Question Answering over Knowledge Base using Factual Memory Networks

Sarthak Jain
Department of Computer Engineering
Delhi Technological University
DL, India
successar@gmail.com

Abstract

In the task of question answering, Memory Networks have recently shown to be quite effective towards complex reasoning as well as scalability, in spite of limited range of topics covered in training data. In this paper, we introduce Factual Memory Network, which learns to answer questions by extracting and reasoning over relevant facts from a Knowledge Base. Our system generate distributed representation of questions and KB in same word vector space, extract a subset of initial candidate facts, then try to find a path to answer entity using multi-hop reasoning and refinement. Additionally, we also improve the run-time efficiency of our model using various computational heuristics.

1 Introduction

Open-domain question answering (Open QA) is a longstanding problem that has been studied for decades. Early systems took an information retrieval approach, where question answering is reduced to returning passages of text containing an answer as a substring. Recent advances in constructing large-scale knowledge bases (KBs) have enabled new systems that return an exact answer from a KB.

A key challenge in Open QA is to be robust to the high variability found in natural language and the many ways of expressing knowledge in large-scale KBs. Another challenge is to link the natural language of questions with structured semantics of KBs. In this paper, we present a novel architecture based on memory networks (Bordes et al., 2015) that can be trained end-to-end using (question, answer) pairs as training set, instead of strong supervision in the form of (question, associated facts in KB) pairs.

The major contributions of this paper are two-fold: first, we introduce factual memory networks, which are used to answer questions in natural language (e.g *"Where was Bill Gates born?"*) using facts stored in the form of (subject, predicate, object) triplets in a KB (e.g *('Bill Gates', 'place of birth', 'Seattle')*). We evaluate our system against current baselines on various benchmark datasets. Since KBs can be extremely large, making it computationally inefficient to search over all entities and paths, our second goal of this paper is to increase the efficiency of our model in terms of various performance measures and provide better coverage of relevant facts, by intelligently selecting which nodes to expand.

2 Related Work

The state-of-the-art methods for QA over a knowledge base can be classified into three classes: semantic parsing, information retrieval and embedding based.

Semantic parsing (Cai and Yates, 2013; Berant et al., 2013; Kwiatkowski et al., 2013; Berant and Liang, 2014; Fader et al., 2014) based approaches aim to learn semantic parsers which parse natural language questions into logical forms and then query knowledge base to lookup answers. Even though these approaches are difficult to train at scale because of the complexity of their inference, they tend to provide a deep interpretation of the question.

Information retrieval based systems retrieve a set of candidate answers and then conduct further analysis to rank them. Their main difference lies in select-

Proceedings of NAACL-HLT 2016, pages 109–115,
San Diego, California, June 12-17, 2016.

ing correct answers from the candidate set. Yao and Van Durme (2014) used rules to extract question features from dependency parse of questions, and used relations and properties in the retrieved topic graph as knowledge base features.

Embedding based approaches (Bordes et al., 2014b; Bordes et al., 2014a) learn low-dimensional vectors for words and knowledge base constitutes, and use the sum of these vectors to represent questions and candidate answers. However, simple vector addition ignores word order information and higher order n-grams. For example, the question representations of *"who killed A?"* and *"who A killed?"* are same in the vector addition model. (Bordes et al., 2015) used strong supervision signal in form of supporting facts for a question during training to improve their performance.

3 Preprocessing KB and Questions

3.1 Processing FREEBASE

Freebase : Freebase (Bollacker et al., 2008) is a huge and freely available database of factual information, organized as triplets (subject Entity, Relationship, object Entity). All Freebase entities and relationships are typed and the lexicon for types and relationships is closed. Each entity has an internal id and a set of alternative names (called aliases, e.g. JFK for John Kennedy) that can refer to that entity in text.

The overall structure of Freebase is a hypergraph, in which more than two entities can be linked together in a n-ary fact. The underlying triple storage involves dummy entities to represent such facts, effectively making actual entities involved linked through paths of length 2, instead of 1. For example, a statement like *"A starred as character B in movie C"* is represented in Freebase as (A, 'starring', dummy entity), (dummy entity, 'movie', C), (dummy entity, 'character', B), where dummy entity has same internal id in all three facts.

