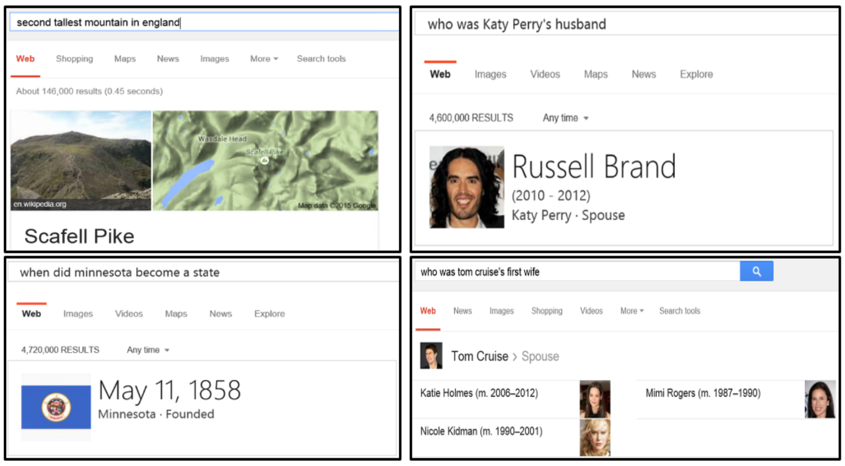
1 Introduction & Motivation

In this project we present an approach to answer factual questions based on Freebase database. The approach uses a combination of neural network based statistical learning of relations between entities and a probabilistic soft logic based inference module that uses the knowledge from the Freebase database.

Knowledge bases such as Freebase [7] and DBPedia [1] contain a huge amount of factual knowledge. The data is stored in a structured manner, and can be queried unambiguously by using structured languages such as SPARQL. Due to this, these knowledge bases have become important resources for supporting open-domain question answering (QA). Open-domain QA is one of the most popular NLP application today because it makes things extremely easy for the users. For example about two to three years ago Google search for *Where is Paris?* used to return a list of web-pages that point to the exact answer, but nowadays Google provides us the exact answer i.e. *France*. A few other examples of such queries are shown in figure 1.


second tallest mountain in england

About 146,000 results (0.45 seconds)

Scafell Pike

who was Katy Perry's husband

4,600,000 RESULTS

Russell Brand
(2010 - 2012)
Katy Perry · Spouse

when did minnesota become a state

4,720,000 RESULTS

May 11, 1858
Minnesota · Founded

who was tom cruise's first wife

Tom Cruise › Spouse

Katie Holmes (m. 2006–2012)

Mimi Rogers (m. 1987–1990)

Nicole Kidman (m. 1990–2001)
Web Shopping Maps News Images More Search tools
Web Images Videos Maps News Explore
Any time
en.wikipedia.org
Map data ©2015 Google
Web News Images Shopping Videos More Search tools

Figure 1: Google Search Query Results

Most state-of-the-art approaches to factual QA are based on semantic parsing, where a question (utterance) is mapped to logical form and then translated to a KB query. The answers to the question can then be retrieved simply by executing the query on the respective KB. The semantic parse provides a deeper understanding of the question, which can be used to justify the answer to users.

There are certain challenges that appear in the traditional approaches of translating the input question into a logical form. For example, the representation used for semantic parsing is not KB specific [15]. In other words, if the semantic translation is a graph then the edge labels in the KB and the semantic parse differ for the sentences with different words but same meaning. On the other hand, if the representation is some logic form then, the predicates vary with the words in the sentence. This causes a problem by creating different KB queries for two different looking questions[1] which have same meaning.

Even when the representation language is closely related to the knowledge base schema, finding the correct predicates from the large vocabulary in the KB to relations described in the utterance remains a difficult problem [6].

[1]For example the questions, *"What was the date that Minnesota became a state?"* and *"When was the state Minnesota created?"* have same meaning but different set of words.

In this report we present our approach that leverages the knowledge base and translates the input questions into a chain of entity nodes and edges that are relations in the KB. The first node in the chain is the topic entity in the input question and the ending node is the answer entity. The approach is inspired by [24, 23, 2] in which the authors define a query graph that can be straightforwardly mapped to a logical form in -calculus and is semantically closely related to -DCS [17].

There are three main steps in their approach of [24], namely, locating the topic entity in the question, finding the main relationship between the topic entity and the answer, and expanding the query graph with additional constraints that describe properties the answer needs to have, or relationships between the answer and other entities in the question.

