Hybrid Deep Open-Domain Question Answering

Ahmad Aghaebrahimian
Charles University, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské náměstí 25, 11800 Praha 1, Czech Republic
ebrahimian@ufal.mff.cuni.cz

Abstract

The explosion of textual data on the Internet over the past few years turns open-domain Question Answering into a scientifically challenging and commercially appealing research area.

In this thesis proposal, I explain my both completed and on-going experiments with a hybrid deep neural system for answering open-domain questions. This system supplements knowledge graphs with the free-texts searching techniques to address sparsity issues in knowledge graphs. It is designed to answer both factoid and non-factoid questions.

After an introduction and a review of the subject matter, I explain the architecture of the system and its components. Then I explain the implemented parts, which yielded state-of-the-art results on a factoid Question Answering test dataset. Finally, I elaborate on other components which are planned to be implemented in the near future.

Introduction

Question Answering (hereinafter, QA) as a commercially appealing and scientifically challenging research area has fascinated many engineers and scientists from the earliest days of artificial intelligence in the 60's. QA has been even suggested to replace Turing test as a measure of intelligence (Clark and Etzioni, 2016). It has gained a worldwide fame since IBM Watson (Ferrucci et al., 2010) beat two lifelong champions of Jeopardy quiz show in 2011 with a large margin.

Although many big strides have been made in recent years in this field, QA is still far from being solved. Until recently with some few exceptions (e.g. IBM Watson), QA was mainly limited to some expert systems in limited domains such as air traffic information systems.

Nowadays however, with current huge interest in information access, open-domain QA systems have attracted much more attention. The emergence of huge data repositories, textual data bases, knowledge graphs, various social networks, and generally the Internet in addition to the introduction of some advanced learning techniques such as Deep Neural Networks(DNN) have opened up an excellent opportunity for developing open-domain QA systems. These systems have posed a challenge and an opportunity in the field at the same time.

They are challenging because inferring the answer of a question among so much data is not a trivial task. They also offer an opportunity, because they pave the way for developing a wide array of useful applications such as dialogue systems, tutoring systems, scientist's assistants, etc.

In recent years, open-domain QA systems succeeded to make use of knowledge graphs efficiently and successfully (Aghaebrahimian and Jurčíček, 2016b, Berant et al., 2013b, Bordes et al., 2015).

Knowledge graphs (e.g. Freebase (Bollacker et al., 2007)) are big ontologies of concepts and their relations. They are clean, efficient and scalable data structures for entity detection and relation extraction. However, they are sparse and they lack many entities and properties [1]. It is because new concepts and properties are always being evolved and any newly compiled knowledge graph today, irrespective of how much comprehensive it is, will be obsolete after passing some relatively short time.

[1] An entity is a well-known thing, place or person. A property is an attribute which is defined over an entity.

Moreover, knowledge graphs are only able to provide answer for factoid questions. Factoid questions are those questions which asks about an entity. In contrast, non-factoid questions are mostly seeking for a non-entity answer such as a definition or any span of consecutive words.

Free-texts or non-factoid QA (Rajpurkar et al., 2016) is a rather new challenge in open-domain QA in which the questions are mostly non-factoid. Compared to factoid QA systems based on knowledge graphs, non-factoid QA is much more computationally demanding.

In order to make up for missing facts in knowledge graphs for answering factoid questions and answering non-factoid questions at the same time, I would like to develop a hybrid system which extracts answers from free texts and a knowledge graph simultaneously.

The extracted answer from free-texts module and the one from knowledge graph module both are analyzed for their confidence levels and the one with the highest confidence will be selected as the final answer.

My already implemented factoid systems (Aghaebrahimian and Jurčíček, 2016b, Aghaebrahimian and Jurčíček, 2016a) are able to produce state-of-the-art results and my work on a non-factoid system is progressing. I am optimistic that the integration of these two subsystems as a unified system could partially solve the challenge of open-domain QA.

In the rest of this text, in order to put my proposal in a proper context, I give a review on other similar systems in Section 1. Then, I elaborate on the notion of open-domain QA, its challenges and the way I try to address them in Section 2. Finally, I explain my both completed and ongoing experiments in Section 3 before I conclude in Section 4.

1 Related work

Natural Language Interface to Database (NLIDB) and Machine Comprehension (MC) systems are two main areas of research which have largely contributed to the development of QA systems. In an attempt to place my QA system detailed in Section 2 in a proper context, I will explain 1) NLIDB systems, 2) cloze-type QA, 3) non-factoid and, finally 4) factoid QA systems. [2]

Then, I will have a review on some common methods for QA in subsection 1.1 and finally, I will elaborate on common architectures of QA systems in subsection 1.2.

NLIDB was the earliest instance of domain-dependent QA systems. Baseball (Green et al., 1961) and Lunar (Woods, 1973) were among the first NLIDB systems.

Baseball was designed to answer questions about American baseball league and Lunar was a scientist's assistant which was able to answer questions about geology. They were intended to help people communicate with databases in a natural language. They used sophisticated language knowledge to translate users' questions into a standard database query. Although the syntactic parsing capacity of these systems was very limited, they were able to answer complex questions.

NLIDB systems did not flourish much in the 80s. One major reason for this decline was the complexity of the natural language which was used in these systems. Basically, their interface language was not much simpler than the database query language itself. According to (Androutsopoulos et al., 1995), they even diminished almost entirely in the 90s [3]. However, after the 90s, the advent of efficient semantic parsing approaches and the possibility of using natural language without any restriction helped to revive QA in some limited-domain applications such as geographical or public transportation QA systems. ARPA-sponsored project, ATIS in air traffic domain and many other projects in restaurant domain flourished in that time.

Compared to NLIDB systems, use of QA as a measure of MC (Kadlec et al., 2016, Hermann et al., 2015) is rather a new research area. The main goal of these systems is to make a machine read a text and then answer some multiple-answer (cloze-type) questions.

