Open-domain Factoid Question Answering via Knowledge Graph Search

Ahmad Aghaebrahimian, Filip Jurčíček
Charles University in Prague, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské náměstí 25, 11800 Praha 1, Czech Republic
{Ebrahimian, Jurcicek}@ufal.mff.cuni.cz

Abstract

We introduce a highly scalable approach for open-domain question answering with no dependence on any data set for surface form to logical form mapping or any linguistic analytic tool such as POS tagger or named entity recognizer. We define our approach under the Constrained Conditional Models framework which lets us scale up to a full knowledge graph with no limitation on the size. On a standard benchmark, we obtained near 4 percent improvement over the state-of-the-art in open-domain question answering task.

1 Introduction

We consider the task of simple open-domain question answering. The answer to a simple question can be obtained only by knowing one entity and one property (Bordes et al., 2015).

A property is an attribute which is asked about a specific thing, place or people in a question. The thing, place or people in the question are samples of an entity. The answer to such a question is again an entity or a set of entities. For instance, in the question *"What is the time zone in Dublin?"*, Dublin is an entity and time zone is a property.

Freebase (Bollacker et al., 2008), the knowledge graph which we used in our experiments, contains about 58 million such entities and more than 14 thousands such properties. Hence, entities which we obtain from the knowledge graph are ambiguous in majority of cases. We extract metadata available in the knowledge graph and integrate them into our system using Constrained Conditional Model framework (CCM) (Roth and Yih, 2005) to disambiguate entities.

In WebQuestions (Berant et al., 2013), a data set of 5810 questions which are compiled using the Google suggest API, 86% of the questions are answerable by knowing only one entity (Bordes et al., 2015). It suggests that a large number of the questions which ordinary people ask on the Internet are simple questions and it emphasizes on the importance of simple question answering systems. Besides, the best result on this task is 63.9% (Bordes et al., 2015) which shows open-domain simple QA is still an unresolved task in NLP.

Despite the title, simple QA is not a simple task at all. Flexible and unbound number of entities and their properties in open-domain questions is an intimidating challenge for entity recognition. However, knowledge graphs by providing a structural knowledge base on entities can help a lot.

We use a knowledge graph to recognize entities at test time. Defining a model for entity disambiguation on a single question instead of a whole data set lets us to scale the system up to a large knowledge graph irrespective to its size. We elaborate on entity recognition in Sections 6.2 and 6.3.

The contributions of this paper are a highly scalable QA system and a high performance entity recognition model using knowledge graph search.

2 Related works

Domain specific QA has been studied well (Clarke et al., 2010; Kwiatkowski et al., 2010; Wong and Mooney, 2007; Zettlemoyer and Collins, 2005; Zelle and Mooney, 1996) for many domains. In majority

Proceedings of 2016 NAACL Human-Computer Question Answering Workshop, pages 22–28,
San Diego, California, June 12-17, 2016.

of these studies, a static lexicon is used for mapping surface forms of the entities to their logical forms. Scaling up such lexicons which usually contain from hundreds to several thousands entities is neither easy nor efficient. Instead, knowledge graphs contain millions of entities and are highly efficient structures which can be used for entity recognition. Knowledge graphs provide rich databases of factual information on well-known people, things and places and they proved being beneficial for different tasks in NLP including question answering.

There are many studies on using knowledge graphs for question answering either through information retrieval approach (Yao and Durme, 2014; Bordes et al., 2015) or semantic parsing (Berant et al., 2013; Berant and Liang, 2014; Cai and Yates, 2013; Kwiatkowski et al., 2013). Even in these studies, there is still a list of predefined lexicons for entity recognition (e.g., (Berant et al., 2013; Cai and Yates, 2013)). Essentially, they use knowledge graphs only for validating their generated logical forms and for entity recognition they still depend on some initial lexicons.

