Natural Language Supported Relation Matching for Question Answering with Knowledge Graphs

Hongyu Li
hongyul@andrew.cmu.edu

Chenyan Xiong
cx@cs.cmu.edu

Jamie Callan
callan@cs.cmu.edu

Carnegie Mellon University

Abstract

This work focuses on the relation matching problem in knowledge based question answering systems. Finding the right relation a natural question asks is a key step in current knowledge based question answering systems, while also being the most difficult one, because of the mismatch between natural language question and formal relation type definitions. In this paper, we present two approaches to tackle this problem. The first approach tries to directly learn the soft match between the question and the relations from the training data using neural networks. The second approach enriches the relation name with natural language support sentences generated from Wikipedia, which provide additional matches with the question. Experiments on the WebQuestions dataset demonstrate that both of our approaches improve the relation matching accuracy of a prior state-of-the-art. Our further analysis reveals the high quality of support sentences and suggests the rich potential of support sentences in question answering and semantic parsing tasks.

1 Introduction

If we were asked the question "Who inspired Obama?", we might first search for the entity "Barack Obama" in a knowledge base such as Freebase [4], follow the relation "influenced by" and then get the answer "Nipsey Russell". Knowledge-based question answering systems mimic this process and aim to answer natural language questions by the rich structured semantics stored in large-scale knowledge bases [3].

Prior research structured the knowledge based question answering as a *semantic parsing* problem: to parse the natural language question into a sub-graph of entities, relations and target nodes in the knowledge base, which leads to the answer (for example, via SPARQL queries) [2, 8]. The key challenge during this process is the *relation matching* step: the system needs to figure out which aspect the question is asking and matches the correct relation type. Due to the different characteristics of natural language questions and the formal relation type definitions, a relation's name (e.g. "influenced by") may not appear in the question (e.g. "Who inspired Obama?"). A simple exact match system would fail due to this vocabulary mismatch; soft match and paraphrasing techniques are required to address the relation matching task.

This work presents two approaches for the relation matching task in knowledge-based QA systems. The first approach tries to directly learn the match between natural language questions to relation names using deep learning techniques. We build two Bidirectional LSTM neural networks to align the question and the relation using end-to-end learned word embeddings and LSTM parameters. The second approach enriches the representation of relation using natural language support sentences (NLSS). We extract all Wikipedia sentences that contain the head and tail entities of a relation, and then

In: L. Dietz, C. Xiong, E. Meij (eds.): Proceedings of the First Workshop on Knowledge Graphs and Semantics for Text Retrieval and Analysis (KG4IR), Tokyo, Japan, 11-Aug-2017, published at http://ceur-ws.org

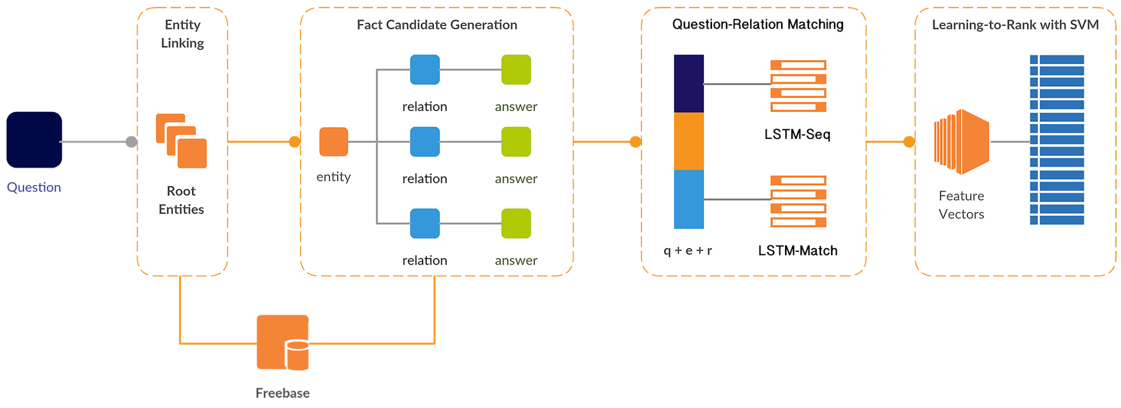

Figure 1: The architecture of our QA system. Our main focus is the question-relation matching stage.

use neural networks to select the most related (best) NLSS for the relation. The best NLSS of the relation is then matched with the question via another LSTM, and provide a richer and more natural description of the relation.