To obtain direct links between entities in such cases, we modify these facts by removing the dummy entities and using the second relationship as the relationship for the new condensed facts. In our example, we condense aforementioned facts into two : (A, 'character', B) and (A, 'movie', C). Also, we label these condensed facts as siblings of each other. Therefore, (A, 'character', B) is **sibling** of (A, 'movie', C) and vice versa. Moving forward, we use the term 'fact' to refer to a triplet in Freebase, containing actual entities only (no dummy entities).

After above preprocessing, we represent each entity and relationship in KB as a vector $\in \mathbb{R}^d$. Each such entity/relationship vector is computed as the average of its word vectors, where each word vector $\in \mathbb{R}^d$. In case of entities, we also include word vectors of all its aliases when calculating the average. Such a scheme has the additional advantage that we can benefit from pre-trained unsupervised word vectors (e.g. from Word2Vec), which in general capture some distributional syntactic and semantic information.

3.2 Processing Question

Let a question be given as sequence of words $(x_1, x_2, \ldots, x_{|q|})$. Each word x_i mapped to its word vector. We experimented with two different ways to compose the question embedding q out of these word-vectors:

Bag-of-words (BOW): It is sum of individual word vectors i.e. $q = \sum_{i=1}^{|q|} x_i$.

Position Encoding (PE): This heuristic take in account order of words in the question. The embedding is of form $q = \sum_{i=1}^{|q|} l_i \odot x_i$ where $\odot$ is an element-wise multiplication. For each i, l_i is a column vector $\in \mathbb{R}^d$ with the structure $l_{ij} = \min(\frac{i \times d}{j \times |q|}, \frac{j \times |q|}{i \times d})$, where d is the dimension of the embedding and j runs from 1 to d. This type of function ensures that initial part of summed vector is weighed higher for initial word vectors and vice versa. Thus, a statement like 'Who is parent of Darth Vader?' will map to a different embedding than statement like 'Who is Darth Vader parent of?'.

4 Model Description

4.1 Fact Extraction

To begin, we generate an initial list of facts (called candidate fact list) which is fed as input to our network. To generate this list, we match all possible n-grams of words of the question to aliases of Freebase entities and discard all n-grams (and their matched entities) that are a subsequence of another n-gram. All facts having one of the remaining entities as subject are added to candidate fact list.

4.2 Factual Memory Network

A L-Hop Factual Memory Network consists of L Computation Layers $C_1 \dots C_L$ connected one after another in a linear chain, and an Output Function. The initial Computation Layer C_1 takes as input the candidate fact list (from previous step) and the initial question embedding (from Section 3.2). The Output Function takes as input the output of final Computation Layer C_L and generates a list of answers (in form of Freebase entities). Layer C_i takes as input the output of layer C_{i-1}.

4.2.1 Computation Layer

A Computation layer accesses two inputs : 1) a fact list (our 'memory') $F = \{f_1 \dots f_{|F|}\}$, where each fact f is of form (s, R, o), s and o being the entity vectors and R the relationship vector, 2) a question embedding $q \in \mathbb{R}^d$. For each fact $f \in F$, we calculate an unnormalised score $g(f)$ and a normalized score $h(f)$.

We visualize a fact $f = (s, R, o)$ as a question-answer pair with (s, R) forming the question and o the answer. Therefore, $g(f)$ calculates similarity between the given question embedding and a hypothetical question of form $q' = (s, R)$.

$$g(f) = q^T(s + R) \tag{1}$$

For example, given a question q = *"Where was Bill Gates born?"* and a fact f = ('Bill Gates', 'place of birth', 'Seattle'), $g(f)$ will compute similarity between the question q and a hypothetical question q' of form *"Bill Gates place of birth?"*.

In case a fact f has some siblings $b_1, b_2, \dots, b_k$ (Section 3.1), we re-calculate its $g(f)$ as follows:

$$g(f) = \text{average}(g(f), g(b_1), g(b_2), \dots, g(b_k)) \tag{2}$$

where $g(f)$ on RHS is calculated using Eq.(1). For each sibling b_i, we calculate $g(b_i)$ using Eq.(1), but with b_i's subject replaced by its object in the formula.

Continuing with example in Section 3.1, if f = (A,'character',B) and b_1 = (A,'movie',C) is sibling of f, then this kind of processing is helpful in answering questions like *"What character does A play in movie C?"*, where fact f alone would not be enough to answer the question. Thus, above processing corresponds to a hypothetical question of form *"A character C movie ?"* for the fact f.