Dependence on predefined lexicons limits the scope of language understanding only to those predefined ones. In our approach, we don't use any data set or lexicon for entity recognition. Instead, we obtain valid entities by querying the knowledge graph at test time. Then, we apply some constraints on valid entities to get the correct entity for each question.

In part of CCM, it is first proposed by Roth and Yih in (Roth and Yih, 2005) for reasoning over classifier results. Some other people used it for different problems in NLP (Chang et al., 2012; Punyakanok et al., 2008). (Clarke et al., 2010) proposed a semantic parsing model using Question-Answering paradigm on Geoquery (Zelle, 1995) under CCM framework. Our work differs from them by the size of our data set and knowledge graph and by the open-domain nature of the questions.

The most recent work and state-of-the-art on simple QA belongs to (Bordes et al., 2015) in which they used memory networks for answering the questions in SimpleQuestions data set (Bordes et al., 2015). They proposed their system on a limited version of Freebase containing 2 and 5 million entities (FB2M, FB5M). A 0.5 % decrease in the performance of their system when scaling from FB2M to FB5M suggests that QA in a full knowledge graph is quite a difficult task. CCM lets us to scale QA from limited freebase to the full version with more than 58 million entities.

3 Knowledge graph (Freebase)

Knowledge graphs contain large amounts of factual information about entities (i.e., well-known places, people and things) and their attributes such as place of birth or profession. Large knowledge graphs generally cover numerous domains and they may be a solution for scaling up domain-dependent systems to open-domain ones by expanding their boundary of entity and property recognition. Besides, knowledge graphs are instances of the linked-data technologies. In other words, they can be connected easily to any other knowledge graph and it increases their domain of recognition.

A knowledge graph is a collection of assertions. In an assertion, a source entity is connected to a target entity through a directed and labeled edge. Large knowledge graphs such as Freebase contain billions of such assertions. The elements in an assertion (i.e., source and target entities and the connecting edge) are identified using a unique id which is called machine id or simply MID. These elements are all objects and it means they have some attributes which are called properties. The number of properties is quite huge. However, for the purpose of this paper is enough to know about *"id"*, *"mid"*, *"name"*, *"alias"*, *"type"*, and *"expected type"*.

Each entity in the graph has one unique *"id"* and one unique *"mid"*. In contrast to *"mid"* which has no meaningful association with the entity, *"id"* sometimes is meaningfully similar to its entity's surface form. *"name"* is one surface form for the respective entity and is usually a literal in form of raw text, date or a numerical value. *"alias"* contains the aliases for its entity.

Each entity has a set of *"type"*s. A *"type"* defines an *"IS A"* relation with its entity. For instance, entity *"dublin"* has types *"/topic"*, *"/book_subject"*[1], etc which says *"dublin"* not only is a general topic, but also is the name of a book.

[1]Properties are in abbreviated form to save space. For instance, the full form for this property is *"/book/book_subject"*

"expected type" is defined only for edges. It states, what you should expect to get as a target entity when traversing the graph through this edge or property. Each edge has zero or at most one expected type.

4 Constrained Conditional Model

Constrained Conditional Model (CCM) provides means of fine-tuning the results of a statistical model by enforcing declarative expressive constraints (Roth and Yih, 2005) on them. Constraints are essentially Boolean functions which can be generated using available metadata about entities and properties in Freebase. Freebase assigns a set of *"type"*s to each entity. It also assigns a unique *"expected_type"* to each property. Intuitively, the *"type"* of an answer to a question should be the same as the *"expected_type"* of the property for that question. To each test question, a set of properties each of which with different probabilities and different *"expected_type"*s is assigned. Some of the properties does not have *"expected_type"* and the types assigned to answers are not usually unique. Due to the huge number of entities and their associated types in large knowledge graphs, translating a typical constraint into a feature set for training a statistical model is practically unfeasible. However, it can be done easily using Integer Linear Programming (ILP) (Wen-tau, 2004). In this way, we simply penalize the results of a statistical model which are not in accordance with our constraints. We elaborate more on the constraints in Section 5.

