End-to-end Representation Learning for Question Answering with Weak Supervision

Daniil Sorokin and Iryna Gurevych

Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
www.ukp.tu-darmstadt.de

Abstract. In this paper we present a knowledge base question answering system for participation in Task 4 of the QALD-7 shared task. Our system is an end-to-end neural architecture for constructing a structural semantic representation of a natural language question. We define semantic representations as graphs that are generated step-wise and can be translated into knowledge base queries to retrieve answers. We use a convolutional neural network (CNN) model to learn vector encodings for the questions and the semantic graphs and use it to select the best matching graph for the input question. We show on two different datasets that our system is able to successfully generalize to new data.

Keywords: Semantic web, Question-answering, Representation learning, Convolutional neural networks, Semantic parsing, Weak supervision

1 Introduction

QALD is a series of international competitions on mapping natural language questions to knowledge base queries [17]. The goal of the competitions is to provide a benchmark for natural language based interfaces to knowledge bases.

In this paper[1], we present a system that was developed for Task 4 of the QALD-7 shared task, "English question answering over Wikidata". The task is formulated as follows: given a natural language question, translate it into a structured query in SPARQL that can be executed against Wikidata to obtain the answer to the question. The provided training data set for Task 4 consists of 100 natural language questions, the answers may be real word entities, numbers or dates. Wikidata [19] is a popular collaboratively constructed knowledge base that contains around 17 million entities and more than 70 million facts of common knowledge.

In our system, we implement a semantic parsing approach to the problem of knowledge base question answering (factoid QA). That is, we produce semantic representations for natural language questions that are then deterministically converted into SPARQL queries and executed against Wikidata.

[1] This is a pre-print version of the paper for self-archival purposes. The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-69146-6_7

Multiple successful question answering systems were presented in the previous QALD competitions [17], as well as in conjunction with other QA datasets [3, 16, 13]. The key challenge in this respect is how to encode the semantics of the question and to use it to find the correct answer. This can be done either by directly encoding the question meaning into a latent vector encoding (end-to-end systems) or by constructing an explicit structural semantic representation (semantic parsing systems). The latent vector representation is normally used to score individual answer candidates contained in the KB [8, 7, 11], whereas the structural semantic representation is converted to a query to be executed against the KB [3, 22].

Semantic parsing systems, such as [2, 3, 9], usually relied on trained models with manually defined features and therefore, suffer from error propagation [15]. End-to-end systems that learn latent vector encodings for questions and answers eliminate this problem [8, 13]. However, latent vector encodings are hard to analyze for errors or to modify with explicit constraints. Questions that require aggregation over several knowledge base entities or temporal constraints are almost impossible to model with the current end-to-end models (see, for example, error analysis in [8]).

In our approach, we combine the best of the latent vector encodings and explicit semantic representation methods. Our main contribution is an end-to-end iterative generation of multi-relational semantic representations that integrates a neural network to learn vector encodings for questions and semantic representations. We use the similarity between the vector encodings to choose the correct semantic representation for a given question.

The end-to-end neural architecture doesn't need handcrafted features or heavy pre-processing that are required in other approaches. It automatically learns a correspondence between structural and lexical features of a semantic representation and a natural language question. Thus, our approach can better generalize to new unseen questions than approaches based on manually defined features and can directly integrate explicit constraints.

We demonstrate the effectiveness of our system on two datasets: QALD-7 Task 4 and WebQuestions [3]. Both dataset contain questions that require complex reasonsing to be answered.