The normalized score $h(f)$ for a fact f is calculated using softmax function over all facts to determine their relative importance.

$$h(f) = \frac{exp(g(f))}{\sum_{f' \in F} exp(g(f'))} \tag{3}$$

Next we modify the fact list and the question embedding on basis of above calculated scores.

Step 1. Fact Pruning: We choose a threshold value ϵ and remove facts from fact list with $h(f) < \epsilon$. We found in our experiments that setting $\epsilon = \frac{\max_{f' \in F} h(f)}{2}$ gives best results. Performing pruning seems to remove non-relevant facts from subsequent computation, significantly improving our response time and allowing us to explore larger search space around the subgraphs of our question entities.

Step 2. The question embedding is modified as follows:

$$q' = q + \sum_{f \in F} h(f)(s + R) \tag{4}$$

Each such modification allows us to incorporate knowledge gathered (in form of hypothetical questions (s, R), weighted by their relevance h) at a particular layer about the question and pass it on to subsequent layer.

Step 3. Fact Addition: For each object entity o belonging to a fact f, we find all the facts (o, R', o') connected to it in KB and assign a score $h(f)q'^T(o + R')$ to each of them. If the new fact's score is $> \epsilon$, it is added to the fact list, effectively increasing path-length of our search space by 1.

The modified fact list along with the new question embedding(q') form the outputs of this layer, which are fed as input to next Computation Layer or the Output function.

4.2.2 Output Function

Output Function takes as input the output of final Computation Layer C_L (i.e. its output fact list and q') and calculate scores $h(f)$. The answer set is formed by the object entity of highest scoring fact as well as object entities of all those facts that have same s **and** R **as the highest scoring fact**.

This simple heuristic can increase utility of our model when there are multiple correct answers for a question. For example, a question like *'Who is*

Anakin Skywalker father of?' has more than one answer entities i.e. [*'Leia Organa'*, *'Luke Skywalker'*], and they are all linked by same s ('Anakin Skywalker') and R ('children') in Freebase. Of course, this follows the assumption that all such facts survive till this stage and atleast one of them is highest scoring fact.

4.3 Training Objectives

Let the QA training set $\mathcal{D}$ be set of question-answer pairs (q, A), where q is the question with list of correct answers A, e.g. (q = *'Who is Anakin Skywalker father of?'*, A = [*'Leia Organa'*, *'Luke Skywalker'*]). To train our parameters, we minimize the following loss function:

$$L_{QA} = \sum_{(q,A)} \sum_{n=1}^{L} \frac{n}{L} \| |F_n| \sum_{a \in A} a - |A| \sum_{f \in F_n} h(f).o \|^2 \quad (5)$$

Here n refers to n-th Computation Layer in our network (with F_n as its input fact list) and L is total number of Computation layers/hops in the network. This loss function defines the degree to which the object entities in fact list of a given layer are near to given answer list, weighted by $h(f)$, by taking pairwise difference between entities in answer list and objects in fact list. It was observed that minimizing this function gives higher weights to facts in which object entities are similar to answer entities as well as allow our network to generate shorter paths to reach answers from the question.

Paraphrasing Loss: Following previous work such as (Fader et al., 2013; Bordes et al., 2015), we also use the question paraphrases dataset **WikiAnswers**[1] to generalize for words and question patterns which are unseen in the training set of question-answer pairs. We minimize hinge loss so that a question and its paraphrases should map to similar representations. For the paraphrase dataset $\mathcal{P}$ of set of tuples (p, p') where p and p' are paraphrases of each other, the loss is defined as:

$$L_{PP} = \sum_{(p,p')} max\{0, 0.1 - p^T p' + p^T p''\} \quad (6)$$

where p'' is random example in $\mathcal{P}$ that is not a paraphrase of p and 0.1 is the margin hyper-parameter.

[1] http://knowitall.cs.washington.edu/paralex/

Backpropagation is used to calculate gradients while Adagrad was used to perform optimisation using max-norm regularisation. At each time step, a sample is drawn from either $\mathcal{P}$ with probability 0.25 or $\mathcal{D}$ with probability 0.75. If sample from $\mathcal{P}$ is chosen, gradient of L_{PP} is calculated. Otherwise, gradient of L_{QA} is calculated. The only parameters optimised in our model are the word vectors.

5 Experimental Setup

5.1 Baselines

We evaluate our model on following datasets:

WebQuestions[2]: This dataset, introduced in (Berant et al., 2013), contains 5,810 question-answer pairs where answer can be a list of entities, similar to (q, A) pairs described before. It was created by crawling questions through the Google Suggest API, and then obtaining answers using Amazon Mechanical Turk. WebQuestions is built on Freebase since all answers are defined as Freebase entities.