5 The learning model

Let's define P and E as the space of all properties and entities respectively in a test question. For each question like *"What is the time zone in Dublin?"*, we intend to find the tuple $(p\_e) \in P \times E$ for which p's probability and e's score with respect to some features and constraints are maximal. Therefore, we would like to get *"/location/location/time_zones"*[2] as the best property and *"/en/Dublin"* as the best-matching entity in this question.

We decompose the learning model into two steps, namely; property detection and entity recognition. In property detection, we decide which property best describes the purpose of a given question. In this step, we model the assignment of properties given questions using the probability distribution in Equation 1. We use logistic regression technique to train the model and use the model for N-best property assignment to each question at test time.

$$P(p|q) = \frac{\exp(\omega_p^T \phi(q))}{\Sigma_{p_i} \exp(\omega_{p_i}^T \phi(q))} \tag{1}$$

Given a question q, the aim is to find N-best properties which best describe the content of the question and generate the correct answer when querying against the knowledge graph. ϕ in the model is a feature set representing the questions in vector space and ω are the parameters of the model.

In the second step, i.e., entity recognition, we detect and disambiguate the main entity of a question. We use an integer linear classifier for assigning the best-matching entity to each test question at test time (Equation 2). Entity recognition consists of entity detection and entity disambiguation.

$$\begin{aligned} &\text{best entity}(q) = \arg\max_e (\alpha^T s(p_q\_e_q)) \\ &(p_q\_e_p) \in P_q \times E_q \end{aligned} \tag{2}$$

A typical question usually contains tokens that all are available in the knowledge graph while only one of them is the main focus of the question. For instance, in the question *"what is the time zone in Dublin?"*, there are eleven entities which are all available in the knowledge graph (*"time, zone, time zone, ..., Dublin"*) while the focus of the question is on *"Dublin"* and we try to detect it in entity detection.

Detected entities are mostly ambiguous. Given an entity like *"Dublin"*, we need to know what Dublin (i.e., Dublin in Ireland, Dublin in Ohio, etc.) is the focus of the question. To help the system with entity disambiguation, we use heuristics as a constraint to increase the chance of correct entities. [3] Having N-best properties assigned to each question, we initial-

[2] i.e., /Type 1/Type 2/Predicate

[3] The search space in these experiments is not exponentially large hence, instead ILP, simple search methods could be used. However, this is our base line system and we are introducing more complex constraints which can not be solved by searching. To make sure our baseline system is compatible with future improvements, we used an ILP solver form the scratch.

ize $s(p_q\_e_p)$ vector. By optimizing a vector of indicator variables (α) subject to two sets of constraints, we eliminate irrelevant entities.

P_q are N-best properties for a given question q and E_q are valid entities in q. $s(p_q\_e_p)$ is a vector of p_q probabilities. α represents a vector of indicator variables which are optimized through constraint optimization. Constraints for each question are categorized into three categories:

- Constraints in the first category enforce the type of answers of $(p_q\_e_p)$ to be equal to the expected type of p_q in each question. We call the constraints in this category type constraints.
- Constraints in the second category dictate that e_p score which is lexical similarity ratio (edit distance) between the string values of *"name"* and *"id"* properties connected to an entity should be maximal. We call the constraints in this category similarity constraints.
- To make sure that the output of the entity recognition step is zero or at most one entity per question, we define the third constraint to return at most one entity.

A type constraint helps in detecting the main focus of a given question among other valid entities. Despite the assigned properties, each question has *E* number of valid entities. By valid we mean entities which are available in the knowledge graph.

After property detection, N-best properties are assigned to each question each of which has no or at most one *"expected_type"*. The product between the N-best properties and the *E* valid entities gives us $N \times E$ tuples of *(entity_ property)*. We query each tuple and obtain the respective answer from the knowledge graph. Each of the answers has a set of *"type"*s. If the *"expected_type"* of the property of each tuple was available in the set of its answer's *"type"*s, type constraint for the tuple holds true otherwise false.

