Semantic Parsing for Single-Relation Question Answering

Wen-tau Yih Xiaodong He Christopher Meek
Microsoft Research
Redmond, WA 98052, USA
{scottyih,xiaohe,meek}@microsoft.com

Abstract

We develop a semantic parsing framework based on semantic similarity for open domain question answering (QA). We focus on single-relation questions and decompose each question into an entity mention and a relation pattern. Using convolutional neural network models, we measure the similarity of entity mentions with entities in the knowledge base (KB) and the similarity of relation patterns and relations in the KB. We score relational triples in the KB using these measures and select the top scoring relational triple to answer the question. When evaluated on an open-domain QA task, our method achieves higher precision across different recall points compared to the previous approach, and can improve F_1 by 7 points.

1 Introduction

Open-domain question answering (QA) is an important and yet challenging problem that remains largely unsolved. In this paper, we focus on answering single-relation factual questions, which are the most common type of question observed in various community QA sites (Fader et al., 2013), as well as in search query logs. We assumed such questions are answerable by issuing a single-relation query that consists of the relation and an argument entity, against a knowledge base (KB). Example questions of this type include: "*Who is the CEO of Tesla?*" and "*Who founded Paypal?*"

While single-relation questions are easier to handle than questions with more complex and multiple relations, such as "*When was the child of the former Secretary of State in Obama's administration born?*", single-relation questions are still far from completely solved. Even in this restricted domain there are a large number of paraphrases of the same question. That is to say that the problem of mapping from a question to a particular relation and entity in the KB is non-trivial.

In this paper, we propose a semantic parsing framework tailored to single-relation questions. At the core of our approach is a novel semantic similarity model using convolutional neural networks. Leveraging the question paraphrase data mined from the WikiAnswers corpus by Fader et al. (2013), we train two semantic similarity models: one links a mention from the question to an entity in the KB and the other maps a relation pattern to a relation. The answer to the question can thus be derived by finding the relation–entity triple $r(e_1, e_2)$ in the KB and returning the entity not mentioned in the question. By using a general semantic similarity model to match patterns and relations, as well as mentions and entities, our system outperforms the existing rule learning system, PARALEX (Fader et al., 2013), with higher precision at all the recall points when answering the questions in the same test set. The highest achievable F_1 score of our system is 0.61, versus 0.54 of PARALEX.

The rest of the paper is structured as follows. We first survey related work in Sec. 2, followed by the problem definition and the high-level description of our approach in Sec. 3. Sec. 4 details our semantic models and Sec. 5 shows the experimental results. Finally, Sec. 6 concludes the paper.

2 Related Work

Semantic parsing of questions, which maps natural language questions to database queries, is a critical component for KB-supported QA. An early example of this research is the semantic parser for answering geography-related questions, learned using inductive logic programming (Zelle and Mooney, 1996). Research in this line originally used small, domain-specific databases, such as GeoQuery (Tang and Mooney, 2001; Liang et al., 2013). Very recently, researchers have started developing semantic parsers for large, general-domain knowledge bases like Freebase and DBpedia (Cai and Yates, 2013; Berant et al., 2013; Kwiatkowski et al., 2013). Despite significant progress, the problem remains challenging. Most methods have not yet been scaled to large KBs that can support general open-domain QA. In contrast, Fader et al. (2013) proposed the PARALEX system, which targets answering single-relation questions using an automatically created knowledge base, ReVerb (Fader et al., 2011). By applying simple seed templates to the KB and by leveraging community-authored paraphrases of questions from WikiAnswers, they successfully demonstrated a high-quality lexicon of pattern-matching rules can be learned for this restricted form of semantic parsing.

The other line of work related to our approach is continuous representations for semantic similarity, which has a long history and is still an active research topic. In information retrieval, TF-IDF vectors (Salton and McGill, 1983), latent semantic analysis (Deerwester et al., 1990) and topic models (Blei et al., 2003) take the bag-of-words approach, which captures well the contextual information for documents, but is often too coarse-grained to be effective for sentences. In a separate line of research, deep learning based techniques have been proposed for semantic understanding (Mesnil et al., 2013; Huang et al., 2013; Shen et al., 2014b; Salakhutdinov and Hinton, 2009; Tur et al., 2012). We adapt the work of (Huang et al., 2013; Shen et al., 2014b) for measuring the semantic distance between a question and relational triples in the KB as the core component of our semantic parsing approach.

