Contents

List of Figures

List of Tables

Chapter 1

Introduction

It has long been a dream to communicate with a computer using natural language, as one might with another human being. We are now closer to this dream, as natural language interfaces become increasingly popular. Our phones are already reasonably good at recognizing speech, and personal assistants, such as Apple Siri, Google Assistant, Microsoft Cortana, Amazon Alexa, *etc.*, help us with everyday tasks and answer some of our questions. Chatbots are arguably considered "the next big thing", and a number of startups developing this kind of technology have emerged in Silicon Valley and around the world[1].

Question answering is one of the major components of such personal assistants. Existing techniques already allow users to get direct answers to some of their questions. However, by some estimates[2], for $\sim 70\%$ of the more complex questions, users still have to dig into the "10 blue links" from search engine results pages, and extract or synthesize answers from information buried within the retrieved documents. In order to make a shift towards more intelligent personal assistants, this gap needs to be closed.

---

[1]http://time.com/4194063/chatbots-facebook-messenger-kik-wechat/

[2]https://www.stonetemple.com/the-growth-of-rich-answers-in-googles-search-results/

The **research goal** of my Ph.D. dissertation is to develop methods and techniques for improving the performance of question answering systems for different information needs, using various types of user generated content, such as text documents, knowledge bases, community question answering archives, and direct human contributions.

Questions come in many varieties, and each type has its own set of challenges [201, 232]. It is common to divide questions into *factoid* and *non-factoid*. Factoid questions are inquiring about certain facts and can be answered with entity names or attributes, such as dates or numbers. An example of a factoid question is "*What book did John Steinbeck wrote about the people in the dust bowl?*" (answer: "*The Grapes of Wrath*"). Of course, there is a variety of questions, that do not fall into this category, *e.g.,* how-to, "why" questions, recommendation and opinion questions, *etc.* The literature usually refers to all such questions as "*non-factoid*". An example of a non-factoid question is "*Why did John Steinbeck name his book the Grapes of Wrath?*". Most of the research in automatic question answering focused on factoid questions [22, 44, 201, 126], but relatively recently more and more works have started targeting the category of non-factoid questions [2, 55, 75, 171, 187, 196, 232].

My approach is to leverage the knowledge contributed by Internet users. More than 3.5 billion people in the world have access to the Internet, and this number has increased tenfold from 1999 to 2013 [3]. Over the years, Internet users have contributed a vast amount of data, and created many highly valuable resources, such as encyclopedias (Wikipedia, WedMD, Britannica), community question answering websites (Quora, Yahoo! Answers, Answers.com), social media (Twitter, Facebook, Snapchat), knowledge bases (Freebase, WikiData, DBpedia, YAGO), and others. With more than a billion websites, the Internet stores a huge volume of information, which could

---

[3]http://www.internetlivestats.com/

	Unstructured data	Structured data
Factoid questions	Text + easy to match against question text + cover a variety of different information types - each text phrase encodes a limited amount of information about mentioned entities	Knowledge Bases + aggregate all the information about entities allow complex queries over this data using special languages (e.g. SPARQL) - hard to translate natural language questions into special query languages - KBs are incomplete (missing entities, facts and properties)
Non-factoid questions	Text + contain relevant information to a big fraction of user needs - hard to extract semantic meaning of text and match it against the question (lexical gap)	Question-Answer pairs + easy to find a relevant answer by matching the corresponding questions - cover a smaller subset of user information needs

Table 1.1: Pros and cons of structured and unstructured data sources for factoid and non-factoid question answering.

be useful to answer user questions.

By their nature, data sources can be classified into *unstructured* (*e.g.,* raw natural language text), *semi-structured* (*e.g.,* tables, question-answer pairs) and *structured* (*e.g.,* knowledge bases). Each of the formats offers a unique set of features, which have their advantages and limitations for question

answering, and can often complement each other (Table 1.1). A number of methods have been proposed for question answering using text collections, knowledge bases (KB) or archives of question-answer (QnA) pairs. Most of the developed systems use either a single source of data or combine multiple independent pipelines, each of which operates over a separate data source.

Two major paradigms for factoid question answering are knowledge base question answering (KBQA) and text-based question answer (TextQA). The information contained in a huge volume of text data can be relatively easily queried using terms and phrases from the original question in order to retrieve sentences that might contain the answer. However, each sentence encodes only limited amount of information about mentioned entities and aggregating it over unstructured data is quite problematic.