The first study in statistical MC dates back to (Hirschman et al., 1999) who devised the first data-driven approach for QA on a small dataset. The dataset included 600 elementary-school-level reading comprehension questions. Long before that, in the 60s, Philips A.V. developed Routine (Phillips, 1960) as the first rule-based open-

[2] There are other types of systems such as QA for solving mathematical problems (Liguda and Pfeiffer, 2011) or QA for answering questions which require complex reasoning (Khashabi et al., 2016) as well. However, these systems are not in the focus of this work.

[3] The reasons for the decline of NLIDB systems are elaborated in (Copestake and Sparck-Jones., 1989)

domain QA system for comprehension tests.

Routine is able to read a text and to answer simple reading comprehension questions. The system categorizes the tokens in a text into subject, object, verb, place and time expressions. Then, it analyzes each question based on the training data to decide what expression is the best match for it.

(Yang et al., 1999) used QA for MC in an open-domain context as well. In his approach, in the first step, the best sentence which possibly contains the answer to a given question is detected. Then, the actual answer can be extracted from this sentence using factor graph on multiple sentence (Sun et al., 2013), dependency trees (Shen and Klakow, 2006), or bootstrapping surface patterns (Ravichandran and Hovy, 2002).

The level of understanding in cloze-type questions is variable depending on the type of missing elements. The missing elements in these systems can be a simple word or a phrase or even a whole sentence. The possibility of using different types of missing elements makes different levels of difficulty in these systems (Hill et al., 2015). For instance, due to cohesion inherent in natural texts, finding prepositions is usually easy, while looking for named entities is more difficult.

Cloze-type QA provides a dependable measure for MC. However, in QA systems for other purposes like in dialogue systems or scientist's assistants, the answers are not known in advance. It is assumed that they are available somewhere on the Internet. Free-texts searching is a reasonable approach for these systems. In contrast to cloze-type questions, in free-texts or non-factoid QA systems (Rajpurkar et al., 2016), the answers, their boundaries and their types (e.g. proper noun, adjective, noun phrase, etc.) are not known in advance which makes this type of QA more challenging.

The last group of QA systems in my review are factoid or simple QA systems (Bordes et al., 2015, Aghaebrahimian and Jurčíček, 2016b, Aghaebrahimian and Jurčíček, 2016a). Simple QA is a factual retrieval technique which attempts to find an answer entity in a knowledge graph. Simple in this context does not mean that it is a simple task. It refers to the fact that answering these questions requires knowing just one property and one entity. Although the systems in this category scale well to big knowledge graphs (e.g. Freebase (Bollacker et al., 2007)), they suffer from sparsity which makes them unable to answer some questions.

1.1 Methods of Question Answering

All possible QA methods can be roughly categorized into three broad categories; Semantic Parsing (SP), Information Retrieval (IR)/Information Extraction (IE) and most recently, end-to-end Neural Networks(NN).

In QA systems based on SP (Clarke et al., 2010, Kwiatkowski et al., 2010, Wong and Mooney, 2007, Zettlemoyer and Collins, 2005, Zelle and Mooney, 1996) natural language utterances are translated into a logical representation of their meaning in a knowledge representation language like lambda calculus expressions (Carpenter, 1997), lambda-Dependency Compositional Semantics (Liang, 2013), robot controller language (Matuszek et al., 2012), etc. In this approach, a common procedure is to over-generate possible meaning representation candidates out of a limited number of predefined entities and properties (i.e. the lexicon). Then, these meaning representations are rated using different possible machine learning methods [4]. Eventually, the highest rated meaning representation is queried against a relevant database to fetch the answer.

The domain of available entities and properties in the lexicon of these systems is usually limited and small. This limitation makes it difficult for them to scale well to open-domain or even to a large ontology. Using big knowledge graphs like Freebase (Bollacker et al., 2007) or Wikipedia has been shown beneficial (Berant et al., 2013a, Cai and Yates, 2013, Kwiatkowski et al., 2013, Berant and Liang, 2014, Bordes et al., 2015, Aghaebrahimian and Jurčíček, 2016b, Aghaebrahimian and Jurčíček, 2016a) for expanding the scope of language understanding in such cases.

Like ontologies, knowledge graphs [5] are knowledge bases of entities and their relationships. Big knowledge graphs usually cover tens of domains and contains millions of entities [6]. The data in a

[4] Please see (Aghaebrahimian and Jurčíček, 2015) for a review on machine learning approaches for semantic parsing

[5] Although the structure which is mentioned here is generally valid for all knowledge graphs, it is described in the context of Freebase (Bollacker et al., 2007) which is used as the knowledge graph in this project.

[6] Freebase has more than 80 domains and 50 million enti-

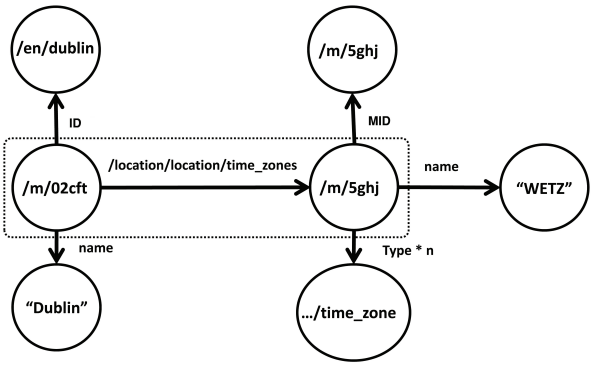

Figure 1: Freebase graph structure

knowledge graph is arranged in a specific structure called assertion. An assertion [7] is a small labeled, directed graph structure in which a subject entity is connected to an object entity. The connection is made by an edge which is labeled by an attribute about the subject. For instance, in Figure 1, "m/02cft" as the subject is connected to "m/5ghi" as the object. The connection is made by "time_zones" property.[8]

Subjects and objects in assertions have some attributes among which *"ID, MID, name, type"*, and *"expected type"* are the most important ones.

Each entity in the graph has one unique *"ID"* which is a human-readable code and one unique *"MID"* which is a machine readable code. *"name"* is a surface form and is usually a literal in form of raw text, date or a numerical value. Each edge (a.k.a property) has zero or at most one expected type and each entity has a set of *"types"*. For instance, *"time_zone"* is the expected type of *"/time_zones"* while *"/film/film/produced_by"* has no expected type.