3 Problem Definition & Approach

In this paper, we focus on using a knowledge base to answer *single-relation* questions. A single-relation question is defined as a question composed of an entity mention and a binary relation description, where the answer to this question would be an entity that has the relation with the given entity. An example of a single-relation question is "*When were DVD players invented?*" The entity is `dvd-player` and the relation is `be-invent-in`. The answer can thus be described as the following lambda expression:

$$\lambda x.\,\texttt{be-invent-in}(\texttt{dvd-player}, x)$$

$$Q \rightarrow RP \wedge M \quad (1)$$

$$RP \rightarrow \textit{when were X invented} \quad (2)$$

$$M \rightarrow \textit{dvd players} \quad (3)$$

$$\textit{when were X invented} \rightarrow \texttt{be-invent-in} \quad (4)$$

$$\textit{dvd players} \rightarrow \texttt{dvd-player} \quad (5)$$

Figure 1: A potential semantic parse of the question "When were DVD players invented?"

A knowledge base in this work can be simply viewed as a collection of binary relation instances in the form of $r(e_1, e_2)$, where r is the relation and e_1 and e_2 are the first and second entity arguments.

Single-relation questions are perhaps the easiest form of questions that can directly be answered by a knowledge base. If the mapping of the relation and entity in the question can be correctly resolved, then the answer can be derived by a simple table lookup, assuming that the fact exists in the KB. However, due to the large number of paraphrases of the same question, identifying the mapping accurately remains a difficult problem.

Our approach in this work can be viewed as a simple semantic parser tailored to single-relation questions, powered by advanced semantic similarity models to handle the paraphrase issue. Given a question, we first separate it into two disjoint parts: the *entity mention* and the *relation pattern*. The entity mention is a subsequence of consecutive words in the question, where the relation pattern is the question where the mention is substituted by a special symbol. The mapping between the pattern and the relation in the KB, as well as the mapping between the mention and the entity are determined by corresponding semantic similarity models. The high-level approach can be viewed as a very simple context-free grammar, which is shown in Figure 1.

The probability of the rule in (1) is 1 since we assume the input is a single-relation question. For the exact decomposition of the question (e.g., (2), (3)), we simply enumerate all combinations and assign equal probabilities to them. The performance of this approach depends mainly on whether the relation pattern and entity mention can be resolved correctly (e.g., (4), (5)). To deter-

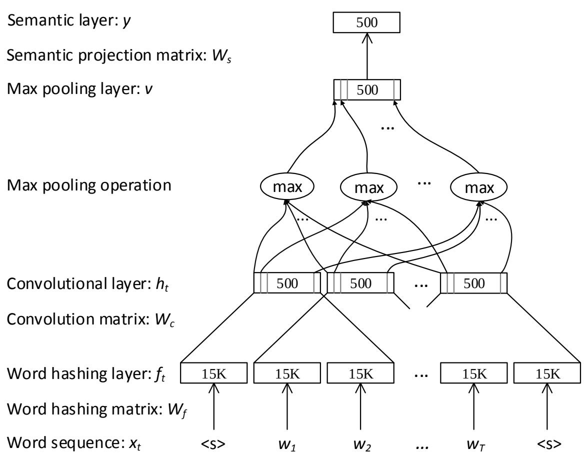

Semantic layer: y
Semantic projection matrix: W_s
Max pooling layer: v
Max pooling operation
Convolutional layer: h_t
Convolution matrix: W_c
Word hashing layer: f_t
Word hashing matrix: W_f
Word sequence: x_t
500
max
15K
<s>
w_1
w_2
w_T