2 Related work

The existing semantic parsing approaches to knowledge base question answering usually consist of a mechanism that generates acceptable semantic representations and of a model that relies on a combination of hand-crafted features to select the correct representation [3, 16, 22]. As opposed to the end-to-end approaches, error propagation is the main downside of the semantic parsing solutions. For example, Reddy et al. [15] estimate that over 35% of errors are being propagated down the pipeline. We try to overcome this in our approach by designing an end-to-end architecture to process the semantic graphs. Dong et al. [8] and Jain [11] achieve the best results on factoid QA with an end-to-end approach and innovative usage

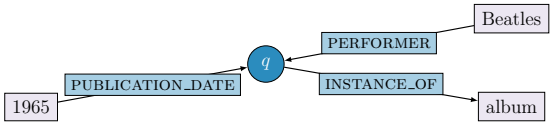

Fig. 1: Graphical semantic representation for a question "What albums did the Beatles release in 1965?"

of neural networks to search through the KB. However, their approaches don't use explicit semantic representations and thus fail on cases when explicit constraints are required.

Encoding the semantics of a questions using semantic graphs is a common way to conceptualize semantic representations [3, 15, 22]. Our graphs are most similar to those of [15] and [22]. We closely follow the approach of [22] who, in contrast to [15], don't rely on syntactic parsing to construct semantic graphs. At the same time, our approach is more flexible than [22] because we don't separate out a single main relation and we are able to process all relations in the same way. This is mainly possible because we are using Wikidata that uniformly encodes all information with binary relations.

A different approach was taken in one of the winning systems of QALD-6 [9]. The authors have used a controlled language to enforce restrictions on syntax and lexical content of a question. This has allowed to unambiguously map the question to a semantic representation and retrieve answers with high precision. The system has demonstarted a very high performance on questions form a closed domain, but wouldn't be able to answer if a question is not covered by the controlled language.

A set of QA system exists that exclusively focus on questions that can be answered using a single triple from the KB [5, 13]. These systems don't incorporate constraints or multi-relational representations and usually model the task as a classification problem. Given a question, one has to predict a relation type from a pre-defined scheme. We don't compare to these approaches, since our focus is on complex questions.

3 Semantic graphs

We use a graphical representation to encode the semantics of a questions (*semantic graph*). Our semantic representations (see Figure 1) consist of a question variable node (q), real world entities (Beatles), constraints (argmin) and relation types from the KB (PERFORMER). The question variable denotes the answer to the question. That is, all entities from the KB that can take its place so that all relations and constraints hold, constitute the answer to the question.

To retrieve the answers given a semantic graph, we convert it to a SPARQL query. All relations in the semantic graph are directed and the conversion is straightforward. We add an ORDER BY clause if there is a temporal constraint in

the graph. The query is executed against a Wikidata RDF dump that is stored locally in Virtuoso[2]. To speed up the query, we blacklist certain relations and entities that are used to encode meta-information in Wikidata.

All relation are attached to the question variable node and we don't allow anything but a Wikidata entity in the position of the question variable. This poses limitations on the types of questions that our system can answers (e.g., aggregate questions or true/false question won't be processed), but it also limits the space of possible graphs and makes the search for the best matching graph more tractable.

Our semantic graphs are coupled to the knowledge base and therefore, only relations and entities defined by the knowledge base scheme are possible. In the following sections, we describe the way we construct semantics graphs for a given question and how we select the graph that matches the semantics of the question the best.

4 System architecture

4.1 Entity linking

Our system takes a natural language question in the form of a string as input. We tokenize it and add part-of-speech tags with the Stanford CoreNLP toolkit [14]. Afterwards, we extract token fragments using a set of regular expression rules that match all sequences of nouns with adjacent modifiers. For each extracted fragment, we generate a set of possible token n-grams and look them up in Wikidata. That gives a list of Wikidata entities that might correspond to the given fragment.

Since Wikidata doesn't offer an entity linking API, we have used alternative labels of the Wikidata entities to perform the look-up. Alternative labels are entered manually for each Wikidata entity and represent different spelling and name variations. For example, the entity *album*:Q482994 has the following alternative labels: [*audio album*, *music album*, *record album*].