We used the similar idea in this project but instead of expanding the query graph with additional constraints, we used a Convolutional Neural Network [13] and Long Short-Term Memory [11] based neural learning framework to step-wise learn the query chains from the topic entity to the answer entity. The features (explained in detail in later sections) used in the learning framework are designed in such a way that ideally they should accommodate the additional constraints present in the input question.

On top of learning the query chains from the input questions (above), we used those chains in the probabilistic soft logic [14] based inference engine to learn the probabilities of final answer(s) by learning the weights of inference rules from Freebase data.

We applied our approach on a well know WebQuestions [5] corpus. The evaluation and test results on the corpus are available in the later sections of this report.

2 Related Works

Answering natural language questions over structured knowledge-bases such as Freebase, has been explored mainly with two goals: i) advancing open-domain semantic parsing of natural language questions and ii) overcoming the representational and linguistic variability which makes it incredibly hard to map words or phrases in natural language to entities and relations in structured knowledge-bases.

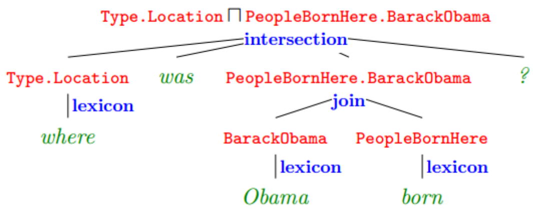


Figure 2: An example of the λ-DCS query created from q question in Freebase ([5]).

The first of these works (such as [5]) on Factoid Question-Answering was mainly on Semantic Parsing with the goal of having an unambiguous mapping between the semantic representation and the nodes and edges of a target structured Knowledge Base. In this work, a sophisticated representation formalism called λ-DCS was used and each question was mapped to this query form (as shown in Figure 2) and this query was executed over Freebase to get the answer. This paper also introduced the WebQuestions dataset which over the years, became the standard dataset to use for factoid QA over Freebase. Though the structural robustness is evident, the accuracy suffered from reasons like entity-disambiguation (41M entity lexicon) and mapping problems from natural language to binary DCS relations that were created.

These works were soon followed by the research from the second genre where people concentrated more on the entity and relation prediction from the natural language questions. The works done in this area includes [23, 21, 24]. In these works, the main challenge that has been attempted to solve, is to map the noun-phrases in the question to entities and question sentences to a single (or more) relation(s) in Freebase. In most of the cases, it has been observed that WebQuestions dataset is still not big enough to employ powerful learning algorithms such as Neural Networks ([24, 21]) and different modules have been added to increase performance of the systems. These modules either attempt to solve the prediction problem jointly (using unsupervised learning ot ranking algorithms) or include knowledge from other knowledge sources such as Wikipedia.

There are works ([4]) where a ranking technique has been employed where a set of predefined templates, candidate entities and relations are ranked based on an Information Retrieval based technique.

3 Corpus & Evaluation

3.1 Corpus

We selected the well known WebQuestion [5] corpus for this project. Following are a number of features it has:

- It consists of 5,810 question/answer pairs.
- It contains questions collected using the Google Suggest API
- The answers were obtained from Freebase with the help of Amazon MTurk.
- The corpus is divided into training (3,778 questions (65%)) and testing sets (2,032 questions (35%))

4 Our Approach

In our approach we develop a system to answer factual questions whose answers are present in a factual database called Freebase. We used a publicly available corpus named WebQuestions corpus and performed out tests on the corpus. The section below provides an overview of our overall approach and the later sections provide technical details.

4.1 Approach Overview

The overall idea of this project is inspired from the [21] and [24].

Let us first briefly describe the core algorithm that is in general followed by the community to answer questions from a structured knowledge base:

- **Entity:** Predict set of candidate/topic entities.
- **Relations:** Detect the sequence of relations that connect the "topic entity" to the "answer" in the Freebase graph.