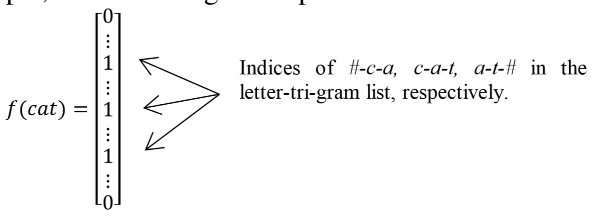
Figure 2: The CNNSM maps a variable-length word sequence to a low-dimensional vector in a latent semantic space. A word contextual window size (i.e., the receptive field) of three is used in the illustration. Convolution over word sequence via learned matrix W_c is performed implicitly via the earlier word hashing layer's mapping with a local receptive field. The max operation across the sequence is applied for each of 500 feature dimensions separately.

mine the probabilities of such mappings, we propose using a semantic similarity model based on convolutional neural networks, which is the technical focus in this paper.

4 Convolutional Neural Network based Semantic Model

Following (Collobert et al., 2011; Shen et al., 2014b), we develop a new convolutional neural network (CNN) based semantic model (CNNSM) for semantic parsing. The CNNSM first uses a convolutional layer to project each word within a context window to a local contextual feature vector, so that semantically similar word-n-grams are projected to vectors that are close to each other in the contextual feature space. Further, since the overall meaning of a sentence is often determined by a few key words in the sentence, CNNSM uses a max pooling layer to extract the most salient local features to form a fixed-length global feature vector. The global feature vector can be then fed to feed-forward neural network layers to extract non-linear semantic features. The architecture of the CNNSM is illustrated in Figure 2. In what follows, we describe each layer of the CNNSM in detail, using the annotation illustrated in Figure 2.

In our model, we leverage the word hashing technique proposed in (Huang et al., 2013) where we first represent a word by a letter-trigram count vector. For example, given a word (e.g., cat), after adding word boundary symbols (e.g., #cat#), the word is segmented into a sequence of letter-n-grams (e.g., letter-trigrams: #-c-a, c-a-t, a-t-#). Then, the word is represented as a count vector of letter-trigrams. For example, the letter-trigram representation of "cat" is:


$f(cat) = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$
Indices of #-c-a, c-a-t, a-t-# in the letter-tri-gram list, respectively.

In Figure 2, the word hashing matrix W_f denotes the transformation from a word to its letter-trigram count vector, which requires no learning. Word hashing not only makes the learning more scalable by controlling the size of the vocabulary, but also can effectively handle the OOV issues, sometimes due to spelling mistakes. Given the letter-trigram based word representation, we represent a word-n-gram by concatenating the letter-trigram vectors of each word, e.g., for the t-th word-n-gram at the word-n-gram layer, we have:

$$l_t = \left[f_{t-d}^T, \cdots, f_t^T, \cdots, f_{t+d}^T\right]^{\mathbf{T}}, t = 1, \cdots, T$$

where f_t is the letter-trigram representation of the t-th word, and $n = 2d + 1$ is the size of the contextual window. The convolution operation can be viewed as sliding window based feature extraction. It captures the word-n-gram contextual features. Consider the t-th word-n-gram, the convolution matrix projects its letter-trigram representation vector l_t to a contextual feature vector h_t. As shown in Figure 2, h_t is computed by

$$h_t = \tanh(W_c \cdot l_t), t = 1, \cdots, T$$

where W_c is the feature transformation matrix, as known as the convolution matrix, which are shared among all word n-grams. The output of the convolutional layer is a sequence of local contextual feature vectors, one for each word (within a contextual window). Since many words do not have significant influence on the semantics of the sentence, we want to retain in the global feature vector only the salient features from a few key words. For this purpose, we use a max operation, also known as max pooling, to force the network to retain only the most useful local features produced by the convolutional layers. Referring to the max-pooling layer of Figure 2, we have

$$v(i) = \max_{t=1,\cdots,T}\{f_t(i)\}, i = 1, \cdots, K$$

where $v(i)$ is the i-th element of the max pooling layer v, $h_t(i)$ is the i-th element of the t-th local feature vector h_t. K is the dimensionality of the max pooling layer, which is the same as the dimensionality of the local contextual feature vectors $\{h_t\}$. One more non-linear transformation layer is further applied on top of the global feature vector v to extract the high-level semantic representation, denoted by y. As shown in Figure 2, we have $y = tanh(W_s \cdot v)$, where v is the global feature vector after max pooling, W_s is the semantic projection matrix, and y is the vector representation of the input query (or document) in latent semantic space. Given a pattern and a relation, we compute their relevance score by measuring the cosine similarity between their semantic vectors. The semantic relevance score between a pattern Q and a relation R is defined as the cosine score of their semantic vectors y_Q and y_R.