The problem with QA systems based on knowledge graph is that their data in the best case, is limited to their knowledge graph beyond which, they are not able to recognize any new entity or property. Besides, in the best case, they are only able to answer factoid questions. Hence, in order to be used in open-domain QA systems, knowledge graphs should have the capacity of expansion in some ways.

Vast amount of works are devoted to knowledge graph expansion by adding new information which are extracted by parsing external textual corpora (Suchanek et al., 2007, Socher et al., 2013, Fader et al., 2011, Snow et al., 2005). However, arbitrarily adding data to knowledge graphs does not guarantee an effective data expansion procedure and it does not scale well through time. A more efficient and scalable way for knowledge graph expansion is to resort to an unlimited source of information like the Internet.

Use of knowledge graphs is common in IR/IE based systems as well. In contrast to SP-based QA systems, IR/IE-based systems (Yao and Durme, 2014, Bordes et al., 2015) directly retrieve or extract the answer from a database using different statistical methods. These methods range from simple techniques like Point-wise Mutual Information(PMI) (Church and Hanks, 1989) to more complex ones like deep neural networks (Aghaebrahimian and Jurčíček, 2016a, Dai et al., 2016).

Beside SP and IR/IE based systems, there are some other less-popular varieties like entailment-based systems (Bentivogli et al., 2008) or ensemble models (Clark et al., 2016) which are mostly proposed for domain-dependent QA systems.

In entailment approach, the system assumes that the entailment "question+answer" is available in the corpus and tries to detect the answer by finding the corresponding entailment. In ensemble models, an array of solvers each with specific inference algorithm is utilized to infer the answer. These algorithms may range from statistical IR to rule-base or constraint optimization solutions.

Constraint optimization is another active research area in QA in both SP and IE approaches. Constraint optimization using Integer Linear Programming (ILP) has been demonstrated being useful and effective in several NLP tasks (Chang et al., 2012, Srikumar and Roth, 2011,
Goldwasser and Roth, 2011, Chang et al., 2010, Roth and Yih, 2004). Especially, it has been reported to obtain state-of-the-art results for answering questions which require complex scientific reasoning (Khashabi et al., 2016).

Constraints are essentially Boolean inequalities which are defined based on some prior linguistic information (e.g. types of entities or properties). Prior information about linguistic structures plays a crucial role in structure prediction spe-

ties

[7] Please see the dotted line in Figure 1

[8] Properties are in abbreviated form to save space. For instance, the full form for this property is *"location/location/time_zones"*

cially, where there is not enough annotated training data or when models are too simple to detect long dependencies (Chang et al., 2012).

The last common method of QA is end-to-end neural network (NN) systems. QA in the context of NN is addressed mostly as a sequences prediction or classification/ranking problem. Different architectures in neural networks showed state-of-the-art performance in QA. They include but not limited to neural tensor networks (Socher et al., 2013), recursive neural networks (Iyyer et al., 2014), convolution neural networks (Yin et al., 2015, Dong et al., 2015, Yih et al., 2014), attention models (Hermann et al., 2015, Yin et al., 2015, Santos et al., 2014) and memory networks (Graves et al., 2014, Weston et al., 2015, Kumar et al., 2016, Sukhbaatar et al., 2015).

There are some reasons why use of DNNs attracted so much attention these days. First, a long-term goal of QA systems is to build general dialogue systems (Weston et al., 2016) and recently, end-to-end DNNs have shown great performance in dialogue systems. Second, the use of DNNs in QA helps to get rid of many cumbersome intermediate processes such as feature engineering at least in end-to-end systems. Finally, DNNs reduce the need for domain specific knowledge and makes domain adaptation easier.

1.2 Architectures of QA systems

A typical QA system includes some core and some peripheral subsystems. There are three core subsystems which can be recognized both in sequential pipelines (Turmo et al., 2009) and end-to-end architectures. They are question processing, passage processing and answer ranker components. Some QA systems may use other peripheral components or may join these components into a single one. In the following subsections, I explain core and some peripheral components of a typical QA system.

Question Processing. Each question entails an explicit or implicit intention and contains possibly one or more entities. Recognizing these intentions and entities (i.e. question understanding) is the job of question processing component.

In factoid QA systems, this component recognizes one property and one entity. In non-factoid QA systems, it extracts the intentions and their binded phrases and entities. For instance, in the question "what decision did NFL owners make on May 21, 2013?", a question processing component realizes "decision_made" as the intention and "NFL owners" and "May 21, 2013" as the entity and the time phrase which are binded to this intention.

In SP approach, Question processing component treats each question as a combination of arguments and predicates (Berant et al., 2013b). These combinations are trained using a machine learning technique to assign the highest rank to the combination which returns the correct answer.

In IR/IE approach, each question is processed either as a bag of words or as a low dimensional vector which is used to train a classifier to estimate a distribution over some predefined intentions given each question (Aghaebrahimian and Jurčíček, 2016a).

Passage Retrieval. Using the extracted key words, syntactic structures (e.g. dependency or constituency trees) or statistical models from a question processing component, a passage retrieval component obtains a document or a short passage like a paragraph which hopefully contains the answer. These passages can be obtained from a specific repository like Wikipedia articles or directly from the Internet. Passage retrieval is usually done using a local IR engine or a commercial search engine.

Although an accurate passage retrieval component reduces the computational complexity and answer space for the next component (i.e. Answer ranker), sometimes finding a passage which contains a desired answer is as hard as finding the answer itself. So, some systems eliminate the need for passage ranker component from the QA pipeline by providing the correct passages (Rajpurkar et al., 2016).

Answer Ranker. An Answer ranker component ranks the answers based on their relevance, accuracy, etc. and returns the highest or n-highest ones.