A similarity constraint helps in entity disambiguation. Each entity has an *"id"* and a *"name"* properties. Ambiguous entities usually have the same *"name"* but different *"id"*s. For instance, entities *"/m/02cft"* and *"/m/013jm1"* both have *"Dublin"* as their *"name"* while the *"id"* for the first one is *"/en/dublin"* and for the second one is *"/en/dublin_ohio"*(and more than 40 other different entities for the same *"name"*).

"name" assigns a surface form for entities and this is the property which we use for extracting valid entities at first place. In this case, similarity constraint for entity *"/m/02cft"* holds true because among all other entities, it has the maximal edit distance ratio between its *"name"* and *"id"* values. It is possible that the content of *"id"* property for an entity is the same as its *"mid"*. In such cases, instead *"id"*, we use *"alias"* property which contains a set of surface forms for entities.

6 Method

At training time, we have training questions accompanied with their knowledge graph assertions each of which includes an entity, a property and an answer. Entities and answers are in their MID formats. We also have access to a knowledge graph (i.e. Freebase) through an MQL query API (Google, 2013). First, all questions are fixate to 20 token. Then we chunk them into their tokens and compute $\phi(q)$ by replacing each token with its vector representation. To train our classifier, we assign a unique index to each property in the training data set and use them as label for the training questions. Given a test question at test time, we first get the N-best properties using our trained model as it is explained below.

6.1 Property detection

In property detection, we train a model which assigns N-best property to each question based on its contents. One approach for doing so is to represent the words in questions as one-hot vectors and train a classifier on them. However, representing words as discreet entities leads the classifier to disregard possible similarity between two tokens. This is known as sparsity problem and vector representation of words partially solve this problem. Word vectors or embeddings capture useful information about words and their relations to other ones. There are studies that shows there are nice directions like gender (king $\rightarrow$ queen) or even major city of countries (Germany $\rightarrow$ Berlin) in learned word vectors (Collobert et al., 2011). This attribute makes embeddings useful features for different NLP tasks spe-

cially those ones which require making decisions on the semantic contents of the texts. We use word embeddings as features in our logistics regression model instead of words themselves. It is like a one layer neural network and it is not much efficient. Still it obtained competitive results with respect to the state-of-the-art.

In another experiment, we added more layers to enhance the performance of the classifier. Finally to enhance the model even further by taking the role of neighboring words into account, we used a Convolutional Neural Network (CNN) with different filter sizes and with a max pool layer on the top. The architecture of our CNN model is similar to Yoon Kim (Kim, 2014) with some minor modifications.

The CNN model contains four consecutive layers in which the first one embeds words into low dimensional vector representation. In this step, we adopted two approaches for learning embeddings. In the first approach, we let the graph to learn word embeddings directly from data by initializing it with a random uniform distribution. In the second approach we used pre-trained Word2vec word embeddings.

The second layer in the CNN slides a convolution window with different sizes over the embeddings and the third layer max pools the result into a vector which is fed into a softmax layer for classification in the last layer. For convolution layer, we used 1,2 and 3 size windows and for the next layer we tried average and max pooling. Finally we tried 1, 2, and 3 fully connected softmax layers at the end. We trained a model in training time and then we use it to obtain N-best properties at test time. Having the properties, the next step in the model is entity recognition which includes entity detection and entity disambiguation.

6.2 Entity detection

Instead relying on external lexicons for mapping surface forms to logical forms, we match surface forms and their MIDs directly using the knowledge graph at test time. For entity detection, we extract spans in questions which are available in the knowledge graph. To do so, we slide a flexible size window over each question and extract all possible spans with all possible sizes. We query a live and full version of Freebase using Meta-Web Query Language (MQL). MQL is a template-base querying language which uses Google API service for querying Freebase on real time.