Our experiments on the WebQuestion [3] dataset demonstrated the effectiveness of the two approaches. The LSTM's themselves provide similar performances with previous relation matching techniques [7] and can be combined to achieve better accuracy. The support sentences significantly increase the representation of the relation and provide an absolute 4% improvement in recall. Our analysis further revealed the potential of natural language support sentences. Although the coverage of support sentences on relations is limited by our simple support sentence finding technique, among those that are covered (25%), support sentences influenced the majority of them, and most of the influences are positive.

2 System Architecture

The system architecture is illustrated in Figure 1. The process of answering each question can be described with the following four steps:

- **Entity Linking:** Identify entities from the question [2],
- **Candidate Answer Generation:** Retrieval all facts that start with the linked entities (root entities) [8],
- **Semantic Parsing:** Match the questions to the candidate facts (relation matching).
- **Answer Ranking:** Rank the fact candidates using Learning to rank [5].

According to our error analysis on one of previous state-of-the-art OpenQA system, `Aqqu`, [2] on **WebQuestions** [3], we identified the following error categories with the following percentages:

- **Ground Truth error** (30.0%): The dataset gives the wrong answer or no answers are provided.
- **Relation Matching error** (41.96%): Wrong relation is matched by the top fact candidate.
- **Entity Linking error** (9.82%): Wrong entities or no entities identified.
- **Other** (18.22%): Knowledge base inconsistency, invalid matching patterns, and combinations of several error types.

The relation matching is the major challenge of the system: most of the errors the system made are due to relation matching mistakes. Thus this work mainly focuses on this step and follows the implementation of the Aqqu [2] system in other stages. In the rest of this paper, we will introduce how we approach the relation matching problem with neural networks and support sentences.

3 Semantic Parsing with LSTM

The goal of the semantic parsing stage is to find a matching score between the question q, and a fact candidate c. A fact candidate is the triple of subject s, relation r, and object o . Because the object mostly does not appear in the question, the semantic parser's goal is to match q with the candidate facts' subject and relation (s, r).

We employ two LSTM models to perform this relation matching: `LSTM-Seq` and `LSTM-Match`.

LSTM-Seq takes the question q and fact pair (subject s and relation r) as the input. It first concatenates them to a sequence $q + s + r$, and maps their words to embeddings. It then uses a Bi-LSTM to map the input sequence $q + s + r$ to hidden representations, and employ a dense layer to predict the ranking score:

$$f_{\text{LSTM-Seq}}(q, c) = \texttt{Dense(Bi-LSTM(Embedding(q + s + r)))}. \tag{1}$$

LSTM-Match takes the question q and the relation r as the input. Instead of concatenating them into one word sequence, LSTM-Match maps q and r to embeddings, uses two Bi-LSTM's to map the q and r to their hidden representations, and computes the cosine similarity of the Bi-LSTM's outputs.

$$f_{\text{LSTM-Match}}(q, c) = \texttt{cosine(Bi-LSTM(Embedding(q)), Bi-LSTM(Embedding(r)))} \tag{2}$$

Training: The two LSTM's are trained using standard pairwise learning to rank:

$$\theta^* = \text{argmin}_\theta \max(0, 1 - f(q, c^+) + f(q, c^-)), \tag{3}$$

The parameters θ to learn include the word embeddings, LSTM parameters, and the Dense layer's parameters. The two LSTM's and their embedding layers are trained independently with separate parameters. The training pairs c^+, c^- are formed by their F1 score [8]. Specifically, for each fact candidate, we use the F1 score of the objects it leads to as its labels (there could be more than one object connected by the same subject and relation, for example, the movies an actress cast).

4 Support Relation Inference with Wikipedia Sentences

The short natural language question and the formally defined relation names lead to vocabulary mismatch problems. For instance, the question "how was pluto discovered?" should be matched with the Freebase relation "discovery technique", which is hard to distinguish from the relation "discoverer". This section introduces our approaches that enrich the relation description with natural language support sentences (NLSS) generated from Wikipedia.

4.1 Support Sentence Generation

For each candidate fact triple (s, r, o), we extract all possible support sentences in Wikipedia that contain both the root entity s and the answer entity o. Specifically, We scan through the entire `enwiki` data dump and parse each Wikipedia page into a list of sentences. Wikipedia has manual annotations for each Wikipedia entity, and we can map them to Freebase entities. For all sentences in Wikipedia that contain both s and o, we add them to the NLSS pool for the candidate fact triple (s, r, o). If the answer node o represents an attribute, not an entity, then all sentences contain the root entity are included as its NLSS.

4.2 Relation Matching via Support Sentences

We first select the *best* support sentences for each candidate fact, given there could be many sentences mentioning the subject and object entities. Then we use these *best* support sentences to enrich the semantic parsing, which provides the 'relation expansion' effect.