In factoid QA systems, the candidate answers are usually a named entity, a noun or a verb (Sun et al., 2005). In non-factoid QA, the constituents (e.g. noun phrases, adjective phrases, etc.) make majority of the answers. The rest of the answers are non-constituents or any arbitrary string of consecutive tokens and it makes the task of answer ranking much more difficult.

Symbolic and statistical techniques are the

two broad approaches for Answer ranking modules. A symbolic (or linguistic) approach assumes having access to rules, patters or other possible linguistic measures like similarity to determine whether a token or a series of tokens is the answer of a question. Similarity can be defined in terms of lexical (e.g. edit distance) (Aghaebrahimian and Jurčíček, 2016b) or syntactic measures (Moschitti and Quarteroni, 2010).

A statistical approach, in contrast, does not assume having access to such patterns. Instead, it has access to large quantities of data out of which it tries to derive some useful patterns. A common approach in statistical approaches is to check how probable is for an answer phrase to come with a questions phrase (Dumais et al., 2002, Lin and Katz, 2003).

Question processing and answer ranking can be both benefited through typing information. Typing systems may use named entity types (Wu et al., 2005), typing system of an ontology (Hovy et al., 2001), or available types in a knowledge graph (Aghaebrahimian and Jurčíček, 2016b, Aghaebrahimian and Jurčíček, 2016a). Type enforcement is done through hand-crafted lexical rules (Prager et al., 2000), syntactic rules (Magnini et al., 2002) or machine learning classifiers (Punyakanok et al., 2004).

Question processing, Passage processing and Answer ranker components are almost always detectable in all QA systems. They constitute the basic components of a QA system. In addition to them, it is possible to see QA systems with some extended capabilities like the ones which follow:

Question Analyzer. A Question analyzer component comes before a Question processing component. It extracts temporal and spacial information and binds them to the answers in the list of its Answer ranker component for retrieving finer answers (Kalyanpur et al., 2011, Hartrumpf et al., 2009). For instance, in the question "what decision did NFL make on May 21, 2013", the answer analyzer extracts "May 21, 2013" and helps the Answer ranker component to increase the score of the candidate answers which are relevant to this date.

Answer Selector. Each QA system is only able to answer a specific type of questions. A system which performs perfectly for IE-type questions performs treble for reasoning-type ones due to the fact that the answers of reasoning-type questions are not explicitly mentioned in a corpus. Therefore, ensemble models (Ko et al., 2007, Mendes and Coheur, 2011, Clark et al., 2016) which include a stack of QA systems with different architectures often outperform each of the included systems individually. In these systems, each model generates an answer and the final answer is chosen by an answer selector module which may use different possible approaches like ordering by likelihood, classification, etc.

Answer Validator. Providing answers with high precision is a critical task in some QA systems (Khani et al., 2016). In these systems, the answer validation component plays a crucial role. It is possible to validate answers using statistical or linguistic measures. In entailment checking as a statistical measure, an answer is validated by comparing it with its question to see if the former entails the latter (Wang and Neumann, 2007). Linguistic measures do the same job by reasoning over sentences which are generated out of a question and its answer using syntax rewriting rules (Bouma et al., 2005)

2 Hybrid Open-domain QA

In a broad perspective, QA systems fall into domain-dependent and domain-independent systems.

In domain-dependent systems, the questions are limited to a specific domain like health care. In these systems, the entities (e.g. different drugs, diseases, products, etc.) and properties (e.g. symptoms, side effects, etc.) are limited and already defined.

A domain-independent QA system in contrast, recognizes non-deterministic number of entities and properties. It requires having access to a big source of information. Some knowledge graphs are big enough to accommodate millions of entities and thousand of properties. However they are not dynamic hence, can not recognize new entities and they can not answer non-factoid questions. The web, on the other hand, is a dynamic source of information. However, compared to knowledge graph-based models, web-based QA models usually require much more computational power.

By hybrid open-domain system, I mean a unified system which is composed of a knowledge graph and a web-based QA models. Integrating

these two technologies in a system makes up for the deficiencies of each and makes the final performance superior to the one of each.

Big knowledge graphs contain enormous amounts of human knowledge. However, they suffer from sparsity. For instance, consider "parenthood" as a typical property in a typical knowledge graph like Freebase. One can query "Barack Obama" for his parents, however querying "Brad Pitt" for his parents returns no answer. Because "Brad Pitt" node in Freebase graph has no value for "parenthood" property.

In an open-domain system, new entities and properties are evolved continuously. Therefore, the system should be able to get new knowledge steadily. This process should be done automatically and without any human intervention. One feasible solution for this problem is to use the web and a big knowledge graph to support each other in a hybrid system.

In the rest of this section, I enumerate three categories of the questions that my hybrid system is designed to answer. Then, I explain the type of reasoning required to answer each category. Finally, I explain the architecture of my proposed system.

The following passage is used as a reference passage for all the following questions.

"On May 21, 2013, NFL owners at their spring meetings in Boston voted and awarded the game to Levi's Stadium. Troy Vincent, the NFL vice President then went to have a meeting with the authorities of the stadium.

My proposed system is designed to answer these three categories of questions.

1. Factoid questions with available answers in a knowledge graph:

   Factoid questions are answered using one property and one entity.

   Question: Who is the vice president of NFL?
   Entity: NFL
   Property: Vice President
   Answer: Troy Vincent
   Reasoning: knowledge graph types and lexical distance

2. Factoid questions with non-available answers in a knowledge graph:

   Like the questions in previous group, the answers to these questions are an entity. However, due to sparsity, their answers are not available in a knowledge graph.

   Question: Who did award the games to Levi's Stadium?
   Property: game_awarded_by
   Entity: Levi's Stadium
   Answer: NFL owners
   Reasoning: constituency, dependency and neural

3. Non-factoid questions with available answers in free texts:

   Unlike the questions in two previous groups, the answers to these questions are a definition, description or generally a span of consecutive words. These questions are answered using free-texts search techniques.