Following the approach in [1], we sort the retrieved list of entities by the combination of the Levenshtein distance between the fragment and the item label and the integer part of the item ID:

$$\begin{aligned} rank =& a\,\text{levenshtein}(fragment, entity\_main\_label) \\ &+ b \log entity\_serial\_id \\ &+ c \max(1 - \frac{\text{len}(entity\_label)}{\text{len}(fragment)}, 0) \end{aligned} \tag{1}$$

Since in some cases only a part of a fragment will match an entity, we also add a term to prefer longer fragment matches. The coefficients a, b and c were

[2] `https://virtuoso.openlinksw.com`

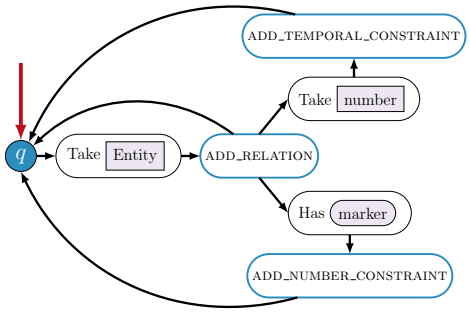

Fig. 2: Scheme of steps that can be undertaken to construct a graph.

heuristically set to 1, 1 and 2 respectively. We select the candidate with the smallest rank for each fragment as the final linking.

For example, in the question "What was the first album released by the Beatles?", we first extract fragments "the first album" and "the Beatles" and then link them to entities *The Beatles (band)*:Q1299 and *album (musical record)*:Q482994.

4.2 Iterative representation generation

Once the list of entities is extracted from the question, we use it to construct possible semantic graphs. We develop a representation generation procedure that defines what kind of graphs can be constructed.

We iteratively generate candidate semantic graphs of the question using a set of actions which can be applied at each step, starting with an empty graph that contains only a question variable. We define three types of actions for graph generation: ADD_RELATION, ADD_TEMPORAL_CONSTRAINT, ADD_NUMBER_CONSTRAINT. The actions define how we search for possible semantic representations. Each action creates a new modified copy of the graph and adds to the list of candidates. Our procedure is inspired by the process of adding constraints to the question in [2], yet our approach is more flexible because we don't divide the representation into the main relation and constraints. Figure 2 shows the application order of the actions.

For each action, we define conditions that must be satisfied in order for the action to be applied at the current step. We list the conditions for each action in Table 1. The conditions control the flow of the graph generation procedure. For example, at the first iteration in Figure 3 we apply the ADD_RELATION action, since it is the only action that can be performed on a empty graph. The result is one graph for each relation that exists for the entity Beatles (Figure 3 shows only three). It is followed by another application of ADD_RELATION since there is a second entity in the question and finally, ADD_TEMP_CONSTRAINT can be applied at the third iteration step because of a temporal marker "first" in the

Action	Conditions	Action description
ADD_RELATION	$\text{LEN}(E) > 0$	Queries Wikidata for relations R that exist for $e, e \in E$, and creates a new representation for each $r, r \in R$
ADD_TEMPORAL_CONSTRAINT	$\text{LEN}(\text{RELATIONS}(s)) > 0 \wedge \text{LEN}(temp\_markers \cap Q) > 0$	Creates a new representation with a constraint that the answer is the last or the first entity in a temporally sorted list
ADD_NUMBER_CONSTRAINT	$\text{LEN}(\text{RELATIONS}(s)) > 0 \wedge \text{CONTAINS}(Q, number)$	Creates a new representation with an added relation that has a numeric argument (e.g. year)

Table 1: The list of actions defined for the iterative representation generation process (E–list of entities, Q–list of question tokens, s–current semantic representation)

question. We check that each candidate semantic graph is valid and those that don't produce answers are not further expanded. For example, in Figure 3 we don't expand the candidate in the middle after the first iteration, since it is impossible to add a relation with album that would result in a valid semantic graph.

4.3 Neural vector encodings

We construct a neural network model to select the best matching semantic graph for the question. It encodes the question and the candidate semantic graphs into fixed-size vectors and then uses the cosine measure to find the correct graph. The semantic graph that has the closest vector to the question vector is taken to be the best semantic representation of the question.