On the other hand, modern large scale knowledge bases, such as Freebase [34], DBpedia [15], YAGO [133], WikiData [202], accumulate information about millions of entities into a graph of `[subject, predicate, object]` RDF triples. One of the issues with KBs is that they are inherently incomplete and miss a lot of entities, facts, and predicates [62]. Some of these issues are addressed by relation extraction techniques, which can derive factual knowledge from raw text data [138], web tables [44] or pages [45]. In my dissertation, I develop a method to further extend this set of sources to question-answer pairs, available in abundance on community question answering platforms. This technique allows us to extract additional information, which may not be easily accessible in other sources.

While extremely useful, relation extraction techniques are not perfect and introduce both precision and recall errors. An alternative approach is to use raw text and knowledge base data together for joint reasoning. However, mapping between natural languages phrases and knowledge base concepts is not trivial, and is traditionally done by building huge lexicons [22, 23]. The *Text2KB* model, which I developed in my Ph.D. work, takes a different

approach, and exploits techniques from text-based question answering to improve different stages of the KBQA process at run-time, *i.e.,* by retrieving relevant pieces of information from unstructured and semi-structured data sources, and using them to help KBQA system to generate and score candidate answers.

Unfortunately, not all user questions align well with a knowledge base schema, even with large ones, such as Freebase [69]. The *EviNets* model I developed in my dissertation, is a memory-augmented neural network architecture, which aggregates evidence in support for answer candidates from multiple different data sources, such as RDF triples and text sentences. The proposed approach improves the performance over both Text and KB-based question answering, by better utilization of all available information.

In non-factoid question answering, one of the main challenges is the diversity of question and answer types. Reusing answers from previously posted similar questions, which could be found, for example, in CQA archives, was demonstrated to be quite effective to answer new questions [48, 174]. Unfortunately, it is not always possible to find a similar question, that has already been answered, because many information needs are unique in general or in details. Alternative strategies include ranking text passages extracted from retrieved web documents by estimating semantic similarity between the question and an answer candidate [180]. TREC LiveQA shared task[4], started in 2015 to advance research in complex question answering, asks participants to build a system to answer real user questions in real-time using any information available. In this dissertation, I developed an open source state-of-the-art system, which utilizes both CQA archives and web search data sources, and combines them in a single model.

No matter how good a QA system is, there likely to be cases when it is unable to return a satisfactory response to a user question, *e.g.,* existing data

---

[4]http://trec-liveqa.org

sources might not contain the necessary information, or a system might fail to rank a good answer on top of others. Such failures can be detrimental to the overall user experience with a QA system. One way to mitigate this challenging situation is to put a human in the loop, *e.g.,* let a system consult a group of workers, who can provide some kind of feedback and help return a more satisfactory answer. *EmoryCRQA* extends my automatic non-factoid question answering system with a real-time crowdsourcing module. Crowd-sourcing serves as an additional source of answer candidates and their quality ratings, which are used for re-ranking to return a better final response. This system achieved the top performance on TREC LiveQA 2016 shared task.

The scenarios described above follow a traditional one-way interaction setting, where a user issues a single query and a system needs to come up with a response. However, modern conversational interfaces open many opportunities to enrich this scenario, and transition to information seeking dialogs, where a system may ask additional clarification questions, accept user feedback, *etc.* In Chapter 5 of my dissertation I focus on conversational question answering. I describe a formative user study, conducted to learn how users use dialogue-based information seeking, either with humans or chatbots, to solve informational needs. The results of the user study suggest directions for future research in the area. Next, we turn to two particular types of interactions, which a system can exploit to help users: search hints, designed to assist a user with complex multi-step tasks, and clarification questions, which may be asked to resolve certain ambiguities in user questions. Together, our results and findings about conversational search provide a number of insights and ideas, which can be helpful for future research.

1.1   Contributions

The main contributions of the dissertation are summarized as follows:

- **Relation extraction from community generated question-answer pairs:** The dissertation develops a model for extracting factual knowledge for KB completion from CQA question-answer pairs. This method allows to extract more [subject, predicate, object] triples from available information and helps with knowledge base incompletion problem (Section 3.1).

- **Techniques for augmenting knowledge base question answering with unstructured text data:** The dissertation develops a novel Text2KB model for knowledge base question answering, which utilizes various available unstructured information, such as web search results, CQA archives, and annotated document collections, and improves the performance of a pure KBQA system (Section 3.2).