   Question: What decision did NFL owners make On May 21, 2013?
   Answer: Awarded the game to Levi's Stadium
   Reasoning: Neural

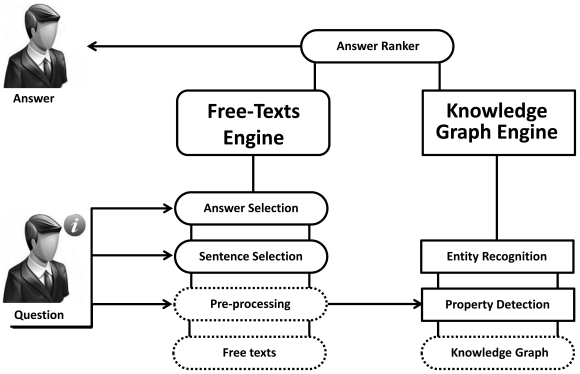

Figure 2: General system architecture

The components required for answering the first category of questions are already implemented and tested in (Aghaebrahimian and Jurčíček, 2016b, Aghaebrahimian and Jurčíček, 2016a). The components required for answering other two categories of questions are to be implemented in the near future. All these components are depicted in Figure 2.

The dotted-line components in Figure 2 are for doing pre-processing tasks for which available off-the-shelf toolkits are used. They include:

- Preprocessing component which contains a syntactic parser and an embedding module. The syntactic parser generates constituency and dependency parse trees and recognizes named entities for the system. Stanford Core NLP toolkit is used in this module. It also uses NLTK toolkit to perform some preprocessing and normalization tasks like tokenization, removing stop words, removing punctuation, etc. The embedding module is implemented in three ways; as a look up table which is trained using available training data, as a pre-trained model using Word2Vec (Mikolov et al., 2013) toolkit and as a pre-trained model using Glove vectors (Pennington et al., 2014).

- Free texts component which is a corpus of Wikipedia articles included in SQuAD (Rajpurkar et al., 2016) dataset.

- Knowledge graph which is a copy of Freebase (Bollacker et al., 2007) loaded into Virtuoso engine.

The right-angle rectangles in Figure 2 are components of the knowledge graph engine in my system. They are already implemented and yielded state-of-the-art results on a QA dataset. They include:

- Property detection component which returns a distribution over the system's properties given each question.

- Entity recognition component which selects the best matching entity given each question.

The round rectangles in Figure 2 are components of the free-texts engine in my system. They are on-going projects and will be implemented in the near future. They include:

- Sentence selection component which selects the best possible answer sentence given a question.

- Answer selection component which selects the shortest best answer from the best selected sentence.

- Answer ranker component which ranks the answers of the knowledge graph and free-texts engines.

In the rest of this section, I explain the knowledge graph and the free-texts engines in detail.

2.1 Knowledge Graph Engine

The knowledge graph engine is optimized to answer factoid questions (i.e. questions in the first category). It makes use of two components; property detection and entity recognition (Section 3.2).

The property detection component uses different models to estimate a distribution over its knowledge graph properties given each question. It generates a list of n-best properties for each question.

Given a list of n-best property and a question, the entity recognition component reasons over all possible entities available in the question to select the best one. The reasoning process is two folded; reasoning over the types of the entities [9] and reasoning over the similarity between knowledge graph entities and question entities. It computes a score for each possible combination of n-best properties and entities in the question and returns the highest scored combination as the best answer.

2.2 Free-Texts Engine

The free-texts engine is planned to be implemented in the near future. It answers non-factoid questions (category two and three of the questions explained above). It includes two main components; sentence selection and answer selection.

Sentence Selection. Since the search space in a paragraph for searching for a span of words as an answer is too huge, in the first step, the sentence selection component selects a sentence which contains the answer and limits the search space to the spans available in it.

In some applications, we are interested not only in the answer but also in the evidence based on which the question is answered. The best sentence selected by this component is also used as an evidence to support the extracted answer.

Answer Selection. Given a question and its best selected sentence by the sentence selection component, the job of the answer selection component is to extract the best shortest span of consecutive words as the final answer.

Answer Ranker. The answer extracted from the free-texts engine in addition to the answer chosen by the knowledge graph engine accompanied

[9] For more about knowledge graph typing system please refer to Fig.1 and the following description.

with their features (scores, types, etc) are fed into a simple classifier which probabilistically decides which one is the correct answer. This is done in an overall system optimization phase when two subsystems perform successfully enough.

3 Experiments

In this section, I describe three standard datasets that I used in my experiments. Then, I explain my completed experiments and their results. Finally, I explain the settings of an on-going experiment. The experiments on the first dataset deal with the knowledge graph engine and the experiments on the second and the third datasets deal with the free-texts engine.

3.1 The Datasets

3.1.1 SimpleQuestions

SimpleQuestions(SQ) (Bordes et al., 2015) is a collection of 108,442 questions composed in natural language. Each question in the dataset is mapped to an assertion in Freebase. The dataset is randomly shuffled and divided into train (70%), development(10%) and test (20%) sets. The experiments on SQ dataset are evaluated based on path accuracy. In path accuracy measure, an answer is considered correct if both the detected property and the entity are correct.

3.1.2 TrecQA

TrecQA (Yao et al., 2013) is a standard and well-studied benchmark for answer sentence selection experiments. It is compiled from the data in TREC 8-13 QA tracks. Each question in TrecQA is mapped to more than one correct and a couple of wrong sentences. Since the questions in TrecQA have more than one correct sentence, the most suitable measures for evaluation purposes are Mean Average Precision(MAP) and Mean Reciprocal Rank(MRR)

3.1.3 SQuAD

Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) is a dataset for QA in the context of MC. It includes 107,785 question-answer pairs posed by crowd workers on 536 Wikipedia articles. The answer can be any span of words in a paragraph which comes with each question.

The dataset is randomly shuffled and divided into train (80%), development (10%) and test (10%) sets. Compared to earlier datasets for MC (Hirschman et al., 1999, Richardson et al., 2013), SQuAD is significantly larger which makes it a good testbed for data-intensive methods such as deep neural networks.