On **WebQuestions**, we evaluate against following baselines : (Berant et al., 2013; Berant and Liang, 2014; Yih et al., 2015) (semantic parsing based methods), (Fader et al., 2013) (uses a pattern matching scheme), (Bordes et al., 2014b; Bordes et al., 2014a; Bordes et al., 2015) (Embedding based approaches). Results of the baselines have been extracted from respective papers, except for (Berant et al., 2013; Berant and Liang, 2014) where we use the code provided by the author to replicate the results[2].

We compare our system in terms of F1 score as computed by the official evaluation script [2] (Berant et al., 2013), which is the average, over all test questions, of the F1-score of the sets of predicted answers.

SimpleQuestions[3]: The SimpleQuestions dataset, introduced in (Bordes et al., 2015), consists of a total of 108,442 questions written in natural language by human English-speaking annotators each paired with a corresponding Freebase fact. Our model only use the answer entity during the training, instead of whole fact. For example, {q = *'Which forest is Fires Creek in?'*, *Fact = '(fires creek, contained by, nantahala national forest)'*} could be data point in SimpleQuestions but we only use {q = *'Which forest is*

[2] www-nlp.stanford.edu/software/sempre/

[3] fb.ai/babi

Fires Creek in?', A = ['nantahala national forest']} for training.

On **SimpleQuestions**, we evaluate against previous result (Bordes et al., 2015) in terms of path-level accuracy, in which a prediction is correct if the subject and the relationship of highest scoring fact were correctly retrieved by the system.

5.2 Experimental Setup

The current dump of Freebase data was downloaded[4] and processed as described before. Our data contained 1.9B triplets. We used following splits of each evaluation dataset for training, validation and testing, same as (Bordes et al., 2015).
WebQuestions (WQ) : [3000, 778, 2032]
SimpleQuestions (SQ) : [75910, 10845, 21687]
We also train on automatic questions generated from the KB, which are essential to learn embeddings for the entities not appearing in either WebQuestions or SimpleQuestions. We generated one training question per fact following the same process as that used in (Bordes et al., 2014a).

The embedding dimension d was chosen 64 and max-norm cutoff was chosen as 4 using validation dataset. We pre-train our word vectors using method described by (Wang et al., 2014) to initialize our embeddings.

We experimented with variations of our model on both test sets. Specifically, we analyze the effect of question encoding **(PE vs BOW)**, number of Hops and inclusion of pruning/fact-additions **(P/FA)** in our model. In subsequent section, the word 'significant' implies that the results were statistically significant ($p < 0.05$) with paired T-test

6 Results

The results of our experiments are presented in Table 1. It shows that our best model outperforms considered baselines by about 3% in case of WebQuestions and even comparable to previous results in case of SimpleQuestions. Note that the best performing system for SimpleQuestions used strong supervision (question with supporting fact) while our model used only answer entities associated with a question for training.

[4]https://developers.google.com/freebase/data

Setup	WQ F1	SQ Acc
Random Guess	1.9	4.9
(Berant et al., 2013)	31.3	n/a
(Bordes et al., 2014a)	39.2	n/a
(Berant and Liang, 2014)	39.9	n/a
(Yih et al., 2015)	52.5	n/a
(Bordes et al., 2015)	42.2	**63.9**
PE + 3-Hop	**55.6**	59.7
BOW + 3-Hop	48.5	54.6
PE + 2-Hop	53.8	57.3
PE + 1-Hop	47.9	55.2
without P/FA	44.3	53.8

Table 1: Results on Evaluation datasets. Acc = Accuracy

We also give the performance for the variations of our model. Position Encoding improves our performance by 7% on WQ and by 5% on SQ, validating our choice of heuristic. Also, most answers in both datasets can be found within path length of two from candidate fact list, thus a 3-Hop network shows only 2% improvement over 2-Hop network.

	WQ	SQ
Top-2	70.1	68.7
Top-3	76.4	74.5
Top-5	80.3	77.6
Top-10	88.9	85.2

Table 2: Top-K results of our best model on each test set

Top-K Performance: In Table 2, we present the top-k results on both datasets. A large majority of questions can be answered from the top two or three candidates. By providing these alternative results (in addition to the top-ranked candidate) to the user, many questions can be answered correctly.