The end-to-end architecture jointly learns vector encodings for questions and semantic graphs. We use the same CNN-based model to encode both the question and the individual relations of the semantic graph. The encodings of the individual relations are later composed into a single vector for the whole graph.

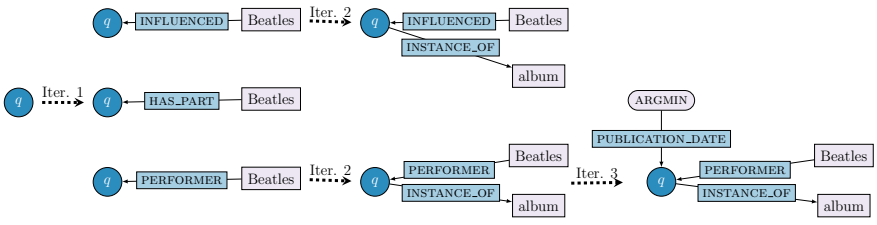

Fig. 3: Generating candidate representations for "What was the first Beatles album?"

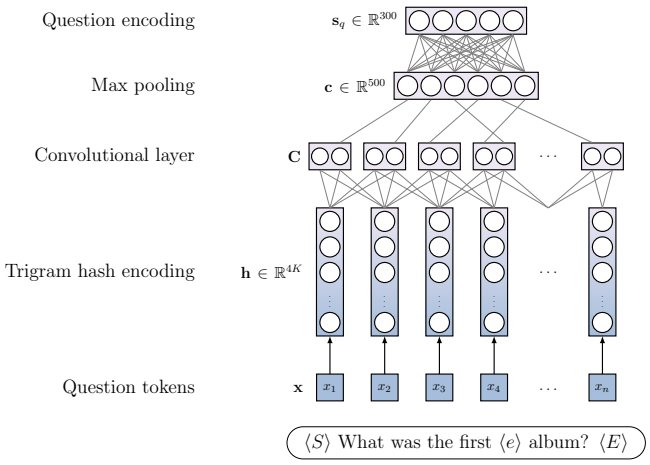

Fig. 4: The architecture of the CNN-based question encoder

We choose CNNs as a basis for our neural network model, since they have proven to be successful for question answering [2, 8].

The architecture of the model is represented in Figure 4, where it is used to encode an example question into a fixed-size vector. The input question is first tokenized and the tokens corresponding to named entities are replaced with a special $\langle e \rangle$ token. We also mark the beginning and the ending of the input with $\langle S \rangle$ and $\langle E \rangle$ respectively. The resulting list of tokens $\mathbf{x} = \{x_1, x_2 \dots x_n\}$ constitutes the input to the model (see at the bottom of Figure 4).

Next, we represent each token as a list of its character trigrams using the hashing technique suggested in [10]. For example, the word "what" has the following trigrams: $\mathbf{t} = \{\#\texttt{wh}, \texttt{wha}, \texttt{hat}, \texttt{at}\#\}$, where # stands for the word boundary. The word is represented as a binary vector $\mathbf{h} \in \mathbb{R}^{|V|}$, where V is the number of possible trigrams in the training data. For the word "what", we mark the positions that correspond to the trigrams in $\mathbf{t}$ with 1 and the rest is 0. Such scheme ensures that different morphological forms of the same word or misspelled words have a similar representation. In the preliminary experiments, we have also observed that this scheme performs more consistent and better than using word or character embeddings.

The list of token representations is further processed by the CNN layer C. For each token, it convolves its representation with the representation of the neighboring tokens. We apply the max pooling operation after the CNN layer to capture the most salient features of the input string. The output of the max pooling operation c is further transformed with a fully connected layer $\mathbf{S}$ and a tanh non-linearity. We take the resulting vector $\mathbf{s}_q$ as the latent encoding of the question.