	Train Set	Dev. Set	Test Set	evaluation
SimpleQuestions	75909	10844	21688	Path Acc.
TrecQA	1229	82	100	MRR/MAP
SQuAD	86228	10778	10778	F1/Exact match

Table 1: Datasets statistics

The two metrics used for SQuAD evaluation are exact match and F1. Exact match measures the percentage of predictions which exactly match the gold answer. F1 measures the overlap between a prediction and its golden answer given each question. Table 1 shows the number of questions in these three datasets with their appropriate evaluation method.

3.2 Completed Experiments with SQ

The task in these series of experiments is to predict the best property and entity given each question. We decompose the task into property detection and entity recognition.

Since the boundary of tokens in questions is not given and we assume no access to any pre-compiled lexicon, the task is quite challenging. First, I explain property detection and then continue with entity recognition.

In property detection, we train a model to return an n-best property list for each question. This is formulated as a classification task. The questions in this task can be represented using bag-of-words technique in which the tokens in the questions are represented as one-hot vectors. However, in order to address sparsity problem inherent in one-hot vectors, we used word embeddings in all our models.

For property detection component, we defined and implemented three models; logistic regression, multi-layer Neural Network and Convolution Neural Network.

- Logistic Regression: In this experiment (Aghaebrahimian and Jurčíček, 2016b), we used concatenated embeddings of the words in each question as the features of a logistic regression model. The size of the concatenated array is fixed to the size of the longest question. Although this representa-

Experiments	Acc.
State-of-the-art (Bordes et al., 2015)	61.6
LR model	61.20
NN model	63.89
CNN model	**65.16**

Table 2: Experimental results on test set of SimpleQuestions (SQ) dataset. LR stands for logistic regression model, NN for neural network and CNN for Convolution neural network models.

tion is not very efficient, the model obtained competitive results to a state-of-the-art system (Bordes et al., 2015).

- Multi-layer Neural Network: This model (Aghaebrahimian and Jurčíček, 2016a) is the extension of the logistic regression model by adding two hidden layers each with 1024 neurons and a softmax layer with 50% dropout rate at the end. It boosted the classification accuracy by almost 4 percent. With this improvement, the whole system outperformed the state-of-the-art system.

- Convolution Neural Network (CNN): The CNN model contains four consecutive layers; embedding, convolution, max pool and soft max layers. The first layer embeds words into low dimensional vector representations. For this layer, we adopted two approaches; one, self-training from a uniform random distribution and two, using pre-trained Word2vec word embeddings. The second layer 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. The CNN model yields the best results among all other models.

Having an n-best property list generated by the property detection component, the next step is entity recognition which includes entity detection and entity disambiguation.

A simple question with 10 tokens can have near to a thousand entities in Freebase. For instance a token like "dublin" has more than hundreds of instances like the names of different locations, bands, books, etc. In entity detection step, all these instances (i.e. entities) in a question are extracted.

Since the boundaries of the entities in questions are not given, we have to consider all possible spans as candidate entities. It sums up to all possible n-grams in the questions excluding those that are not available in the knowledge graph. To extract these entities, we slide a flexible size window over each question and query each span as a surface form.

Surface forms are saved as values to "name" or "alias" properties of entities. By detecting an exact match between the span and "name" or "alias" of an entity, we can get the "MID" identifier for that specific entity.

Word spans with at least one entity MID are recognized as valid entities. The permutation of these valid entities and the n-best property list returns tuples of subject entity and one property. If we query each tuple against the knowledge graph, we receive either nothing or an object entity. In this step, we retain entities which returns an object entity.

The detected entities in the last step (a.k.a. entity detection) in many cases are still ambiguous. In entity disambiguation step, I heuristically used lexical distance to obtain the entity which has the highest similarity with a surface form in a given question. I assumed that the lexical similarity ratio between *"ID"* and *"name"* properties connected to the correct entity is maximal among other false entities. Finally, the tuple with the highest similarity score is selected as the correct answer. Table 2 are the results of all above experiments measured in path accuracy.

3.3 Experiments with SQuAD and TrecQA

Given a question and a paragraph, the task in free-texts QA is to extract the shortest possible span out of the accompanying paragraph. Statistically, I am interested in estimating

$$p(\mathbf{a}|\mathbf{q},\mathbf{p})$$

where $\mathbf{a}$ is an answer given question $\mathbf{q}$ and paragraph $\mathbf{p}$. Maximizing this probability returns the most probable answer given each question.

$$a_{best} = argmax_a \quad p(\mathbf{a}|\mathbf{q},\mathbf{p})$$

As I explained in Section 2, the first step is sentence selection. Therefore, the objective above is decomposed into following objectives

$$s_{best} = argmax_{s\in\mathbf{p}} \quad p(\mathbf{s}|\mathbf{q})$$

$$a_{best} = argmax_{a \in \mathbf{s_{best}}} \quad p(\mathbf{a}|\mathbf{q})$$

For doing so, I am planning an experiment on TrecQA and SQuAD using a new neural architecture. The overall idea is that we can train a neural model using a loss function which learns the questions and sentences vectors by maximizing the similarity between questions and their correct sentences and minimizing the similarity between questions and their wrong sentences. My experiments on this system is progressing [10]. I intend to use a similar architecture for the answer selection component and I am hopeful that it obtains positive results as well. If it doesn't, I will consider a constraint optimization approach using an ILP model constrained by the constituency types of the questions and their passages.

4 Conclusion and Future Work

The final goal of my current research is to develop a hybrid system which integrates knowledge graphs and free texts together for answering open-domain factoid and non-factoid questions. To the best of my knowledge, the proposed system is the first fully neural system integrating knowledge graphs and free texts as a hybrid QA system.

In my work I use mainly DNNs, which is an evident decision because of the state-of-the-art performance of these models not only in my completed experiments, but also in many other NLP tasks. In my experiments, I showed that DNNs can produce state-of-the-art results for factoid QA in a knowledge-graph-based system. However, for a robust open-domain QA system, knowledge-graph based systems should be supplemented with free-texts search techniques to make up for the missing data in knowledge graphs and also to answer non-factoid questions. I plan to pursue this objective by fulfilling the following steps in my future work:

1. Completing and optimizing my algorithm for the Sentence selection component
2. Developing an algorithm based on deep learning for the Answer selection component
3. Doing more experiments with more datasets to enhance both algorithms mentioned above
4. Optimizing the Answer ranker component after the completion the previous tasks

[10] This experiment has already conducted successfully and the results are submitted to a conference.

