Computational Linguistics and Intellectual Technologies:
Proceedings of the International Conference “Dialogue 2017”

Moscow, May 31—June 3, 2017

NEURAL NETWORK BASED END-TO-END LEARNING HIERARCHY-AWARE SEMANTIC PARSER FOR RUSSIAN LANGUAGE

Tarasov D. S. (dtarasov@meanotek.io), **Lukina N. M.**, **Izotova E. D.**

Meanotek AI Research, Kazan, Russia

We present neural network semantic parser for Russian language, that utilizes new copying mechanism, intermediate layers supervision and explicit handling of hierarchical nature of the output by means of having RNN blocks operating on different timeframes. Due to the lack of standard Russian dataset for validating semantic parsers, we develop our own small dataset in the domain of logistics and task management and demonstrate that our model can obtains good results on this dataset, despite it very limited size.

Keywords: semantic parser,Russian language, hierarchical representation, neural networks

НЕЙРОСЕТЕВОЙ СЕМАНТИЧЕСКИЙ ПАРСЕР ДЛЯ РУССКОГО ЯЗЫКА

Тарасов Д. С. (dtarasov3@gmail.com), **Лукина Н. М.**, **Изотова Е. Д.**

Отдел исследований компании «Меанотек», г. Казань, Россия

В работе описан нейросетевой семантический парсер для русского языка, в котором используется специализированный механизм копирования, и явное представление иерархической природы выходных данных за счет использование рекуррентных нейросетевых блоков, работающих на разных масштабах времени. С целью проверки работы парсера в работе создан обучающий набор данных на русском языке в предметной области логистики и управления задачами. Показано,

что несмотря на малый объем выборки, разработанный парсер позволяет получать удовлетворительные результаты.

Ключевые слова: семантический парсер, русский язык, иерархическое представление, нейронные сети

1. Introduction

Semantic parsing is a task of translation of natural language phrases into precise logical forms. It has many applications in question answering and other language understanding problems [Liang et al, 2011; Berant et al, 2013;Xu et al, 2016]. Traditional semantic parser are complex software tools, that depend on hand-crafted features, rules, and lexicons and are not easily adaptable for different source and target languages [Tang et all, 2001;Moschitt 2004; Berant et al, 2013].

Recently, neural network based end-to-end semantic parsers were developed [Xiao et al, 2016; Locascio et al, 2016], based on sequence to sequence models, that translate natural language into linearized logical expressions. Such parsers can learn directly from examples and theoretically applicable to any source/target language pairs. However, they require a lot of training examples to work (10 000 and more) [Locascio et al, 2016]. Such datasets are difficult to obtain, as their development is time consuming.

A number of approaches to deal with this problem were developed, including data augmentation by generation of new natural language statements [Jia et al, 2016] and crowdsourcing of paraphrases [Wang et al, 2015], however, they either lack conceptual simplicity (moving complexity associated with classical semantic parsers into data generation stage) or additional expenses and limited coverage of actual query space (paraphrases, written by humans may not represent actual questions, that actual users of real system may ask).

In addition, most of the work in neural semantic parsing to day was done using English as source language. Russian language poses a number of additional difficulties, such as lack of pre-existing datasets, rich morphology and larger freedom in sentence composition.

In this work we develop neural semantic parser for Russian language. In contrast with previous work, that focused on question answering, we consider more complex task of dialog based knowledge acquisition, where system has to extract information from conversational natural language statements in specific domain and then provide answers to questions based on stored knowledge. We also augment our neural model with structural enchantments, that improve parsing accuracy of Russian language compared to to sequence to sequence model, used previously in English semantic parsers [Locascio et al, 2016], while keeping low computation cost.

Specific model improvements that we implement:

1. Simplified copying mechanism, that allows efficient copying of objects and words from source to target text
2. Supervised learning applied to intermediate layers, as well as to output layer
3. Decoder operation on different timescales

We evaluate our model using small manually prepared dataset, and demonstrate that good performance can be obtained despite very limited data size.