- **Framework for combining text and knowledge base evidence for factoid question answering:** The dissertation develops EviNets, a memory-augmented neural network architecture for aggregating multiple different pieces of information, as evidence in support of different answer candidates. To show the efficiency of the proposed architecture over a variety of user questions, I developed a new entity-centric factoid QA dataset, derived from the Yahoo! Answers archive (Section 3.3).

- **Non-factoid question answering system:** The dissertation develops a state-of-the-art system for non-factoid question answering, which showed very competitive results on both TREC LiveQA 2015 and 2016 shared tasks. The system combines vertical CQA and general web searches to build candidates from both retrieved question-answer pairs

and web documents, which are then ranked with a trained learning-to-rank model (Section 4.1).

- **Techniques for real-time crowdsourcing for question answering:** The dissertation proposes a method to incorporate crowdsourcing into a real-time question answering system. EmoryCRQA system, which extends our LiveQA approach with the crowdsourcing module to obtain additional and rate existing answer candidates, shows significant improvements over the baseline performance and achieves the top result on TREC LiveQA 2016 task (Section 4.2).

- **Exploration of conversational question answering:** The dissertation investigates user perception and behavior patterns in dialog-based information seeking. We further study search hints and clarification questions, as particular examples of actions available to a QA system in dialog settings, and provide the results and implications of the analysis we conducted on the topics (Chapter 5).

Together, the results presented in this Ph.D. dissertation push research in question answering forward by improving the key components in the QA pipeline with techniques to combine different unstructured, semi-structured and structured data source, and by exploring the possibilities of conversational interfaces for information seeking scenarios.

Chapter 2

Background and Related Work

The field of automatic questions answering has a long history of research and dates back to the days when the first computers appear. By the early 60s, people have already explored multiple different approaches to question answering and a number of text-based and knowledge base QA systems existed at that time [175, 176]. In the 70s and 80s, the development of restricted domain knowledge bases and computational linguistics theories facilitated the development of interactive expert and text comprehension systems [14, 173, 221, 223]. The modern era of question answering research was motivated by a series of Text Retrieval Conference (TREC[1]) question answering shared tasks, which were organized annually from 1999 to 2007 [201]. A comprehensive survey of the approaches from TREC QA 2007 can be found in [58]. To track the progress made during the years of active research in question answering I can refer the readers to a number of surveys, such as [9, 13, 82, 93, 115, 150, 209].

In this Chapter, I describe the works that provide the foundation and give the context to the research of my Ph.D. dissertation.

---

[1]http://trec.nist.gov

2.1 Factoid Question Answering

Most of the research in the earlier days of QA have focused on questions, which can be answered with a name of an entity, or its attributes, which are usually referred to as factoid questions. In the 60s and 70s, researchers explored different sources of information, which can be used for question answering, which lead to the development of the two major approaches to factoid QA: text-based (TextQA) and knowledge base question answering (KBQA) [175]. I first describe related work in TextQA (Section 2.1.1), then introduce KBQA (Section 2.1.2), and finally in Sections 2.1.3 and 2.1.4 present existing techniques for combining different information sources together.

2.1.1 Text-based Question Answering

A traditional approach to factoid question answering over text documents, popularized by the TREC QA task, starts by querying a collection with a possibly transformed question and retrieving a set of potentially relevant documents, which are then used to identify the answer. Information retrieval for question answering has certain differences from traditional IR methods [110], which are usually based on keyword matches. A natural language question contains certain information, that is not expected to be present in the answer (*e.g.,* question words *who*, *what*, *when etc.*), and the answer statement might use language that is different from the question (lexical gap problem) [25]. On the other side, there is a certain additional information about expected answer statement, that a QA system might infer from the question (*e.g.,* we expect to see a number in response to the "how many" question). One way to deal with this problem is to transform the question in certain ways before querying a collection [5, 40]. Another idea is to extend the raw text data with certain semantic annotations, *e.g.,* part of speech tags, semantic role labels,

named entity tags, *etc.* By indexing these annotations a question answering system gets an opportunity to query collection with additional attributes, inferred from the question [33, 52, 238].