We train two CNN semantic models from sets of pattern–relation and mention–entity pairs, respectively. Following (Huang et al., 2013), for every pattern, the corresponding relation is treated as a positive example and 100 randomly selected other relations are used as negative examples. The setting for the mention–entity model is similar.

The posterior probability of the positive relation given the pattern is computed based on the cosine scores using softmax:

$$P(R^+|Q) = \frac{\exp(\gamma \cdot \cos(y_{R^+}, y_Q))}{\sum_{R'} \exp(\gamma \cdot \cos(y_{R'}, y_Q))}$$

where γ is a scaling factor set to 5. Model training is done by maximizing the log-posteriori using stochastic gradient descent. More detail can be found in (Shen et al., 2014a).

5 Experiments

In order to provide a fair comparison to previous work, we experimented with our approach using the PARALAX dataset (Fader et al., 2013), which consists of paraphrases of questions mined from WikiAnswers and answer triples from ReVerb. In this section, we briefly introduce the dataset, describe the system training and evaluation processes and, finally, present our experimental results.

5.1 Data & Model Training

The PARALEX training data consists of approximately 1.8 million pairs of questions and single-relation database queries, such as "When were DVD players invented?", paired with `be-invent-in(dvd-player,?)`. For evaluation, the authors further sampled 698 questions that belong to 37 clusters and hand labeled the answer triples returned by their systems.

To train our two CNN semantic models, we derived two parallel corpora based on the PARALEX training data. For relation patterns, we first scanned the original training corpus to see if there was an exact surface form match of the entity (e.g., `dvd-player` would map to "DVD player" in the question). If an exact match was found, then the pattern would be derived by replacing the mention in the question with the special symbol. The corresponding relation of this pattern was thus the relation used in the original database query, along with the variable argument position (i.e., 1 or 2, indicating whether the answer entity was the first or second argument of the relation). In the end, we derived about 1.2 million pairs of patterns and relations. We then applied these patterns to all the 1.8 million training questions, which helped discover 160 thousand new mentions that did not have the exact surface form matches to the entities.

When training the CNNSM for the pattern–relation similarity measure, we randomly split the 1.2 million pairs of patterns and relations into two sets: the training set of 1.19 million pairs, and the validation set of 12 thousand pairs for hyper-parameter tuning. Data were tokenized by replacing hyphens with blank spaces. In the experiment, we used a context window (i.e., the receptive field) of three words in the convolutional neural networks. There were 15 thousand unique letter-trigrams observed in the training set (used for word hashing). Five hundred neurons were used in the convolutional layer, the max-pooling layer and the final semantic layer, respectively. We used a learning rate of 0.002 and the training converged after 150 iterations. A similar setting was used for the CNNSM for the mention–entity model, which was trained on 160 thousand mention-entity pairs.

5.2 Results

We used the same test questions in the PARALEX dataset to evaluate whether our system could find

	F_1	Precision	Recall	MAP
CNNSM$_{pm}$	**0.57**	0.58	**0.57**	**0.28**
CNNSM$_p$	0.54	0.61	0.49	0.20
PARALEX	0.54	**0.77**	0.42	0.22

Table 1: Performance of two variations of our systems, compared with the PARALEX system.

the answers from the ReVerb database. Because our systems might find triples that were not returned by the PARALEX systems, we labeled these new question–triple pairs ourselves.

Given a question, the system first enumerated all possible decompositions of the mentions and patterns, as described earlier. We then computed the similarity scores between the pattern and all relations in the KB and retained 150 top-scoring relation candidates. For each selected relation, the system then checked all triples in the KB that had this relation and computed the similarity score between the mention and corresponding argument entity. The product of the probabilities of these two models, which are derived from the cosine similarity scores using softmax as described in Sec. 4, was used as the final score of the triple for ranking the answers. The top answer triple was used to compute the precision and recall of the system when reporting the system performance. By limiting the systems to output only answer triples with scores higher than a predefined threshold, we could control the trade-off between recall and precision and thus plot the precision–recall curve.