We query the entity Mid of each span against the knowledge graph. We have two alternatives to obtain initial entity Mids; greedy and full. In greedy approach, only the longest valid entities are reserved and the rest which may be still valid are disregard. In full approach, however, all the entities are reserved. For instance, in a simple span like "time zone", while greedy approach returns only *"time zone"*, full approach returns *"time"*,*"zone"* and *"time zone"*. Spans with at least one entity Mid are recognized as valid entities. Enforcing the type constraints on valid entities distinguishes relevant entities from irrelevant ones.

6.3 Entity disambiguation

Detected entities in the last step in many cases are ambiguous. Entities in large Knowledge graphs each have different meanings and interpretations. In a large knowledge graph, it is possible to find *"dublin"* as the name of a city as well as the name of a book. Moreover, when it is the name of a city, that name is not still unique as we saw in earlier section. We consider similarity constraints as true, if the lexical similarity ratio between *"id"* and *"name"* properties connected to that entity is maximal. It heuristically helps us to obtain an entity which has highest similarity with the surface form in a given question.

7 Experiment

The input for training step in our approach is training and validation sets with their knowledge graph assertions. Using Word2Vec toolkit (Mikolov et al., 2011), we replaced the tokens in the data sets with their vector representations to use them as $\phi(q)$ in our model. We pruned questions with more than 20 tokens in length and fixate shorter ones by adding extra $<.>$ token. We already did some simple pre-processing jobs on the input data including non-alphanumeric character removal and space-separation tokens.

Using these features and the model described above, we trained a classifier using logistic regression, neural network and CNN technique. We used the trained classifier at test time for detecting 100-best properties for each test question. For multilayer

	trained on	property Acc.	over all Acc.	knowledge graph
Bordes et al.	SQ	-	61.6	FB5M
Constraint-based(LR)	SQ	69.80	61.20	Full FB
Constraint-based(NN)	SQ	74.08	63.89	Full FB
Constraint-based(CNN-1)	SQ	78.02	65.16	Full FB
Constraint-based(CNN-2)	SQ	78.82	65.19	Full FB

Table 1: Experimental results on test set of SimpleQuestions (SQ) data set. LR stands for logistic regression model, NN for neural network, CNN-1 for convolutional neural network with one channel and CNN-2 for CNN two channels model.

neural network, we used two hidden layer each with 1024 neurons and for CNN we used the same number of neurons for convolution and softmax layers. We tried to enforce regularization on weight vectors, however as already tested in (Zhang and Wallace, 2015) it had no effect on final results. We also included a new channel in our CNN using POS of tokens which improved the final model but not significantly.

The trained classifier, test questions and Freebase knowledge graph (February 2016) are inputs at test time. For testing our system, we used SimpleQuestions data set (Bordes et al., 2015) which contains 108442 factoid questions. We divided the data set into 70%, 10%, and 20% portions for train, validation and test sets respectively and we did this for three times randomly as a mean of cross validation. However, to make our result comparable to the results of SimpleQuestion authors we reported our results on the official separation test data.

For entity recognition (including detection and disambiguation), we defined two sets of constraints which are described above. Constraints in the first category (i.e., type constraints) helped with entity detection and constraints in the second category (i.e.,lexical similarity) helped with entity disambiguation. We used Gurobi solver for searching through the space of possible entities. The space of search for each question is around 100 to 500 thousands entities. With this space, Gurobi solver was able to detect entity for 20 questions per second.

We used path-level accuracy for evaluating the system. In path-level accuracy, a prediction is considered correct if the predicted entity and the property both are correct. This is the same evaluation metric which is used by the data set authors.

We obtained our best validation accuracy using greedy approach for entity recognition and 128 dimensional embeddings for property detection. Using the same configuration, we reported the accuracy of our system on test data.

8 Result

We used only SimpleQuestions to train our system and then reported the results on official test data separation (Table 1).