While some recent papers consider building semantic parser from query execution results (distant supervision) rather then from actual parses [Liang et all, 2016], we do not follow that approach, because it is very difficult to do it in limited dataset size, and also because we want to work with dialog based knowledge acquisition scenario, where no results are produced by the query aside from database modification.

2. Methods and algorithms

2.1. Output language

Similar efforts in end-to-end semantic parsers utilized various target languages, such as linearized logical forms [Xiao et al, 2016] and lisp-like domain-specific language [Liang et, 2016]. For our work we use subset of Python as target language, where user can formulate queries to graph database using predefined set of functions and classes. Example query is shown in Table 1. While resulting system could be considered as separate query language, all queries can be run using standard Python interpreter after importing our library of functions.

Table 1. Example natural language queries and reference parsing results (Russian and English translation)

Natural language query	Parse
Добавить на склад 3 подушки (Add 3 pillows to the stock)	MatchOne ({"type":"склад"}).Add("name":"подушки", "количество":"3") *MatchOne*({"*type*":"*stock*"}).*Add*("*name*":"*pillow*","*count*":"*3*")
есть ли на складе подушки ? (are there any pillows in stock?)	MatchOne ({"type":"склад"}).child ({"имя":"подушки"}). NotEmpty( ) *MatchOne*({"*type*":"*stock*"}).*child*({"*name*":"*pillow*"}).*NotEmpty*( )

Our motivation for using custom query language instead of existing one, is that we want to investigate how changing syntax of target language may affect quality of neural network parsers (since, complexity of target language can obviously important factor).

2.2. Datasets

Small dataset

Since no common open datasets in Russian exist for our task, we had to prepare our own data. We made two datasets in the domains of logistics and task management. Our small dataset contain 63 entries, each entry representing unique user need and its translation into specific query (no paraphrases of the same query). Dataset was prepared manually by one of authors of this paper.

Extended dataset

Based on our small dataset, rule based recognizer was created that was able to parse exact patterns into queries and system was given to five different users. User queries were collected in two weeks period, resulting in a new dataset of 1344 queries, of which approximately 61% (824) were variations of original queries, 30% were new queries and 10% were task-unrelated comments. All task related natural language queries, that were not parsed by pattern matching, were then manually annotated with correct formal queries, that produced desired outcome. This dataset was split into training, validation and test sets in 60:30:10 ratio.

2.3. Evaluation metrics

Two metrics were used for evaluation: BLEU score [Papineni et al, 2002] between generated and reference parses and accuracy with respect to results. For accuracy, parse was considered correct, if its execution produced result, identical to result of reference query.

BLEU was used mainly to monitor training progress, because this metrics changes more smoothly then execution accuracy.

2.4. Model

Encoder

The encoder converts input vectors $x_1 \ldots x_n$ into a fixed size representation by means of recurrent neural network. A recurrent neural network [Elman, 1990] is a type of neural network that has recurrent connections. This makes them applicable for sequential prediction tasks, including NLP tasks.

In an Elman-type network, the hidden layer activations $r(t)$ at time step t are computed by transformation of the current input layer $x(t)$ and the previous hidden layer $r(t-1)$. Output y(t) is computed from the hidden layer $r(t)$. More formally, given a sequence of vectors $\{x(t)\}$ where $t = 1..T$, an Elman-type RNN computes memory and output sequences:

$$r(i)=\mathrm{f}\left(w^{[r]} \cdot x_i+\mathrm{w}_h^{[r]} \cdot r(i-1)+\mathrm{b}^{[r]}\right) \quad (1)$$

where f is a nonlinear function, such as the sigmoid or hyperbolic tangent function and W and W^h are weight matrices between the input and hidden layer, and between the hidden units.

In our case, hidden layer activations are Long Short-Term Memory (LSTM) cells [Hochreiter et al, 1997]. LSTM cells have capability to retain longer memories then simple Elman RNNs. Encoder representation (concatenation of output and LSTM memory units vectors) servers as input to Decoder.