In [24], authors provide a robust definition of such a chain of relations and term this as the **core-inferential chain**. In this work, a question is represented using a query graph as shown in the Figure 3. In a query graph, the nodes are entities

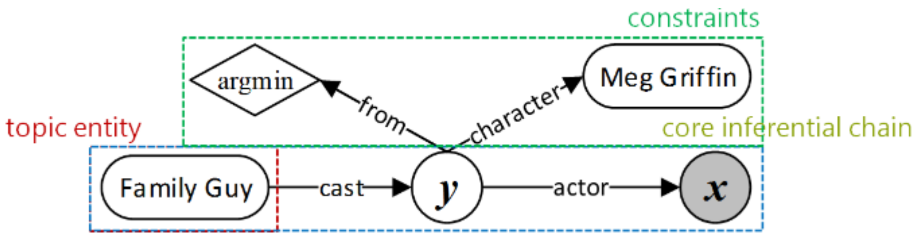


Figure 3: Question: "Who first voiced Meg on Family Guy?". The core Inferential chain is the sequence of (Freebase) relations (or in other terms, edges) than connect the "Topic Entity" to the final answer. In this case, it is "cast-actor".

or constraints. Entities can be directly mapped to Freebase entities or it can be symbolic too (i.e. variables that will be mapped to a Freebase entity). Constraint nodes are created to represent the qualifier adjectives or adverbs in the questions such as "first", "second", "tallest", "fastest" etc. As shown in the figure, based on their definition, an input question is assumed to have a primary topic entity. In the example, the topic entity is "Family Guy"[2]. Finally the **core-inferential chain** is defined as the sequence of relations connecting the topic entity to the answer-node. There could be multiple branches for each intermediate nodes (basically can be thought of as properties that this chain should satisfy).

- **Refine Predictions:** As the data is quite small and the above predictions result in poor accuracy, refine the results using some joint method.

- **Query Graph:** Using the topic entity and the sequence of relations, get the answer node by querying freebase.

In this work, we follow a similar method with some variations. For each question, we get the candidate entities using the entity linking tool S-MART ([22]). Then, based on the idea of "core inferential chain" described above, we predict the chain directly from features from a question using two different Neural Network architectures. Understandably, due to the smaller size of training data and large number of possible relations, the accuracy suffers and we employ a refinement algorithm. In the other works, different stages of predictions and graph querying is employed. In our work, we Probabilistic Soft Logic provides the theoretical robustness of logic and mathematical basis for correctly encoding joint dependencies, we employ a PSL module to refine the predictions and answer the question in one shot.

In short, our system has two parts namely the core inference chains learning module and the probabilistic soft logic module that infers the final answer based on the inferential chains and the knowledge from Freebase.

[2]Note, there could be other entities. But the assumption is that answer can be found if we concentrate on the nodes reachable from the topic entity.

4.2.2 Convolutional Neural Network with K-Parser

Below is a flow diagram 5 of the CNN and K-Parser based approach that we used to predict the core inferential chains for each question in the corpus. There are several modules shown in the flow diagram such as the topic entity and linking module and the answer set programming based path extractor module. These sub-modules along with where they fit in the system flow are explained below in detail.