Support Sentence Selection. There are two ways to select the *best* support sentence: (1) the one with the highest similarity with the question; (2) the one with the highest similarity with the fact candidate. Both similarity scores are determined by the trained `LSTM-Seq` model.

Semantic Parsing with Support Sentences. We use another `LSTM-Seq` model to match the *best* support sentences with the question. It is trained and used the same as described in the last section, only that the fact candidate is replaced by its best support sentences. This generates two additional ranking scores, which are then combined with the other semantic parsers' (for example, `LSTM-Seq` and `LSTM-Match`) using RankSVM [5]. The RankSVM treats these relation matching scores as ranking features, and is trained using standard pairwise loss, the same as described in the last section. When testing, the relation matching methods (LSTM's and support sentences) are first applied, and their scores are combined by the trained RankSVM to generate the final relation matching score.

5 Experimental Methodology

WebQuestions [3], the standard testbed of OpenQA, is used. It contains 5810 questions with crowd-sourced answers in Freebase. The official train-test split is used: 3778 (70%) training questions and 2032 (30%) testing questions.
Evaluation metrics include precision, recall, and F1 score. The Top K (K=$\{1, 2, 5, 10\}$) accuracy that evaluates whether the correct answer appears in the top k ranking results is also included.

Table 1: Overall accuracies of the semantic parsing systems. The first column lists the semantic parsers used in the corresponding system (multiple parsers are combined by RankSVM). Top k is the number and fraction of questions whose correct answer can be found in top k ranked facts.

Method	**Recall (%)**	**Precision (%)**	**F1**	**Top 1**	**Top 2**	**Top 5**	**Top 10**
CDSSM	45.2	31.0	31.3	748 (36.8%)	1047 (51.5%)	1311 (64.5%)	1475 (72.6%)
LSTM-Seq	46.6	33.4	33.6	752 (37.0%)	1060 (52.2%)	1320 (65.0%)	1488 (73.2%)
LSTM-Match	41.7	30.9	30.5	707 (34.8%)	1021 (50.2%)	1285 (63.2%)	1449 (71.3%)
LSTM-Seq + LSTM-Match	51.0	35.7	36.2	806 (39.7%)	1086 (53.4%)	1363 (67.1%)	1518 (74.7%)
LSTM-Seq + LSTM-Match + CDSSM	53.1	37.2	37.7	824 (40.6%)	1094 (53.8%)	1399 (68.8%)	1573 (77.4%)
All Three + Support Sentences	**57.2**	**39.6**	**38.2**	**860** (42.3%)	**1118** (55.0%)	**1412** (69.5%)	**1611** (79.3%)

Table 2: Distribution of the number of support sentences per fact candidate.

Number of Support Sentences	0	1 − 10	11 − 100	101 − 1000	1001 − 10000	10001 − 100000
Number of Fact Candidates	232,368 (74.9%)	42,167 (13.6%)	23,760 (7.66%)	8,395 (2.71%)	3,310 (1.07%)	147 (0.0474%)

Baseline: The main purpose of this work is to study the effectiveness of LSTM parsers and the support sentence's effect in enriching them. So we only compare them with our implementation of the CDSSM relation matching method in the recent state-of-the-art STAGG system [8]. The integration of our semantic parsers into the full QA system is left for future work.
Implementation details: All embeddings' dimensions are 300. All LSTM layers' dimensions are 32. Hinge loss is used. Training was done with mini-batch (size 64) and the RMSProp optimizer.

6 Evaluation

The performances of our relation matching methods are shown in Table 1 shows the performance results. The first column lists the semantic parsers used, and the '+' sign refers to the combination of multiple parsers using RankSVM.

Effectiveness of LSTM: `LSTM-Seq` and `LSTM-Match` alone are able to achieve similar performance with the state-of-the-art relation matching model `CDSSM`. Combining `LSTM-Seq` and `LSTM-Match` together outperforms `CDSSM` by 5% absolute F1 score. Combining the three neural networks together leads to another $2-3\%$ relative improvements in precision and recall, and greatly improves the robustness of the system: 5% more questions are correctly answered in Top 10. This shows that `LSTM-Seq` and `LSTM-Match` cover different aspects of the relation matching and can be further combined with the previous states-of-the-art.

Effectiveness of Support Sentences: Adding support sentences further improves the recall from 53.1% to 57.2%, and precision from 37.2% to 39.6%. Intuitively, the support sentences provide additional expressions of the candidate fact's relation, mitigate the vocabulary mismatch problem, and thus are more effective in improving recall.