The next stage in TextQA is to select sentences, which might contain the answer. One of the most frequently used benchmark datasets for the task, designed by Mengqiu Wang *et al.* [211], is based on the questions from the TREC QA tasks and sentences retrieved by participating systems[2]. The early approaches for the task used mostly keyword matching strategies [101, 181]. However, in many cases, keywords does not capture the similarity in meaning of the sentences very well and researchers started looking on syntactic information. Syntactic and dependency tree edit distances and kernels help us measure the similarity between the structures of the sentences [90, 151, 172, 210, 237]. Recent improvements on the answer sentence selection task are associated with deep learning, *e.g.,* recursive neural networks using sentence dependency tree [102], convolutional neural networks [242, 161], recurrent neural networks [189, 207], and some techniques for term matching in the embeddings space [88, 216, 231]. Another dataset, called WikiQA [233], raises a problem of answer triggering, *i.e.,* detecting cases when the retrieved set of sentences does not contain the answer.

To provide a user with the concise response to her factoid question, QA systems extract the actual answer phrase from retrieved sentences. This problem is often formulated as a sequence labeling problem, which can be solved using structured prediction models, such as CRF [237], or as a node labeling problem in an answer sentence parse tree [134]. A couple of recently developed benchmark datasets, such as WebQA [123], Stanford QUestion

---

[2]A table with all known benchmark results and links to the corresponding papers can be found on

http://aclweb.org/aclwiki/index.php?title=Question_Answering_(State_of_the_art)

Anwsering Dataset SQuAD [154][3], and Microsoft MARCO [142][4], have considerable size ($\sim 100K$ questions), which allows researchers to train and reliably test different models, including various deep learning architectures.

Unfortunately, passages include very limited amount of information about the candidate answer entities, *i.e.,* very often it does not include the information about their types (person, location, organization, or more fine-grained, CEO, president, basketball player, *etc.*), which is very important to answer question correctly, *e.g.,* for the question "*what country did host the 2016 summer olympics?*" we need to know that `Rio de Janeiro` is a city and `Brazil` is a country to be able to respond correctly. Therefore, multiple works have put some effort into developing answer type typologies [95, 96], prediction, and matching of expected and true candidate answer entity types [124, 125, 149]. Many approaches exploited external data, such as large-scale open domain knowledge bases, and I will describe some of these efforts in Section 2.1.4.

When dealing with large document collections, such as the Web, we often have a situation of information duplication, *e.g.,* same knowledge is stated in text dozens, hundreds and sometimes thousands of times, possibly using different language. It is also frequent to have contradictory or false information, presented in some of the documents, intentionally or not. In such circumstances, it is quite useful to aggregate the information across multiple pieces of evidence and use redundancy for the benefit. AskMSR system was one of the first to exploit this idea, and it achieved very impressive results on TREC QA 2001 shared task [41]. The system starts by transforming a question into search queries, extracts snippets of search results from a web search engine, and consider word n-grams as answer candidates, ranking them by frequency. A recent revision of the AskMSR QA system [195] introduced

---

[3]https://rajpurkar.github.io/SQuAD-explorer/
[4]http://www.msmarco.org/

several improvements to the original system, *i.e.,* named entity tagger for candidate extraction, and additional semantic similarity features for answer ranking. It was also observed, that modern search engines are much better in returning the relevant documents for question queries and query generation step is no longer needed. Some other notable systems, that used the web as the source for question answering are MULDER [118], Aranea [127], and a detailed analysis of what affects the performance of the redundancy-based question answering systems can be found in [53, 126].

2.1.2 Knowledge Base Question Answering

Despite the abundance of knowledge available in textual resources, it is often challenging for a computer system to extract and understand this information. Knowledge bases, on the other hand, encode precise factual information, which can be effectively queried and reasoned with, which is quite natural in computer science.

Knowledge Bases and Datasets

In the early days of QA research, knowledge bases were relatively small and contained information specific to a particular domain. Many approaches have been developed to answer detailed questions about these domains, *e.g.,* baseball [81], lunar geology [223], or geography [243]. Recent development of large scale knowledge bases (*e.g.,* DBpedia [15], Freebase [34], YAGO [184], WikiData[5]) shifted attention towards open domain question answering. One of the problems of techniques developed in the earlier days is domain adaptation, as it is quite challenging to map from natural language phrases to database concepts in open domain when the search space is quite large. KBQA approaches can be evaluated on an annual Question Answering over

---

[5]http://www.wikidata.org

Linked Data (QALD[6]) shared task, and some popular benchmark dataset, such as Free917 [46], WebQuestions [22] and WebQuestionsSP [240]. A series of QALD evaluation campaigns has started in 2011, and since then a number of different subtasks have been offered, *i.e.,* since 2013 QALD includes a multilingual task, and QALD-4 formulated a problem of hybrid question answering. These tasks usually use DBpedia knowledge base and provide a training set of questions, annotated with the ground truth SPARQL queries. The hybrid track is of particular interest to the topic of this dissertation, as the main goal in this task is to use both structured RDF triples and free form text available in DBpedia abstracts to answer user questions. A survey of some of the proposed approaches can be found in [197].