Decoder with copying mechanism

We observe, that major limitation of seq2seq model is inability to generalize to unseen words. For example, while commands “find all tasks in project ‘some project’” and “find all products at the main storage” are similar and have basically same semantic interpretation in our language, the model that never seen words like

"product" and "main storage" won't be able to generate correct parse. This is even more serious when dealing with task and product titles.

In our model, we used same basic LSTM encoder, but modified decoder to use attention over output words mechanism:

Previous state z of LSTM decoder (concatenation of output and memory units vectors) is first fed into fully connected layer h of size m (eq. 2):

$$h(z) = tanh\left(w^{[h]} \cdot \mathrm{z}+\mathrm{b}^{[h]}\right) \tag{2}$$

Assume that q_i is n-dimensional vector, corresponding to projection of i-th candidate output word. Sequence $q_{i:i+j}$ is calculated by concatenating each element q_i with vector h (eq. 3)

$$q_{i:\mathrm{i+j}} = (h \circ q_1) \circ (h \circ q_2) \circ \ldots (h \circ q_n) \tag{3}$$

Values of the next layer u are computed by applying one dimensional convolution to sequence q with a bank of k filters of length $n+m$ with stride $n+m$ to the sequence q, following application of single filter of the same size.

The resulting vector u thus has the size of candidate word dictionary and candidate word in then selected by applying sofmax function (4) and then taking argmax of vector y (See Fig. 1 for overview of entire process).

$$output(u)=\mathrm{softmax}\left(\left[y_{1,}y_{2,\ldots}y_{\mathrm{n}}\right]\right) \tag{4}$$

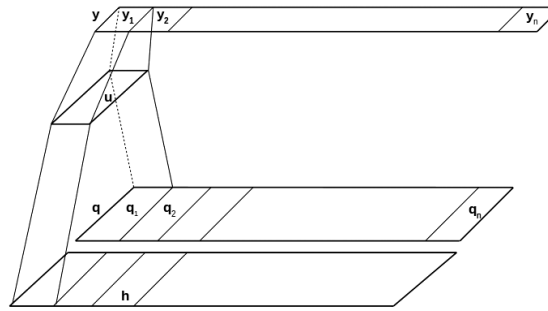

Figure 1. Overview of proposed decoder structure

Pointer networks [Vinyals et al, 2015] were proposed with the aim of solving the same problem, but using more complex attention-based mechanism over recurrent encoder states. Our solution is simpler, and use convolutional networks, so it works faster then pointer-networks approach, and operates directly is source words space, rather then in decoder space.

In our model, layer h produces a "guess" of what could be embedding of output word as a point in vector space, and layer u selects output word out of candidate words list. Since candidate word list is in fact unordered set of possible choices, we replaced recurrent encoder with one dimensional convolution operation. We also do not need separate softmax layers for copy operation and regular vocabulary, because all candidate words are put in a single list.

Generation of candidate words

Instead of using full vocabulary at each generation step (generation of subsequent word), we apply following algorithm to determine list of candidate words:

1. Add words from fixed list of functions and special tokens like "(".
2. Add all words from user query in their original and stemmed forms
3. Remove all tokens whose generation will result in syntax error

Therefore list of words that can be generated at the next step is not static vocabulary but dynamic list that depends on output of syntactic query parser, that checks if result is syntactically correct query.

Supervison of intermediate layers

In our initial experiments, we found, that standard log-loss function over target words does not produce optimal results, because convergence of training was slow and resulting model had poor abilities to generalize over unknown words. We therefore changed error function to sum of error from target word selection with euclidean distance between vector h and projection of target word, forcing vector h to be similar with target word vector. This change lead to significant improvement in training speed and model capability to use unknown words.

Handling different timescales

We noticed, that with long parses, that contain application of more then two graph query functions, our model has difficulty with determining correct sequence of operations. Our hypothesis is that model have problems with keeping information about sequence of previously applied functions, because of there exists a lot of intermediate words that need to be generated (function arguments, argument names and syntactic elements). This problem is particularly evident when arguments themselves are multi-word statements, like titles of objects, or descriptions, that need to be put in the database verbatim.