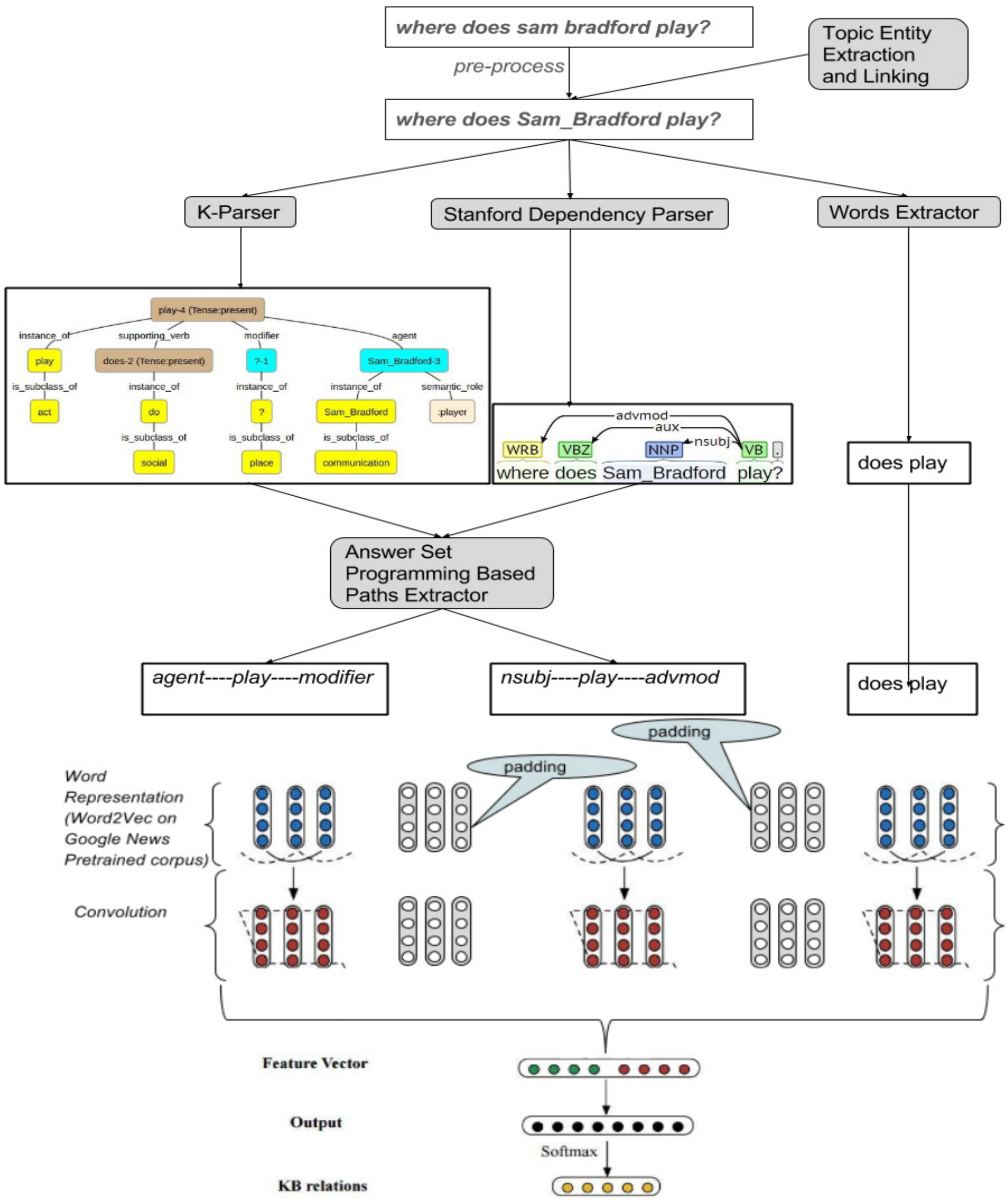


Figure 5: System Flow Diagram

Topic Entity Extraction & Linking Topic entity is the main entity present in the input question. The answer to the question is some property related to the topic entity. For example in the question, *where does sam bradford play?*, the topic entity is *Sam_Bradford*. And the answer to the question is the *Philadelphia Eagles* football team, and the position where *Sam_Bradford* plays is *Quarterback*. Each question in the WebQuestions corpus has one topic entity and the answer is some property of that entity only.

So, the first step of answering these questions is to find the correct topic entity in the questions and then linking it to the correct entity node in the Freebase database. We used an already processed data from an advanced entity linking system for short text [22]. The entities taken from the above data are then used to process the input questions. In the processing, every occurrence of the topic entity in the question is replaced by the actual entity in the Freebase database. For example the question *where does sam bradford play?* is processed to *where does Sam_Bradford play?*. The Freebase id for *Sam_Bradford* i.e. */m/02qjht0* is also saved as meta-data for later use. The algorithm to process the question and replace every occurrence of the topic entity in the input question is shown below.

Algorithm 1: Algorithm for Preprocessing Questions

```
1  $processedQuesMap \leftarrow newMap()$;
2  for $qID \in mapOfQuestions$ do
3      $ques \leftarrow mapOfQuestions.get(qID)$;
4      $listOfEntities \leftarrow getEntities(qID)$;
5      $newListOfEntities \leftarrow newList()$;
6      $newListOfEntities \leftarrow getMortProbableEntities(listOfEntities)$;
7      $insertionSort(newListOfEntities)$   ▷ Based in start index in the original
         question;
8      $proQues \leftarrow$ “”;
9      for $entity \in newListOfEntities$ do
10         $intentStartIndx = entity.getStartIndex()$;
11         $intentEndIndx = entStrtIndx + entity.getLabelLength()$;
12         $proQues =$
             $proQues + ques.substring(endIndx, entStrtIndx) + entity.getEntity()$;
13         $endIndx = entEndIndx$;
14     $processedQuesMap.put(qID, proQues)$;
15 return $processedQuesMap$
```

The data from [22] provides multiple possibilities for each entity present in the input question with different weights. In this project we have used the entities which had the maximum weights. For example, from the following results for *“sam bradford”* we chose the first one.