Systems Architecture

The architecture of most KBQA systems are based on one of the two major approaches: semantic parsing and information extraction. Semantic parsing starts from question utterances and works to produce the corresponding semantic representations, *e.g.,* logical forms. The model of J.Berant *et al.* [22] uses a CCG parser, which can produce many candidates on each level of parsing tree construction. A common strategy is to use beam search to keep top-k options on each parsing level or agenda-based parsing [24], which maintains current best parses across all levels. An alternative information extraction strategy was proposed by Xuchen Yao *et al.* [236], and it can be very effective for relatively simple questions. The idea of the information extraction approach is that for most of the questions the answer lies in the neighborhood of the question topic entity. Therefore, it is possible to use a relatively small set of query patterns to generate candidate answers, which are then ranked using the information about how well involved predicates and entities match the original question. A comparison of this approaches can be found in [235].

---

[6]www.sc.cit-ec.uni-bielefeld.de/qald/

Question entity identification and disambiguation is the key component in such systems, they cannot answer the question correctly if the question entity is not identified. Multiple systems used NER to tag question entities, which are then linked to a knowledge base using an entity names lexicon [22, 23, 227]. However, NER can easily miss the right span, which would not allow this question to be answered correctly. Most of the recently developed KBQA systems consider a reasonable subset of token n-grams, each of which can map to zero or more KB entities. Top entities according to some entity linking scores are kept and disambiguated only at the answer ranking stage [20, 191, 234]. Ranking of candidates can be done using either a simple linear classification model [234] or more complex algorithms, *e.g.,* gradient boosted trees ranking model [20, 191].

Some questions contain certain conditions, that require special filters or aggregations to be applied to a set of entities. For example, the question *"who won 2011 heisman trophy?"* contains a date, that needs to be used to filter the set of heisman trophy winners, the question *"what high school did president bill clinton attend?"* requires a filter on the entity type to filter high schools from the list of educational institutions, and *"what is the closest airport to naples florida?"* requires a set of airports to be sorted by distance and the closest one to be selected. Information extraction approaches either need to extend the set of candidate query templates used, which is usually done manually, or to attach such aggregations later in the process, after the initial set of entities have been extracted [191, 226]. An alternative strategy to answer complex questions is to extend RDF triples as a unit of knowledge with additional arguments and perform question answering over n-tuples [241]. Z.Wang *et al.* [215] proposed to start from single KB facts and build more complex logical formulas by combining existing ones while scoring candidates using paraphrasing model. Such a template-free model combines the benefits of semantic parsing and information extraction approaches.

Question to Query Mapping

One of the major difficulties in KBQA is the problem of a lexical gap and lexicon construction for mapping natural language phrases to knowledge base concepts [23, 68]. The earlier systems were mainly trained from questions annotated with ground truth logical forms, which are expensive to obtain. Such approaches are hard to scale to large open domain knowledge bases, which contain millions of entities and thousands of different predicates.

An idea to extend a trained parser with an additional lexicon, built from the Web and other resources, has been proposed by Q. Cai and A. Yates [46]. However, most of the parses of a question produce different results, which means that it is possible to use question-answer pairs directly [22], however Scott Wen-tau Yih *et al.* [240] showed that for relatively simple questions, obtaining true semantic parse ground truth might be easier than correct answers, and gives better system performance. PARALEX system of A.Fader *et al.* [68] constructs a lexicon from a collection of question paraphrases from WikiAnswers[7]. A reverse approach was proposed in ParaSempre model of J.Berant *et al.* [23], which ranks candidate structured queries by first constructing a canonical utterance for each query and then uses a paraphrasing model to score it against the original question.

Another approach to learning term-predicate mapping is to use patterns obtained using distant supervision [138] labeling of a large text corpus, such as the ClueWeb [235]. Such labeled collections can also be used to train a KBQA system, as demonstrated by S.Reddy *et al.* [156, 157]. This approach is very attractive as it does not require any manual labeling and can be easily transferred to a new domain. However, learning from statements instead of question-answer pairs has certain disadvantages, *e.g.,* question-answer lexical gap, and noise in distant supervision labeling.