To solve this without increasing model size or introducing complexity with additional attention over input mechanism, we implemented a version of Clockwork RNN model [Koutnik et al, 2014]. In Clockwork RNN, certain hidden units are only active on certain time steps, while on intermediate timesteps, their state is not updated, allowing model to operate on different timescales. In [Koutnik et al, 2014] timescales are fixed, based on a timestep number. In our situation, we have 3 important levels in hierarchy—function sequence level, arguments sequence level, and single argument level. Transitions between levels can be identified by monitoring output token sequence for arguments and function separators (like '.' and ',' symbols), forcing updates into hidden states of units that are designed for specific timescale.

Table 2. Examples of sequences that are generated and visible on different timescales for query *MatchOne*({"*type*":"*stock*"}).*Add*("*name*":"*pillow*", "*count*":"*3*"). *"<start>"* indicates beginning new sequence on given timescale

Timescale	Sequence visible to dedicated units
function level	<start> MatchOne → Add
arguments sequence level	<start> type → stock <start> name → count
single argument level	<start> stock <start> pillow <start> 3

This way we can explicitly model hierarchical structure of the output and introduce implicit task knowledge in the model, without the need to stacking multiple layers of convolutional or LSTM layers.

In this paper we will use (*F/S/A*) notation to describe specific configuration of model, where *F*—number of units dedicated to processing function level timescale, *S*—number of units dedicated to processing of arguments sequence timescale and *A*—number of single argument level units.

2.5. Baselines

We compared our model with sequence to sequence model described in [Sutskever et al, 2014], (LSTM encoder-decoder model without any modifications), and to pattern based recognizer, created by replacing words in natural language statements with variables (e.g. Add $qty $object to $storage_name). Patterns were then tuned by humans, by manually adding some possible variations.

3. Results

3.1. Small dataset

Out of 14 test queries, that were paraphrases of original queries, our model produced correct parse for 6 examples, vanilla seq2seq model produced no correct parses, and pattern matching resulted in 4 correct examples. BLEU score of our model was 35, while vanilla LSTM reached value of 25, mainly from learning syntactic patterns of queries.

3.2. Extended dataset

Result on test subset of extended dataset (126 queries) are summarazed in Table 2. We can see that model with all suggested extensions reached best accuracy from all approaches studied.

Table 2. BLEU and parsing accuracy on extended dataset

Model	Number of LSTM cells	Accuracy (percentage queries that can be run and produce correct result),%	BLEU
Pattern matching	—	32%	—
seq2seq	200	0	34
seq2seq	500	0	30
Copying mechanism	200	20%	28
Copying mechanism + intermediate layer supervison	500	25%	35
Copying mechanism + intermediate layer supervison + timescales (100/200/200)	500	55%	62

Example generated parses are shown in Table 3.

Table 3. Examples of results

User query	Generated parse	Reference parse
есть ли на складе подушки? (are there any pillows in the store?)	MatchOne({"type":"склад"}). child({"имя":"подушки"}). NotEmpty( ) *MatchOne* ({"*type*":"*store*"}). *child* ({"*name*":"*pillows*"}). *NotEmpty*( )	MatchOne({"type":"склад"}). child({"имя":"подушки"}). NotEmpty()
забрать со склада все подушки (All pillows are taken from the store)	MatchOne({"type":"склад"}). child({"имя":"подушки"}). Set("количество","unk") *MatchOne*({"*type*":"*store*"}). *child*({"*name*":"*pillows*"}). *Set*("*quantity*", " *unk*")	MatchOne ({"type":"склад"}). child({"имя":"подушки"}). Set("количество","0")
задачу проверить склад перенести в раздел вопросы склада (Move task of checking the store in the section “store-related questions”	MatchOne ({"type":"задача", "имя":"проверить склад"}). Set ("раздел", "вопросы склад") *MatchOne* ({"*type*":"*task*", "*name*":"*checking store*"}). *Set* ("*section*","*store related questions*")	MatchOne({"type":"задача", "имя":"проверить склад"}). Add("раздел","вопросы склада")