To encode a semantic graph, we first break it into individual relations. For each relation, we construct a string label by taking the Wikidata relation type

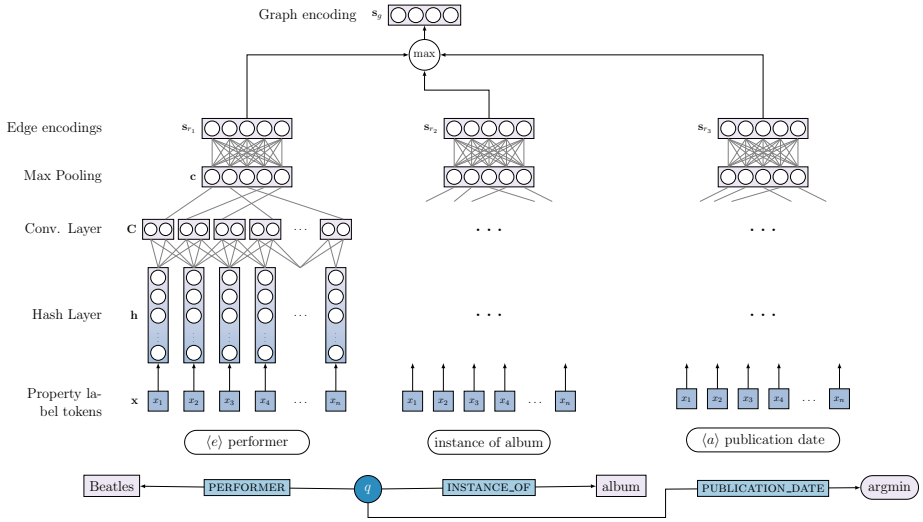

Fig. 5: Graph encoder architecture, here used to encode the example graph in Figure 1.

label and adding the $\langle e \rangle$ token either at the beginning or the end depending on the direction of the relation. For temporal constraints, we always use the label "point in time" and a $\langle a \rangle$ token instead.

We tokenize the relation labels and use them as an input to the same CNN-based model that was used to encode the question (see Figure 5). The output is a semantic vector for each individual relation in the graph: $\{\mathbf{s}_{r_1}, \mathbf{s}_{r_2} \dots \mathbf{s}_{r_m}\}$. The weights of the neural network model are shared in both cases and the vector encodings for questions and semantic graphs are learned jointly. To get a single vector for the whole graph $\mathbf{s}_g$, we apply another max pooling operation on the set of the relation vectors. The order of relations in the graph is not important and the max pooling disregards the order of input elements. The final vector encoding for a candidate graph encodes the most prominent features of the relations that it contains.

5 Question answering as graph generation

In this section, we describe how the system proceeds to answer a given question. First, the input question is encoded into a vector with the question encoder (Figure 4) and the question vector encoding is stored for further reference. Second, we extract entities from the question to start the graph construction. We take the steps described in Section 4.2 to construct possible semantic graphs for the input question. Each constructed variant is encoded by processing individual semantic relations in the graph and combining them into a single graph encoding (Figure 5).

Finally, we score each semantic graph using the cosine distance between the vector encoding of the question and the vector encoding of the graph. During evaluation we perform a beam search and score the constructed graphs at each step, selecting the top 10 graphs for further processing. The semantic graph with the highest score is selected as the final choice for the given question and is used to retrieve the answers from the KB.

6 Model training

To train the model we need positive pairs of questions and semantic graphs. We use weak supervision in the form of question-answer pairs as suggested in [3] to train the neural network model. Weak supervision can provide more training data than available in the form of manually annotated semantic representations. We take the training subset of the WebQuestions dataset [3] which contains 3778 questions and manually retrieved answers. To get pairs of questions and semantic graphs for model training, we run our graph generation procedure on each question. We evaluate each possible semantic graph against Wikidata and compare the extracted answers to the manually provided answers in the dataset. The semantic graphs that result in F1 higher than a certain threshold are stored as positive training instances and the rest of the graphs generated during the same process are used as negative instances. We set the threshold to 0.2 to capture as much of the positive semantic graphs as possible. To increase the search space, we additionally allow second-order relations at this step.