Here is an example question that is answered correctly after using the support sentences:

- **Question**: who was vp for lincoln?
- **Matched fact candidate:** [Abraham Lincoln] → government/us_president/vice_president → Andrew Johnson, Hannibal Hamlin
- **Fact candidate matched previously:** [Abraham Lincoln] → person.profession → Lawyer, Politician, Statesman
- **Top support sentence**: Also in 1864, Brandegee was a member of the Connecticut delegation to the [[1864 Republican National Convention—National Republican Convention]] in [[Baltimore]], which re-nominated President [[Abraham Lincoln—Lincoln]], and nominated [[Andrew Johnson]] for the [[Vice President of the United States—Vice Presidency]].

From this example, we can see that the QA system seems to confuse the question "who was vp for lincoln" with "who was lincoln". The matched relation is "person.profession" before using the support sentences. The support sentences enrich the vocabulary of the fact candidate, and increases the matching score with the question term 'vp'.

We further studied the per question win/loss of the support sentences. We found that adding support sentences gives different answers for a total of 234 questions in testing, with 87 wins, 33 losses, and other ties. The high win/loss ratio (87/33) demonstrates the advantage of the support sentences, but the small number of influenced questions (234/2032) limits the overall improvements. The reason is that many fact candidate sentences (75%) have no support sentences at all, as shown in Table 2. Our support sentence finding method is rather simple and may miss too many possible support sentences. How to align more natural language sentences to Freebase triples while maintaining their quality is an important task in the future work.

Error Analysis: We further performed an error analysis of our system. Among the 58% errors our system made, $38\%(771)$ of the questions are answered with wrong relations, 12% are due to entity linking error, and the rest 8% are either because the answer or the entities cannot be found in Freebase. The distribution shows that the core problem of the system is still relation matching. Many such errors are rather subtle and would require more fine-grained language understanding. For example, in the example below the system mistakenly ranked 'place of birth' before 'nationality' while the question wants the country.

- **Question**: where is jason mraz from?
- **Correct fact candidate:** [Jason Mraz] $\rightarrow$ people.person.nationality $\rightarrow$ United States of America (ranked at 3rd)
- **Matched fact candidate:** [Jason Mraz] $\rightarrow$ people.person.place_of_birth $\rightarrow$ Mechanicsville (ranked at 1st)

7 Conclusions and Future Work

This work focuses on the relation matching problem in knowledge based question answering systems. This task is challenging due to the different characteristics of natural language questions and formal relation type schema: Our manual error analysis of a prior state-of-the-art system, Aqqu, suggests that most of the errors the QA system made are because of relation matching mistakes. We developed two different approaches to address this problem. The first one uses two Bi-LSTM's to directly learn the match between the question and the candidate fact triples, while the second one enriches the relation descriptions with support sentences found in Wikipedia, and feed the support sentences into the LSTM to address the vocabulary mismatches.

Our experiments demonstrate that both of our approaches provide similar or better relation match accuracy compared to the CDSSM used in a prior state-of-the-art knowledge based QA system [8]. The LSTM's and the support sentences cover different aspects of the relation matches, and can be combined with prior work for better accuracy. Our analysis further demonstrates the high quality of generated support sentences: among those questions whose fact answers have any support sentences, adding support sentences influences the major of them and most changes are improvements.

Possible future work includes experimenting with better support sentence generating techniques, hoping to improve the coverage while maintaining the high quality. Another direction is to incorporate the presented relation matching methods into state-of-the-art QA systems, for example, Aqqu [2] and STAGG [8]. Better relation matching models is another important future research topic, given it is still the biggest error source in our system.

Perhaps the most interesting finding of our exploration is the promising potential of the joint utilization of the formal semantics in the knowledge graphs and the rich natural language support sentences. Previously such support sentences were mostly used during the extraction of the knowledge base facts, for example, in OpenIE [1] or distant supervision [6]. After the knowledge bases are constructed, only the formal entity-relation triples are kept; the downstream inference and alignment operations were performed with only the formal semantics stored in the knowledge graph. This work demonstrated the effectiveness of using the natural language support sentences in one of the most important applications of knowledge graphs and suggests the possible usage of them in many other knowledge graph related tasks.

References

[1] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction for the web. In *Proceedings of the the 14th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 2670–2676. IJCAI, 2007.

[2] H. Bast and E. Haussmann. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1431–1440, New York, NY, USA, 2015. ACM.

[3] J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544, 2013.

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA, 2008. ACM.

[5] T. Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 217–226, New York, NY, USA, 2006. ACM.

[6] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL 2009*, pages 1003–1011, 2009.

[7] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 101–110, New York, NY, USA, 2014. ACM.

[8] W. Yih, M. Chang, X. He, and J. Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1321–1331, 2015.