Entity in question	**Entity from Freebase**	**Freebase ID**	**Weight**
sam bradford	Sam_Bradford	/m/02qjht0	1000.0
sam bradford	Sam Bradford	/m/0k203gm	90.0
sam bradford	Samuel Bradford	/m/065zx3c	1.0

Convolutional Neural Network: After the questions are preprocessed, we implemented an idea inspired from [21] to extract the core inferential chains from the input questions. The idea is to extract both syntactic and semantic features from the input text, translate them into vector representations and then provide them as input to a convolutional neural network to learn the combined vector representation for the input question. Following different ways were used to extract the semantic and syntactic features.

K-Parser and Stanford Dependency Paths: We parsed the preprocessed questions from above and then parsed them using both K-Parser [19] and Stanford Dependency Parser [8]. Both the parsers output graphs/trees. We used those output and extracted the paths between each entity in the graphs. For example, as shown in figure **??**, the

K-Parser and Stanford Dependency Parser paths extracted for the question *where does Sam_Bradford play?* are *agent-play-modifier* and *nsubj-play-advmod* respectively. These paths contain the edge labels such as *agent* and *nsubj* and nodes in graphs such as *play*. Note that the paths do not contain any entities (Noun Phrases). Both of these graphs have similar structural properties, so we used a common Answer Set Programming module [10, 3] to extract the paths. The details of the module are explained below.

Answer Set Programming Based Path Extractor: This module takes as input any graph/tree in ASP format. We translated the input graph into RDF[4] style *has* triplets. Below are the K-Parser and Stanford Parser outputs for the input question *where does Sam_Bradford play?*.

```
% K-Parser output for \textit{where does Sam\_Bradford play?}
has(play_4,agent,sam_bradford_3).
has(play_4,modifier,q_1).
has(play_4,supporting_verb,does_2).
has(play_4,pos,verb).
has(sam_bradford_3,pos,noun).
has(q_1,pos,noun).
has(does_2,pos,verb).

% Stanford Dependency Parser output for \textit{where does Sam\_Bradford play?}
has(play_4,nsubj,sam_bradford_3).
has(play_4,advmod,q_1).
has(play_4,aux,does_2).
has(play_4,pos,verb).
has(sam_bradford_3,pos,noun).
has(q_1,pos,noun).
has(does_2,pos,verb).
```

The ASP module also has a set pf predefined logical rules that are used to find the paths. Various predicates are defined to extract the paths.

The first predicate is *ancestor* predicate. Its arity is two. Following rules defines the meaning of this predicate.

```
ancestor(A,X) :- has(A,R,X),R!=pos.
ancestor(A,Y) :- has(A,R,X), ancestor(X,Y),R!=pos.
```

The first rule means that if there is an edge R between two nodes A and X, where R is not *pos* (part-of-speech) label, then A is an ancestor of X. Similarly, the second rule means that if X is an ancestor of Y and A is a parent of X then A is an ancestor of Y.

The second predicate is *sameAncestor*. Its arity is three. Following rule is used to define the meaning of this predicate.

```
sameAncestor(X,A,Y) :- ancestor(A,X), ancestor(A,Y), X!=Y, not ancestor(X,Y),
                       not ancestor(Y,X).
```

This rule means that is A is an ancestor of X and A is an ancestor of Y where X is not same as Y and X and Y are not ancestors of each other, then A is a common ancestor of X and Y.

Another similar rule is used to define if a node is not a common ancestor of the two given nodes. The rule is:

[4] *https://www.w3.org/TR/PR-rdf-syntax/*

```
notSameAncestor(X,A,Y) :- sameAncestor(X,A,Y), ancestor(A,A1), A!=A1,
                          ancestor(A1,X), ancestor(A1,Y).
```

Four more predicates are defined to define four different types of paths as explained below the following rules.

```
longPath(X,R0,X1,R1,A,R2,Y1,R3,Y) :- sameAncestor(X,A,Y),
                                     not notSameAncestor(X,A,Y),
                                     has(X1,R0,X), has(A,R1,X1), has(A,R2,Y1),
                                     has(Y1,R3,Y), has(A,pos,v),R0!=pos,
                                     R1!=pos,R2!=pos,R3!=pos.
longPathLeft(X,R0,X1,R1,A,R2,Y) :- sameAncestor(X,A,Y),
                                   not notSameAncestor(X,A,Y),
                                   has(X1,R0,X), has(A,R1,X1), has(A,R2,Y),
                                   has(A,pos,v),R0!=pos,
                                   R1!=pos,R2!=pos.
longPathRight(X,R1,A,R2,Y1,R3,Y) :- sameAncestor(X,A,Y),
                                   not notSameAncestor(X,A,Y),
                                   has(A,R1,X), has(A,R2,Y1), has(Y1,R3,Y),
                                   has(A,pos,v),R3!=pos,
                                   R1!=pos,R2!=pos.
shortPath(X,R1,A,R2,Y) :- sameAncestor(X,A,Y), not notSameAncestor(X,A,Y),
                          has(A,R1,X), has(A,R2,Y),has(A,pos,v),
                          R1!=pos,R2!=pos.
```