---

[7]https://answers.wikia.com/

Modern knowledge bases also contain certain names or surface forms for their predicates and entities, which makes it possible to convert KB RDF triples into questions and use them for training [35]. Finally, many systems work with distributed vector representations for words and RDF triples and use various deep learning techniques for answer selection. A common strategy is to embed question text and knowledge base concepts into the same space and perform reasoning using operations in this vector space. For example, character n-gram text representation as input to a convolutional neural network can capture the gist of the question and help map phrases to entities and predicates [239]. Joint embeddings can be trained using multi-task learning, *e.g.,* a system can learn to embed a question and candidate answer subgraph using question-answer pairs and question paraphrases at the same time [35].

Memory Networks, developed by the Facebook AI Lab, can also be used to return triples stored in network memory in a response to user questions [36]. This approach uses embeddings of predicates and can answer relatively simple questions, that do not contain any constraints and aggregations. A nice extension of this idea is so called key-value memory networks [136], which simplify retrieval by replacing a single memory cell, which has to be selected using softmax layer, with a key-value pair. Thus, one can encode subject and predicate of a KB triple as the key and let the model return the object as the value of a memory cell. Both regular and Key-value Memory Networks summarize the whole set of memories into a single vector, which is then used to score answer candidates, and this can lead to information losses. This limitation has been partially addressed in [91, 213], which propose to accumulate evidence for each answer separately using a recurrent neural network. To extend deep learning framework to more complex questions, Li Dong *et al.* [61] used a multi-column convolutional neural network to capture the embedding of entity path, context, and type at the same time. Another idea that allows

memory networks to answer complex questions is multiple iterations over the memory, which helps the model to focus on different parts of the question and extend the current set of candidate facts, as shown by S.Jain *et al.* [104].

The described approaches have significantly advanced the state-of-the-art in knowledge base question answering [8]. However, one of the major drawbacks of knowledge bases is their incompleteness, which means that many entities, predicates, and facts are missing from knowledge bases, which limits the number of questions one can answer using them. This brings up a question of combining data from multiple sources, and in the next Section I describe information extraction, which targets the problem of extending knowledge bases with data, extracted from other available data sources.

2.1.3 Information Extraction

One approach to increase the coverage of knowledge bases is to extract information from other resources, such as raw text [83, 107, 138], web tables [44], or infer from existing knowledge [37, 79, 119]. As most of the information in the world is present in unstructured format, relation extraction from natural language text has been an active area of research for many years, and a number of supervised [177], semi-supervised [4] and unsupervised [67] methods have been proposed. These techniques analyze individual sentences and can extract facts stated in them using syntactic patterns, sentence similarity, *etc.*

One of the approaches for information extraction, that has received a considerable attention in the recent years, thanks to the rise of neural network research, is a joint representation of text and knowledge base data. The introduction of text-based edges, extracted from sentences mentioning a pair of entities, to the Path Ranking Algorithm was demonstrated to be supe-

---

[8]A table with some of the results on WebQuestions dataset are available at https://goo.gl/sePBja

rior to KB data alone for knowledge base construction [119]. Such a graph, consisting of KB entities, predicates, and textual data can be viewed as a heterogeneous information network, and such a representation was effectively used to represent text documents for clustering and classification [204, 205]. The idea of universal schemas for relation extraction is to represent KB and natural language predicates with embeddings in low dimensional space. The original work of Sebastian Riedel *et al.* [158] proposed factorizing a matrix, in which rows correspond to entity pairs and columns to KB predicates and natural language phrases connecting these entity mentions in text. These techniques were further improved by learning embeddings of individual entities [200], which allows the model to generalize to unseen entity pairs, and compositionality-aware embeddings of natural language [193] to better capture the variability of the language. Zhen Wang *et al.* [214] showed how to embed entities and words into the same space by preserving entity relations and word co-occurrences in text. These approaches aim at computing a similarity between KB predicates and the ways they are expressed in sentences, and they do not attempt to solve a problem of detecting relations not present in KB, which users might ask about, nor they are trying to cross the sentence boundary and extract information scattered across multiple sentences. However, embedding of various modalities, such as knowledge base predicates and text, into the same space have been effectively used for different tasks, including question answering with so-called memory networks [36, 136].