Acknowledgments

This work 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

[Aghaebrahimian and Jurčíček2015] Ahmad Aghaebrahimian and Filip Jurčíček. 2015. Machine learning for semantic parsing in review. In *Proceedings of 7th Language and Technology Conference, Poznan, Poland*.

[Aghaebrahimian and Jurčíček2016a] Ahmad Aghaebrahimian and Filip Jurčíček. 2016a. Constraint-based open-domain question answering using knowledge graph search. In *Proceedings of the 19th International Conference on Text, Speech and Dialogue(TSD), LNAI 9924*.

[Aghaebrahimian and Jurčíček2016b] Ahmad Aghaebrahimian and Filip Jurčíček. 2016b. Open-domain factoid question answering via knowledge graph search. In *Proceedings of the Workshop on Human-Computer Question Answering, The North American Chapter of the Association for Computational Linguistics*.

[Androutsopoulos et al.1995] I Androutsopoulos, G Ritchie, and P. Thanisch. 1995. Natural language interfaces to databases, an introduction. *Natural Language Engineering*.

[Bentivogli et al.2008] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2008. The sixth pascal recognizing textual entailment challenge. *Text Analysis Conference(TAC)*.

[Berant and Liang2014] J. Berant and P. Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the Association for Computational Linguistics*.

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

[Berant et al.2013b] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013b. Semantic parsing on freebase from question-answer pairs. In *Proceedings of Empirical Methods in Natural Language Processing*.

[Bollacker et al.2007] Kurt Bollacker, Patrick Tufts, Tomi Pierce, and Cook Robert. 2007. A platform for scalable, collaborative, structured information integration. In *Proceedings of the Sixth International Workshop on Information Integration on the Web*.

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

[Bouma et al.2005] Gosse Bouma, Jori Mur, and Gertjan van Noord. 2005. Reasoning over dependency relations for qa. In *Proceedings of the IJCAI workshop on Knowledge and Reasoning for Answering Questions (KRAQ)*.

[Cai and Yates2013] Q. Cai and A. Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. *Proceedings of the Association for Computational Linguistics*.

[Carpenter1997] Bob Carpenter. 1997. Type-logical semantics. In *The MIT Press*.

[Chang et al.2010] Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Vivek Srikumar. 2010. Discriminative learning over constrained latent representations. In *Proceedings of the Workshop on Human-Computer Question Answering, The North American Chapter of the Association for Computational Linguistics*.

[Chang et al.2012] Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2012. Learning with constrained conditional models. *Machine Learning*, 3(88).

[Church and Hanks1989] K. W. Church and P Hanks. 1989. Word association norms, mutual information and lexicography. In *Proceedings of the Association for Computational Linguistics*.

[Clark and Etzioni2016] P. Clark and O. Etzioni. 2016. My computer is an honor student but how intelligent is it? standardized tests as a measure of ai. *AI Magazine*, 37(1):5âĂŞ12.

[Clark et al.2016] Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Turney, and Daniel Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *Proceedings of the 30th AAAI*.

[Clarke et al.2010] Clarke, Dan James, Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Conference on Computational Natural Language Learning*.

[Copestake and Sparck-Jones.1989] A. Copestake and K. Sparck-Jones. 1989. Natural language interfaces to databases. *Knowledge Engineering Review*.

[Dai et al.2016] Zihang Dai, Lei Li, and Wei Xu. 2016. Cfo: Conditional focused neural question answering with large-scale knowledge bases. In *Proceedings of the Association for Computational Linguistics*.

[Dong et al.2015] 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 and the 7th International Joint Conference on Natural Language Processing*.

[Dumais et al.2002] Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. 2002. Web question answering: Is more always better? In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, ACM*.

[Fader et al.2011] Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of Empirical Methods in Natural Language Processing*.

[Ferrucci et al.2010] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya a. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3):59âĂŞ79.

[Goldwasser and Roth2011] Dan Goldwasser and Dan Roth. 2011. Learning from natural instructions. In *Proceedings of International Joint Conference on Artificial Intelligence(IJCAI)*.

[Graves et al.2014] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. In *arXiv:1410.5401v2*.

[Green et al.1961] Bert F. Jr. Green, Alice K. Wolf, Carol Chomsky, and Kenneth Laughery. 1961. Baseball: an automatic question answerer. *Massachusetts Institute of Technology- Lincoln Laboratory*.

[Hartrumpf et al.2009] Sven Hartrumpf, Ingo Glockner, and Johannes Leveling. 2009. Efficient question answering with question decomposition and multiple answer streams. In *Evaluating Systems for Multilingual and Multimodal Information Access*.

[Hermann et al.2015] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*.

[Hill et al.2015] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading childrens books with explicit memory representations. In *arXiv :1511.02301*.

[Hirschman et al.1999] Lynette Hirschman, Marc Light, Eric Breck, and John D. Burger. 1999. Deep read: A reading comprehension system. In *Proceedings of the Association for Computational Linguistics*.

[Hovy et al.2001] Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. 2001. Question answering in webclopedia. In *Proceedings of the Tenth Text REtrieval Conference (TREC-2001)*.

[Iyyer et al.2014] Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal DaumÃl' III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of Empirical Methods in Natural Language Processing*.

[Kadlec et al.2016] Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the Association for Computational Linguistics*.

[Kalyanpur et al.2011] Aditya Kalyanpur, Siddharth Patwardhan, Branimir Boguraev, Adam Lally, Chu-Carroll, and Jennifer. 2011. Fact-based question decomposition for candidate answer re-ranking. In *Proceedings of the 20th ACM international conference on Information and knowledge management*.

[Khani et al.2016] Fereshte Khani, Martin Rinard, and Percy Liang. 2016. Unanimous prediction for 100semantic mappings. In *Proceedings of Association for Computational Linguistics*.