6.1 Efficiency and Error Analysis

Efficiency: All experiments for this paper were done on an Intel Core i5 CPU @ 2.60GHz, 8GB RAM with average HDD Access time of 12.3 ms. We calculated the Average Response time / Query (ART/Q), defined as average time taken by the system from input of query till the generation of answer list, including both computational and search-and-access time for KB. ART/Q for 1, 2 and 3 Hop Networks was 200, 350 and 505 ms respectively. Also the training time (including time to search for

hyper-parameters) for each of these networks was 740, 1360 and 2480 min respectively.

The major bottleneck in ART/Query for our network was the search-and-access of large amount of KB Data, therefore we implemented efficient search procedures using Prefix Trees for String Matching and pipelined different stages of the model using multi-threading (i.e. Fact Extraction, Computation and Back-Propagation were performed on individual threads) to improve our response and training time.

Effect of Pruning/Fact-Additions : From Table 1, we can see that pruning and fact-additions have significantly improved scores on all datasets. We analyzed 200 random data points from set of examples that were correctly answered only when P/FA was used (for each test set). In 97.5% of these samples, we observed that pruning allowed our model to remove spurious facts generated during initial fact extraction, making soft-max calculation in Eq. (3) more robust.

We also observed that the set of correctly answered questions using model without P/FA was proper subset of one with P/FA on each evaluation dataset, signifying that if relevant facts were scored higher in previous Computation layers, their scores are not reduced as more facts are added in subsequent layers. Removing pruning alone didn't improve our performance by more than 0.4% on any dataset while exponentially increasing the response time, signifying that pruning itself didn't remove relevant/correct fact in majority of examples.

On the computational end, we tried to determine the effect of pruning on our model response time (excluding search and access time). Including pruning improved our ART/Q by approximately 44% during test phase and by 21% during training phase for our 3 Hop network.

Manual Error Analysis : We sampled 100 examples from each test set to identify major sources of errors made by our model. Following classes of errors were determined :

Complex Questions (55%) : These types of questions involved temporal or superlative qualifier like 'first', 'after', 'largest', etc. This problem occurred in both test sets. We may be able to solve this problem using small set of rules for comparison on final answer set or better semantic representations for numerical values and qualifiers.

Question Ambiguity (20%) : This error class contains those questions that may have ambiguity in their interpretation. For example, a question like 'Where is shakira from?' generated answer as 'place_of_birth' (Baranquilla) while ground truth is 'nationality' (Colombia). This occurred mostly in WebQuestions dataset.

Ground truth Inconsistency (10%) : This type of error occurred when ground truth differed from correct entity present in Freebase KB (even though both are correct in many cases). For example, the question 'Where did eleanor roosevelt died?' have ground truth as 'New York City' whereas KB delivers the entity 'Manhattan', even though both are entities in Freebase. It occurred only in WebQuestions dataset.

Miscellaneous (15%) : This error class contains bad entity/relationship extraction (for example, mapping Anakin Skywalker to Darth Vader), bad question/answer pairs (e.g. q = "what time does american horror story air?" A = [Tom Selleck]), Typos in Question, etc.

7 Conclusion

This paper presents a Factual Memory Network model that aims to perform question-answering using facts from Knowledge bases like Freebase. The system uses a multi-hop neural network that can perform reasoning over facts generated from named entities in a given question as well as traverse the knowledge graph to include more information.

In future, we hope to extend our system so that it can work better with n-ary relations present in Freebase to deal with qualifiers, improve entity disambiguation mechanism in our model as well as include a mechanism to involve user interaction with system to improve our rates. Another goal is to add support for KBs with noisy data generated through automated relation extraction from unstructured data (for example OLLIE, etc) as well as for unstructured sources of knowledge (like Wikipedia) in our model, to extend and improve its utility.

References

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *Proceedings*

of the 24th ACM International on Conference on Information and Knowledge Management, pages 1431–1440.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1415–1425.

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.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 615–620.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014b. Open question answering with weakly supervised embedding models. In *Machine Learning and Knowledge Discovery in Databases*, pages 165–180. Springer.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Qingqing Cai and Alexander Yates. 2013. Semantic parsing freebase: Towards open-domain semantic parsing. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 328–338.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 260–269.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545.

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*, pages 1608–1618.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1156–1165.

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.

Thomas Lin, Oren Etzioni, et al. 2012. Entity linking at web scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 84–88.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 645–650.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Association for Computational Linguistics (ACL)*.