	—	0.633	0.653	0.637	—	0.581
tagging	train	—	0.527	0.537	0.571	0.724	0.747	—	—	0.549	0.567	—
	dev	—	0.449	0.487	0.505	0.692	0.710	—	—	0.492	0.530	—

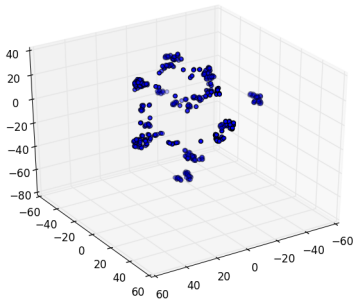

Figure 3: *t-sne* visualization of the encoder last hidden states of fully-trained model.

a wide range of databases. The model achieves a 59.7% test accuracy on Wikitable, which is a new benchmark in this dataset.

References

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Stroudsburg, PA, USA, CoNLL '10, pages 18–27. http://dl.acm.org/citation.cfm?id=1870568.1870571.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. *CoRR* abs/1601.01280.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *CoRR* abs/1606.03622.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. *ACL (1)* pages 590-599. http://arxiv.org/abs/1109.6841.

Arvind Neelakantan, Quoc V. Le, Martín Abadi, Andrew McCallum, and Dario Amodei. 2016. Learning a natural language interface with neural programmer. *CoRR* abs/1611.08945.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *ACL (1)* pages 1470-1480. http://arxiv.org/abs/1508.00305.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 2557–2563. http://aclweb.org/anthology/D15-1306.

Wentau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL*.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2015. Neural enquirer: Learning to query tables. *CoRR* abs/1512.00965. http://arxiv.org/abs/1512.00965.

Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *UAI 2005* pages 658–666. https://homes.cs.washington.edu/ lsz/papers/zc-uai05.pdf.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *EMNLP-CoNLL*.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*.