[Khashabi et al.2016] Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Peter Clark, Oren Etzioni, and Dan Roth. 2016. Question answering via integer programming over semi-structured knowledge. In *Proceedings of International Joint Conference on Artificial Intelligence(IJCAI)*.

[Ko et al.2007] Jeongwoo Ko, Luo Si, and Eric Nyberg. 2007. A probabilistic graphical model for joint answer ranking in question answering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*.

[Kumar et al.2016] Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of The 33rd International Conference on Machine Learning*.

[Kwiatkowski et al.2010] 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*.

[Kwiatkowski et al.2013] 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*.

[Liang2013] Percy Liang. 2013. Lambda dependency-based compositional semantics. In *arXiv:1309.4408*.

[Liguda and Pfeiffer2011] Christian Liguda and Thies Pfeiffer. 2011. A question answer system for math word problems. In *Proceedings of the First International Workshop on Algorithmic Intelligence*.

[Lin and Katz2003] Jimmy Lin and Boris Katz. 2003. Question answering from the web using knowledge annotation and knowledge mining techniques. In *Proceedings of the twelfth international conference on Information and knowledge management, ACM*.

[Magnini et al.2002] Bernardo Magnini, Matteo Negri, Roberto Prevete, and Hristo Tanev. 2002. Mining knowledge from repeated cooccurrences: Diogene at trec-2002. In *Proceedings of the Eleventh Text REtrieval Conference (TREC-2002)*.

[Matuszek et al.2012] Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. 2012. Learning to parse natural language commands to a robot control system. In *Proceedings of the 13th International Symposium on Experimental Robotics*.

[Mendes and Coheur2011] Ana Cristina Mendes and Luisa Coheur. 2011. An approach to answer selection in question-answering based on semantic relations. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*.

[Mikolov et al.2013] T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*.

[Moschitti and Quarteroni2010] Alessandro Moschitti and Silvia Quarteroni. 2010. Linguistic kernels for answer re-ranking in question answering systems. In *Information Processing and Management*.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of Empirical Methods in Natural Language Processing*.

[Phillips1960] A. V. Phillips. 1960. A question-answering routine. Technical report, Cambridge, MA, USA.

[Prager et al.2000] John Prager, Eric Brown, and Anni Coden. 2000. Question answering by predictive annotation. In *Proceedings of the SIGIR*.

[Punyakanok et al.2004] Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2004. Natural language inference via dependency tree mapping: An application to question answering. In *Computational Linguistics*.

[Rajpurkar et al.2016] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv:1606.05250*.

[Ravichandran and Hovy2002] Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the Association for Computational Linguistics*.

[Richardson et al.2013] Matthew Richardson, Burges, Christopher J.C., and Renshaw Erin. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of Empirical Methods in Natural Language Processing*.

[Roth and Yih2004] Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of CoNLL*.

[Santos et al.2014] Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2014. Attentive pooling networks. In *arXiv:1602.03609v1*.

[Shen and Klakow2006] Dan Shen and Dietrich Klakow. 2006. Exploring correlation of dependency relation paths for answer extraction. *International Conference on Computational Linguistics and Association for Computational Linguistics*.

[Snow et al.2005] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*.

[Socher et al.2013] 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*.

[Srikumar and Roth2011] Vivek Srikumar and Dan Roth. 2011. A joint model for extended semantic role labeling. In *Proceedings of Empirical Methods in Natural Language Processing*.

[Suchanek et al.2007] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*.

[Sukhbaatar et al.2015] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*.

[Sun et al.2005] Renxu Sun, Jing Jiang, Yee Fan, Tan Hang, Cui Tat-seng, and Chua Min-yen Kan. 2005. Using syntactic and semantic relation analysis in question answering. In *Proceedings of Fourteen Text Retrieval Conference (TREC-2005)*.

[Sun et al.2013] Hong Sun, Nan Duan, Yajuan Duan, and Ming Zhou. 2013. Answer extraction from passage graph for question answering. *International Joint Conference on Artificial Intelligence*.

[Turmo et al.2009] J. Turmo, P. Comas, S. Rosset, O. Galibert, N. Moreau, D. Mostefa, P. Rosso, and D. Buscaldi. 2009. Overview of qast 2009. In *Proceedings of the CLEF 2009 Workshop*.

[Wang and Neumann2007] Rui Wang and Gunter Neumann. 2007. Dfki-lt at ave 2007: Using recognizing textual entailment for answer validation. In *online proceedings of CLEF*.

[Weston et al.2015] Jason Weston, Sumit Chopra, and Bordes Antoine. 2015. Memory networks. In *Proceedings of ICLR*.

[Weston et al.2016] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merrienboer, Armand Joulin, and Tomas Mikolov. 2016. Towards ai complete question answering: A set of prerequisite toy tasks. In *arXiv:1502.05698*.

[Wong and Mooney2007] Y-W. Wong and R. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of Association for Computational Linguistics*.

[Woods1973] William Woods. 1973. Progress in natural language understanding - an application to lunar geology. In *AFIPS Conference Proceedings*, volume 42.

[Wu et al.2005] Min Wu, Mingyuan Duan, Samira Shaikh, Sharon Small, and Tomek Strzalkowski. 2005. Ilqua - an ie-driven question answering system. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC)*.

[Yang et al.1999] Yi Yang, Yih, and Christopher Meek. 1999. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of Empirical Methods in Natural Language Processing*.

[Yao and Durme2014] X. Yao and B. Van Durme. 2014. Information extraction over structured data: Question answering with freebase. *Proceedings of the Association for Computational Linguistics*.

[Yao et al.2013] Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *Proceedings of the Workshop on Human-Computer Question Answering, The North American Chapter of the Association for Computational Linguistics*.

[Yih et al.2014] Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

[Yin et al.2015] Wenpeng Yin, Sebastian Ebert, and Hinrich Schutze. 2015. Attention-based convolutional neural network for machine comprehension. In *arXiv:1602.04341v1*.

[Zelle and Mooney1996] 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*.

[Zettlemoyer and Collins2005] 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)*.