Since WebQuestion was originally developed for the Freebase knowledge base, not all of the questions in the dataset can be answered with Wikidata. With our method, we generate positive semantic graphs for 2334 question from the training part of WebQuestions and reserve 702 of them for validation.

At each training epoch we take all positive semantic graphs and sample up to 20 negative graphs per question. We use the respective F1-scores of the positive semantic graphs to define the training objective. We apply the softmax transformation on the list of F1-scores of the positive graphs and the sampled negative graphs and use the Kullback-Leibler divergence as the loss function. This proves to be effective, since many questions have multiple positive graphs, none of which achieve a perfect F1-score.

The loss is computed for each training instance (a question and a set of semantic graphs) and is averaged over a batch of size 128. The Adam optimizer [12] is used to perform the updates on the weights of the network. We determine the rest of hyper-parameters with the random search on the validation set. We set the filter length of the CNN to 3 and the step size is set to 1. The size of the CNN layer output is 500 and the dimension of the question and graph vector encodings is 300.

	Processed	Right	Partially	Precision	Recall	F1	G. F1
WDAqua (full) [6]	100			0.320	0.323	0.322	0.322
WDAqua (keywords)[6]	100			0.280	0.280	0.280	0.280
Our system	80	25	36	0.351	0.432	0.364	0.291
Our system (ideal model)	80	47	30	0.760	0.898	0.727	0.581

Table 2: Evaluation results on the QALD-7 Task 4 training (100 questions)

7 Experiments

In Table 2, we report preliminary evaluation results on the training dataset for Task 4 of the QALD-7 Shared Task using the metrics from [17]. Our model was not trained on this dataset and, therefore, the reported results represent an expected generalization error of our system.

As mentioned in Section 3, the system currently doesn't cover the questions that require a number or a year as an answer. Therefore, only 80 out of 100 Task 4 dataset questions could be processed by our system.

For comparison, we list results for a competitor system, WDAqua [6], on the same dataset (see the paper for the description of the system). These are the only other results that were published on the QALD-7 Task 4 dataset so far. As opposed to our system, WDAqua can produce numbers and boolean values as answers, but it only allows for a maximum of two relations in a question and doesn't support superlative constructions. Our system proofs to be more flexible and outperforms WDAqua on precision, recall and F1 metrics.

Additionally, we include a version of our system with an oracle neural network model, that always chooses the correct semantic graph. This demonstrates the limitation of our semantic graphs, as the oracle system only achieves an F1 of 0.727. Right now, the semantic graphs don't cover questions that require complex semantic representations and comparison functions. Therefore, questions such as "Show me all basketball players that are higher than 2 meters." could be only partially answered.

To directly compare our system to related work, we also perform an evaluation on the test subset of the WebQuestions dataset. It contains simple questions that can be answered with a single relation as well as complex questions that require multiple relations and constraints. WebQuestion has been a common benchmark for semantic parsers and information retrieval systems for many years.

A system's performance on WebQuestions is measured using *precision*, *recall* and *F1-score*. That ensures a fair evaluation, since a system might provide a partially correct answer that is nevertheless better than a complete miss.

Table 3 summarizes our results on the test part of WebQuestions. We evaluate on a subset of the test set that is substantially covered by Wikidata. We define this subset by searching through the space of possible semantic graphs of arbitrary depth to find questions that can be answered with Wikidata. Practically, this

		Prec.	Rec.	F1
no pre-training	Yao and Van Durme (2015) [21]	0.372	0.596	0.422
	Berant et al. (2013) [3]	0.521	0.591	0.534
	Berant and Liang (2014) [4]	0.550	0.601	0.561
	Yao (2015) [20]	0.565	0.761	0.603
	Our system	0.604	0.638	0.610
	Reddy et al. (2016) [16]	0.663	0.750	0.679
systems with pre-training	Yih et al. (2015) [22]	0.670	0.815	0.698
	Jain (2016) [11]	0.693	0.853	0.725

Table 3: Evaluation results on the WebQuestion dataset

amounts to evaluating if a property path exists between entities in the question and the answers. We retain the questions that have a Wikidata answer with an F1-score higher than 0.8, which results in a subset of 460 questions for evaluation. We compute the results on this subset for other systems that were previously evaluated on the WebQuestions dataset using the systems' output posted by the authors.