We reported the accuracy for property detection and overall system separately. As we mentioned earlier, in future studies we intend to work on more complex and effective constraints to improve the system performance. However, in these series of experiments, the improvement on overall system accuracy is due only to the improvement on property detection. Although our system makes query on the whole knowledge graph, to make sure our results are comparable, we eliminate entities which are not available in FB5M. In this settings, with 99% coverage, we obtained 61.2% accuracy in our logistics regression model which is competitive to the results in (Bordes et al., 2015) when training on the same data set (61.6%). Our neural network system obtained 63.89 which is the same with (Bordes et al., 2015) best results when they trained on three training data sets (WebQuestions and paraphrase data sets in addition to Simple Question). Finally our CNN model obtained 65.19% accuracy only trained on SimpleQuestions. Since we work on full knowledge graph, we hope that our system can answer every possible simple question which its answer is available in the full knowledge graph but not in the FB2M.

9 Conclusion

We introduced a question answering system with no dependence on external lexicons or any other tool. Using our system and on a full knowledge graph, we obtained state-of-the-art results on simple question answering task without any linguistic analytic tool or lexicon. By means of enforcing expressive constraint on statistical models, our approach is able to easily scale up QA systems to a large knowledge graph irrespective to its size.

Acknowledgments

This research was partially funded by the Ministry of Education, Youth and Sports of the Czech Republic under the grant agreement LK11221, core research funding, SVV project number 260 333 and GAUK 207-10/250098 of Charles University in Prague. This work has been using language resources distributed by the LINDAT/CLARIN project of the Ministry of Education, and Sports of the Czech Republic (project LM2010013).

References

J. Berant and P. Liang. 2014. Semantic parsing via paraphrasing. *In Proceedings of Association for Computational Linguistics*.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on freebase from question-answer pairs. *In Proceedings of Empirical Methods in Natural Language Processing*.

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. 2008. a collaboratively created graph database for structuring human knowledge. *In Proceedings of the 2008 ACM SIGMOD international conference on Management of data*.

A. Bordes, N. Usunier, S. Chopra, and J. Weston. 2015. Large-scale simple question answering with memory networks. *In arXiv preprint arXiv:1506.02075*.

Q. Cai and A. Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. *In Proceedings of of Association for Computational Linguistics*.

M. Chang, L. Ratinov, and D. Roth. 2012. Structured learning with constrained conditional models. *Machine Learning*.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. *In Proceedings of the Conference on Computational Natural Language Learning*.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. kuksa. 2011. Natural language processing (almost) from scratch. *Machine Learning Research*.

Google. 2013. Freebase data dumps.

Y. Kim. 2014. Convolutional neural networks for sentence classification. *In Proceedings of EMNLP*.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

T. Kwiatkowski, C. Eunsol, Y. Artzi, and L. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

T. Mikolov, S. Kombrink, A. Deoras, L. Burget, and J. Černocký. 2011. Rnnlm - recurrent neural network language modeling toolkit.

V. Punyakanok, D. Roth, and W. Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*.

D. Roth and W. Yih. 2005. Integer linear programing inference for conditional random fields. *International Conference on Machine Learning*.

Y. Wen-tau. 2004. Global inference using integer linear programming.

Y-W. Wong and R. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. *In Proceedings of Association for Computational Linguistics*.

X. Yao and B. Van Durme. 2014. Information extraction over structured data: Question answering with freebase. *In Proceedings of Association for Computational Linguistics*.

J. M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic proramming. *In Proceedings of the National Conference on Artificial Intelligence*.

J. M. Zelle. 1995. Using inductive logic programming to automate the construction of natural language parsers. *Ph.D. thesis, Department of Computer Sciences, The University of Texas at Austin*.

L. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *In Proceedings of the Annual Conference in Uncertainty in Artificial Intelligence (UAI)*.

Y. Zhang and B. C. Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.