The *longPath(X,R0,X1,R1,A,R2,Y1,R3,Y)* predicate (arity 9) finds a path between the entities X and Y through the nodes *X1, A,* and *Y1*, and edges *R0, R1, R2,* and *R3.* Here A is the least common ancestor of X and Y.

The *longPathLeft(X,R0,X1,R1,A,R2,Y)* predicate (arity 7) finds a path between the entities X and Y through the nodes *X1* and A, and edges *R0, R1* and *R2.* Here A is the least common ancestor of X and Y.

The *longPathRight(X,R1,A,R2,Y1,R3,Y)* predicate (arity 7) finds a path between the entities X and Y through the nodes A and *Y1*, and edges *R1, R2* and *R3.* Here A is the least common ancestor of X and Y.

The *shortPath(X,R1,A,R2,Y)* predicate (arity 5) finds a path between the entities X and Y through the node A, and edges *R1* and *R2.* Here A is the least common ancestor of X and Y.

The paths or semantic/statistic features extracted in the above paragraphs are then translated into vectors with the help of Word2Vec [18] tool. We used the Word2Vec model, which was pre-trained on Google news corpus, for translating both semantic and syntactic features into vectors.

These vectors are then fed into the convolution layers of the CNN used [13]. After the convolution layers, a softmax layer over the set of core inferential chains from the training corpus is used to classify the input question into the set of core inferential chains' set.

Various parameters such as the word vectors, neural network transition edge weights, and the network bias are learned during the training phase. These parameters are saved as a CNN model and then later used for classifying the test questions.

The output of the classifier is a set of core inferential chains. We have extracted the top 10 core inferential chains based on the output probability from the softmax layer.

These top 10 inference chains are later used in the probabilistic soft logic module defined in the following sections.

Intermediate Results:

The CNN with K-Parser approach resulted in the best validation accuracy of 42.07% and test accuracy is 35.86% while training (this train-test-validation splits are automatically chosen while performing 10-fold cross-validation). This is not accuracy over the test dataset.

References

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. *Dbpedia: A nucleus for a web of open data.* Springer, 2007.

[2] Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. Knowledge-based question answering as machine translation. *Cell*, 2(6), 2014.

[3] Chitta Baral. *Knowledge representation, reasoning and declarative problem solving.* Cambridge university press, 2003.

[4] Hannah Bast and Elmar Haussmann. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1431–1440. ACM, 2015.

[5] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, volume 2, page 6, 2013.

[6] Jonathan Berant and Percy Liang. Semantic parsing via paraphrasing. In *ACL (1)*, pages 1415–1425, 2014.

[7] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 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. ACM, 2008.

[8] Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454, 2006.

[9] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[10] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, volume 88, pages 1070–1080, 1988.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[13] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[14] Angelika Kimmig, Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. A short introduction to probabilistic soft logic. In *NIPS Workshop on Probabilistic Programming: Foundations and Applications*, 2012.

[15] Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. Scaling semantic parsers with on-the-fly ontology matching. In *In Proceedings of EMNLP. Percy.* Citeseer, 2013.

[16] Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

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

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[19] Arpit Sharma, Nguyen H Vo, Somak Aditya, and Chitta Baral. Towards addressing the winograd schema challenge-building and using a semantic parser and a knowledge hunting module. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, pages 25–31, 2015.

[20] Joao Silva, Luísa Coheur, Ana Cristina Mendes, and Andreas Wichert. From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*, 35(2):137–154, 2011.

[21] Kun Xu, Yansong Feng, Siva Reddy, Songfang Huang, and Dongyan Zhao. Enhancing freebase question answering using textual evidence. *CoRR*, abs/1603.00957, 2016.

[22] Yi Yang and Ming-Wei Chang. S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking.

[23] Xuchen Yao and Benjamin Van Durme. Information extraction over structured data: Question answering with freebase. In *ACL (1)*, pages 956–966. Citeseer, 2014.

[24] 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 of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1321–1331, 2015.