However, the larger the knowledge base gets, the more difficult it is to find a mapping from natural language phrases to KB concepts. Alternatively, open information extraction techniques [66] can be used to extract a schema-less knowledge base, which can be very effective for question answering. Open question answering approach of Anthony Fader *et al.* [69, 241] combines multiple structured (Freebase) and unstructured (OpenIE) knowledge bases together by converting them to string-based triples. User question

can be first paraphrased using a paraphrasing model learned from WikiAnswers data, then converted to a KB query, where certain query rewrite rules can be applied, and all queries are ranked by a machine learning model.

Abstract Meaning Representation (AMR) [18] is an attempt to build a universal semantic representation schema. The potential of AMR has been demonstrated on many tasks, including reading comprehension [213], however it is not clear how this result can be scaled to the open domain setting.

The work I describe in my dissertation (Section 3.1) builds on the research in relation extraction for knowledge base completion, and extends it to a new domain: question-answer pairs, which helps to increase the amount of information we can extract from available resources.

Relation extraction methods have made a big progress, however, they are not perfect and still leave a lot of useful data behind, and add noise in a form of the erroneously triples. In the next Section, I will describe research on using the raw structured and unstructured data for joint reasoning in question answering.

2.1.4   Hybrid Question Answering

A natural idea of combining available information sources to improve question answering has been explored for a long time. A very detailed overview of these approaches can be found in a recent book by Hannah Bast, Björn Buchhold and Elman Haussmann [19].

Researchers have used various additional resources, such as WordNet [137], Wikipedia[9] and structured knowledge bases along with textual document collections. WordNet lexical database was among the first resources, that were adapted by QA community for such tasks as query expansion and definition extractions [97, 146]. Next, Wikipedia, which can be characterized as an un-

---

[9]http://www.wikipedia.org

structured and semi-structured (infoboxes) knowledge base, quickly became a valuable resource for answer extraction and verification [7, 43]. Developers of the Aranea QA [127] system noticed that structured knowledge bases are very effective in answering a significant portion of relatively simple questions. They designed a set of regular expressions for popular questions that can be efficiently answered from a knowledge base and fall back to regular text-based methods for the rest of the questions. Knowledge bases can also be used as an external source of structured data for some lower level text tasks, such as language modeling [8].

Another great example of a hybrid question answering system is IBM Watson, which is arguably the most important and well-known QA system ever developed so far. It was designed to play the Jeopardy TV show[10]. The system combined multiple different approaches, including text-based, relation extraction and knowledge base modules, each of which generated candidate answers, which were then pooled together for ranking and answer selection. The full architecture of the system is well described in [72] or in the full special issue of the IBM Journal of Research and Development [73]. YodaQA [21] is an open source implementation of the ideas behind the IBM Watson system.

On the other hand, knowledge base question answering systems can benefit from lexical resources. After the information is encoded into RDF triples in a knowledge base, we need to be able to map it back to natural language in order to answer user questions. An idea of extended knowledge graphs [65, 229] is to augment the RDF triples with keywords, which could be extracted from the context of the triple in a text, *e.g.,* from a relation extraction model. These keywords encode a context of a triple and can be used to match against keywords in the question. To query such knowledge graphs authors proposed an extension of the SPARQL language, that allows specifying key-

---

[10]https://en.wikipedia.org/wiki/Jeopardy!

words for some triple patterns. However, such queries now require special answer ranking mechanism, *e.g.,* based on a language model idea [65]. When answering natural language questions, it is often hard to decide whether to map phrases to some KB concepts and which ones. Therefore, many translated queries might become overspecific and return no results at all because of the incorrect translation or lack of knowledge in a KB. Mohamed Yahya *et al.* [229, 228] proposed to use query relaxation techniques to reduce a set of triple patterns in translated SPARQL queries and use some of the question phrases as keywords in the query instead. As an extreme case of such relaxation, we can get a query with a single triple pattern, that retrieves all entities of a certain type and then ranks them using all keywords from the question.

Extension of knowledge bases with textual metadata is subject to some of the above mentioned limitations of knowledge bases. There are several approaches that propose to use data in their original format for QA. K. Xu *et al.* [226] proposed to use textual evidence to do answer filtering in a knowledge base question answering system. On the first stage the system produces a list of answers using traditional information extraction techniques, and then each answer is scored using its Wikipedia page on how well it matches the question. Knowledge bases can also be incorporated inside TextQA systems. Modern KBs contain comprehensive entity type hierarchies, which were utilized in QuASE system of [186] for answer typing. In addition, QuASE exploited the textual descriptions of entities stored in Freebase knowledge base as answer supportive evidence for candidate scoring. However, most of the information in a KB is stored as relations between entities, therefore there is a big potential in using all available KB data to improve question answering.