Table 1 shows the performance in F_1, precision, recall and mean average precision of our systems and PARALEX. We provide two variations here. CNNSM$_{pm}$ is the full system and consists of two semantic similarity models for pattern–relation and mention–entity. The other model, CNNSM$_p$, only measures the similarity between the patterns and relations, and maps a mention to an entity when they have the same surface form.

Since the trade-off between precision and recall can be adjusted by varying the threshold, it is more informative to compare systems on the precision–recall curves, which are shown in Figure 3. As we can observe from the figure, the precision of our CNNSM$_{pm}$ system is consistently higher than PARALEX across all recall regions. The CNNSM$_m$ system also performs similarly to CNNSM$_{pm}$ in the high precision regime, but is inferior when recall is higher. This is understandable

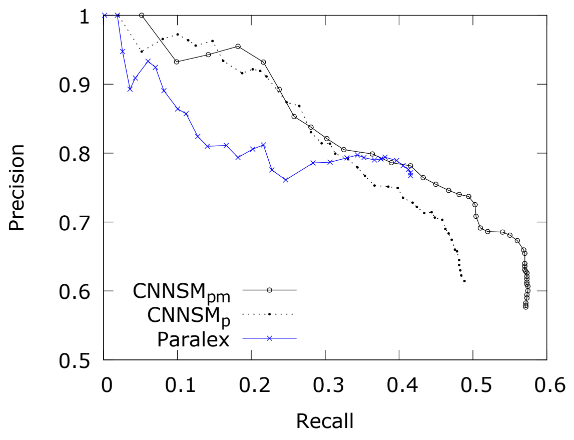


Figure 3: The precision–recall curves of the two variations of our systems and PARALEX.

since the system does not match mentions with entities of different surface forms (e.g., “Robert Hooke” to “Hooke”). Notice that the highest F_1 values of them are 0.61 and 0.56, compared to 0.54 of PARALEX. Tuning the thresholds using a validation set would be needed if there is a metric (e.g., F_1) that specifically needs to be optimized.

6 Conclusions

In this work, we propose a semantic parsing framework for single-relation questions. Compared to the existing work, our key insight is to match relation patterns and entity mentions using a semantic similarity function rather than lexical rules. Our similarity model is trained using convolutional neural networks with letter-trigrams vectors. This design helps the model go beyond bag-of-words representations and handles the OOV issue. Our method achieves higher precision on the QA task than the previous work, PARALEX, consistently at different recall points.

Despite the strong empirical performance, our system has room for improvement. For instance, due to the variety of entity mentions in the real world, the parallel corpus derived from the WikiAnswers data and ReVerb KB may not contain enough data to train a robust entity linking model. Replacing this component with a dedicated entity linking system could improve the performance and also reduce the number of pattern/mention candidates when processing each question. In the future, we would like to extend our method to other more structured KBs, such as Freebase, and to explore approaches to extend our system to handle multi-relation questions.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October. Association for Computational Linguistics.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 423–433, Sofia, Bulgaria, August. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*.

Scott Deerwester, Susan Dumais, Thomas Landauer, George Furnas, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6).

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference of Empirical Methods in Natural Language Processing (EMNLP '11)*, Edinburgh, Scotland, UK, July 27-31.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1608–1618, Sofia, Bulgaria, August. Association for Computational Linguistics.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556, Seattle, Washington, USA, October. Association for Computational Linguistics.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech*.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978.

Gerard Salton and Michael J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw Hill.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014a. A convolutional latent semantic model for web search. Technical Report MSR-TR-2014-55, Microsoft Research.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014b. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, pages 373–374.

Lappoon Tang and Raymond Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Machine Learning: ECML 2001*, pages 466–477. Springer.

Gokhan Tur, Li Deng, Dilek Hakkani-Tur, and Xiaodong He. 2012. Towards deeper understanding: deep convex networks for semantic utterance classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 5045–5048. IEEE.

John Zelle and Raymond Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1050–1055.