As can be seen, our system compares favorably to the rest of the published results outperforming 4 out of 7 systems. 2 out of 3 systems that score better than our approach, [22] and [11], use unsupervised and semi-supervised pre-training on large web corpora such as ClueWeb. Yih et al. [22] additionally employ an entity linking system that is not openly available. They note that their system's performance drops by more than 8% when using alternatives. Reddy et al. [16] don't use unsupervised pre-training, but rely on a deprecated Freebase API for entity linking and make a heavy use of syntactic pre-processing that is not required for our approach. It is important to note that our system currently doesn't use any additional training data or unsupervised pre-training, but the same techniques can be used to improve our approach as well. We leave this directon for future work.

8 Model analysis

Our architecture is able to learn fixed-size vectors for question and semantic graphs. In this section, we briefly analyze the vector encodings for questions that were learned by our model. We take the multi-dimensional encodings and map them into 2-D space using T-SNE [18] to be able to inspect them visually. We use our training dataset for the analysis, since it contains many question of similar semantics.

Figure 6 shows some of the clusters that can be identified among the question encodings. There we can see in detail that formed clusters correspond to questions with similar meaning. The left cluster consists of questions that ask about various

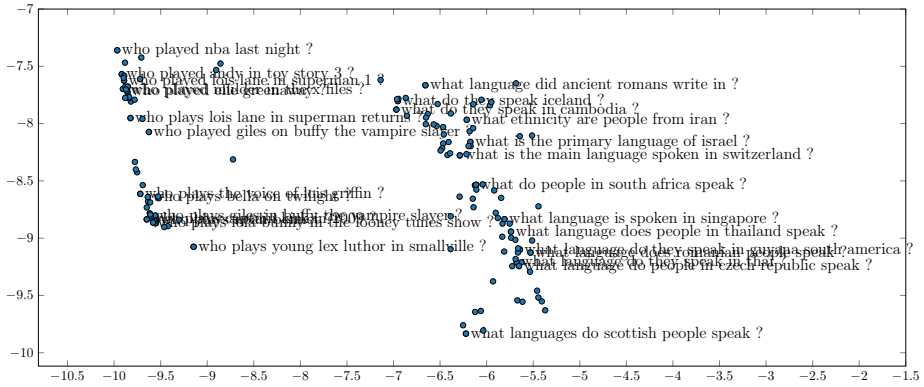

Fig. 6: Two clusters of question encodings learned by the model, every 5^{th} question is labeled. Clusters corresponding to questions about an official language of a country and an actor's role are visible.

character roles in movies. The right cluster groups questions that are concerned with languages spoken in a particular country.

It can been seen that the model learns to group questions with similar meanings but no obvious lexical overlap. For example, questions "What is the primary language of Israel?" and "What do people speak in South Africa speak?" both appear in the center of the right cluster on Figure 6. Some errors are also evident from the diagram: for example, the question "Who played NBA last night?" is incorrectly placed near the character-role-cluster because of the second meaning of the word "play".

9 Conclusions

In this paper, we have presented an end-to-end system that produces semantic representations for natural language questions and evaluates them on Wikidata. We have demonstrated the soundness of our approach by comparison with other systems on two different QA datasets. Our system produces Wikidata items as answers and can successfully process more than 50% of the questions in the QALD-7 Task 4 dataset. On a popural WebQuestion dataset, our system shows the strongest results among the systems that don't rely on semi-supervised or unsupervised pre-training.