We also conducted experiment where 5 different users were given interface to rule based and neural parsers and measured user satisfaction. Each user was asked to rate each system on the scale 1 to 10. Neural system was rated 5.6 and rule based 4.8. Given small sample size, these results can not be interpreted in favor of a neural system, but we believe that they indicate that neural parser is viable alternative to rules based systems even with relatively small datasets.

4. Conclusions

We presented a set of initial experiments with neural network-based semantic parser for Russian language, focused on direct translation of natural language queries into graph database queries in the situation where only limited amount of training data is available. We found, that naive application of sequence to sequence models to the task does not work, and suggested a number of extensions, including using of Russian word vectors to represent database identifiers, simplifying query syntax, augmenting model with copying capability, and use of specialized error functions. With these extensions, the task become more feasible and results are competitive with other strategies, that are available to developers, facing similar kind of problem, like rule-based pattern matching.

While obtained accuracy levels are still low in absolute values, it is important to note that task itself is very complex, and we find results encouraging, because there is a lot of room for improvement.

References

1. *Berant, J., Chou, A., Frostig, R., & Liang, P.* (2013, October). Semantic Parsing on Freebase from Question-Answer Pairs. In EMNLP Vol. 2, No. 5, p. 6
2. *Elman, J. L.* (1990). Finding structure in time. Cognitive science, 14(2), 179–211.
3. *Hochreiter, S., & Schmidhuber, J.* (1997). Long short-term memory. Neural computation, 9(8), 1735–1780.
4. *Jia, R., & Liang, P.* (2016). Data recombination for neural semantic parsing. arXiv preprint arXiv:1606.03622.
5. *Koutnik, J., Greff, K., Gomez, F., & Schmidhuber, J.* (2014). A clockwork rnn. arXiv preprint arXiv:1402.3511.
6. *Liang, C., Berant, J., Le, Q., Forbus, K. D., & Lao, N.* (2016). Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. arXiv preprint arXiv:1611.00020.
7. *Liang, P., Jordan, M. I., & Klein, D.* (2011). Learning dependency-based compositional semantics. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1 pp. 590–599. Association for Computational Linguistics.
8. *Locascio, Nicholas, et al.* "Neural Generation of Regular Expressions from Natural Language with Minimal Domain Knowledge." arXiv preprint arXiv:1608.03000(2016).
9. *Moschitti, A.* (2004, July). A study on convolution kernels for shallow semantic parsing. In Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (p. 335). Association for Computational Linguistics.
10. *Papineni, Kishore, et al.* "BLEU: a method for automatic evaluation of machine translation." Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, 2002.
11. *Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le.* "Sequence to sequence learning with neural networks." Advances in neural information processing systems. 2014.
12. *Tang, L. R., & Mooney, R. J.* (2001). Using multiple clause constructors in inductive logic programming for semantic parsing. In European Conference on Machine Learning, pp. 466–477. Springer Berlin Heidelberg.
13. *Vinyals, O., Fortunato, M., & Jaitly, N.* (2015). Pointer networks. In Advances in Neural Information Processing Systems (pp. 2692–2700).
14. *Wang, Y., Berant, J., & Liang, P.* (2015, July). Building a Semantic Parser Overnight. In ACL (1) (pp. 1332–1342).
15. *Xiao, C., Dymetman, M., & Gardent, C.* (2016). Sequence-based structured prediction for semantic parsing. Proceedings Association For Computational Linguistics, Berlin.
16. *Giovanni Campagna, Rakesh Ramesh, Silei Xu, Michael Fischer, Monica S. Lam.* (2016). Sabrina: The Architecture of an Open, Crowdsourced, Privacy-Preserving, Programmable Virtual Assistant.