There are several obvious directions for future work that we hope to pursue. First, the unsupervised pre-training seems to be a logical way to improve the performance of our system. Second, as observed in [22], entity linking can have a big impact on the overall performance. Our approach to entity liking utilizes high-quality alternative labels, but suffers from coverage issues if relevant labels are not yet in Wikidata. Third, we plan to further develop our semantic graphs to cover other domains of question answering, such as non-factoid QA.

10 Acknowledgments

We thank the anonymous reviewers for their valuable comments and insights that helped us to improve upon the initial version of the paper.

This work has been supported by the German Research Foundation as part of the QA-EduInf project (grant GU 798/18-1 and grant RI 803/12-1). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

References

1. Ahmad Aghaebrahimian and Filip Jurčíček. Open-domain Factoid Question Answering via Knowledge Graph Search. In *Proceedings of 2016 NAACL Human-Computer Question Answering Workshop*, pages 22–28, 2016.
2. Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. Constraint-Based Question Answering with Knowledge Graph. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 2503–2514, 2016.
3. Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1533–1544, 2013.
4. Jonathan Berant and Percy Liang. Semantic Parsing via Paraphrasing. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1415–1425, 2014.
5. Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale Simple Question Answering with Memory Networks. *arXiv preprint*, 2015.
6. Dennis Diefenbach, Kamal Singh, and Pierre Maret. WDAqua-core0: A Question Answering Component for the Research Community. In *Proceedings of the 7th Open Challenge on Question Answering over Linked Data (QALD-7) at ESWC*, 2017.
7. Li Dong and Mirella Lapata. Language to Logical Form with Neural Attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 33–43, 2016.
8. Li Dong, Furu Wei, Ming Zhou, and Ke Xu. Question Answering over Freebase with Multi-Column Convolutional Neural Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 260–269, 2015.
9. Sherzod Hakimov, Christina Unger, Sebastian Walter, and Philipp Cimiano. Applying Semantic Parsing to Question Answering over Linked Data: Addressing the Lexical Gap. In *Natural Language Processing and Information Systems: 20th International Conference on Applications of Natural Language to Information Systems (NLDB)*, pages 103–109, 2015.
10. Po-sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning Deep Structured Semantic Models for Web Search using Clickthrough Data. In *The 22nd ACM international conference on information & knowledge management (CIKM)*, pages 2333–2338, 2013.
11. Sarthak Jain. Question Answering over Knowledge Base using Factual Memory Networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 109–115, 2016.

12. Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint*, 2014.
13. Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. Neural Network-based Question Answering over Knowledge Graphs on Word and Character Level. In *Proceedings of the 26th International Conference on World Wide Web - WWW '17*, pages 1211–1220, 2017.
14. Christopher D. Manning, John Bauer, Jenny Finkel, Steven J. Bethard, Mihai Surdeanu, and David McClosky. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics*, pages 55–60, 2014.
15. Siva Reddy, Mirella Lapata, and Mark Steedman. Large-scale Semantic Parsing without Question-Answer Pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392, 2014.
16. Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics*, 4:127—-140, 2016.
17. Christina Unger, Corina Forascu, Vanessa Lopez, Axel-Cyrille Ngonga Ngomo, Elena Cabrio, Philipp Cimiano, and Sebastian Walter. Question answering over linked data (QALD-5). In *CEUR Workshop Proceedings*, volume 1391, 2015.
18. Laurens Van Der Maaten and Geoffrey Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.
19. Denny Vrandečić and Markus Krötzsch. Wikidata: A Free Collaborative Knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.
20. Xuchen Yao. Lean Question Answering over Freebase from Scratch. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 66–70, 2015.
21. Xuchen Yao and Benjamin Van Durme. Information Extraction over Structured Data: Question Answering with Freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 956–966, 2014.
22. Wen Tau Yih, Ming-wei Chang, Xiaodong He, and Jianfeng 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*, pages 1321–1331, 2015.