QALD evaluation campaigns include a hybrid track in a couple of most recent challenges. The goal of this track is to answer questions, that were

designed in such a way, that can only be answered by a combination of a knowledge base and textual data. The targeted textual data is usually descriptions of each entity, stored in DBpedia. These descriptions often represent an overview of the most important information about the entity and can be matched against some parts of the question. The questions designed for this task typically contain multiple parts, one or more of which require textual resources. An example question is: "*Who was vice president under the president who approved the use of atomic weapons against Japan during World War II?*". Due to this specifics and relatively small size of the dataset (QALD-5 training set for multilingual question answering includes 300 examples and 40 examples for the hybrid task), most of the systems are based on certain rules, *e.g.,* splitting the question into parts and issuing individual queries into full-text index or KB [145, 198]. My dissertation focuses on more open settings, where the text does not have to come from inside the knowledge base.

Overall, this dissertation research advances the field of hybrid question answering in two ways. *Text2KB* model, which I describe in Section 3.2, proposes a set of techniques to improve knowledge base question answering using unstructured and semi-structured textual resources, which allows us to bring advances in TextQA over to the KBQA world. This approach relies on KBQA as the primary method, whereas *EviNets* framework (Section 3.3) proposes a neural network architecture, that aggregates information of different nature, as evidence in support for the extracted answer candidates.

2.2 Non-factoid Question Answering

During the earlier days of QA research, non-factoid questions received relatively little attention. The TREC QA tasks started to incorporate certain categories of non-factoid questions, such as definition questions, during the

last 4 years of the challenge. One of the first non-factoid question answering systems was described by R. Soricut and E. Brill [180] and was based on web search using chunks extracted from the original question. The ranking of extracted answer candidates was done using a translation model, which showed better results than the n-gram based match score.

The growth of the popularity of community question answering (CQA) websites, such as Yahoo! Answers, Answers.com, *etc.*, contributed to an increased interest towards non-factoid QA. Some questions on CQA websites are repeated very often and answers can easily be reused to answer new questions, Y.Liu *et al.* [131] studied different types of CQA questions and answers and analyzes them with respect to answer re-usability. A number of methods for similar question retrieval have been proposed [26, 63, 105, 174].

The candidate answer passages ranking problem becomes even more difficult in non-factoid questions answering as systems have to deal with a larger piece of text and need to "understand" what kind of information is expressed there. WebAP is a dataset for non-factoid answer sentence retrieval, which was developed in [232]. Experiments conducted in this work demonstrated, that classical retrieval methods do not work well for this task, and multiple additional semantic (ESA, entity links) and context (adjacent text) features have been proposed to improve the retrieval quality. One of the first extensive studies of different features for non-factoid answer ranking can be found in Mihai Surdeanu *et al.* [187], who explored information retrieval scores, translation models, tree kernel and other features using tokens and semantic annotations (dependency tree, semantic role labeling, *etc.*) of text paragraphs. Alignment between question and answer terms can serve as a good indicator of their semantic similarity. Such an alignment can be produced using a machine learning model with a set of features, representing the quality of the match [217]. Alignment and translation models are usually based on term-term similarities, which are often computed from a monolin-

gual alignment corpus. This data can be very sparse, and to overcome this issue [75] proposed higher-order lexical semantic models, which estimates similarity between terms by considering paths of length more than one on term-term similarity graph. An alternative strategy to overcome the sparsity of monolingual alignment corpora is to use discourse relations of sentences in a text to learn term association models [171]. Some of the more recent works proposed to use neural networks to encode and score the quality of answer passages in non-factoid QA [55, 231].

Questions often have some metadata, such as categories on a community question answering website. This information can be very useful for certain disambiguations and can be encoded in the answer ranking model [244]. Information extraction methods can also be useful for the more general case of non-factoid question answering. For example, there is a huge number of online forums, FAQ-pages and social media, that contain question-answer pairs, which can be extracted to build a collection to query when a new question arrives [56, 60, 106, 122, 230]. The structure of the web page, from which the answers are extracted can be very useful as well. Wikipedia articles have a good structure, and the information encoded there can be extracted in a text-based knowledge base, which can be used for question answering [178].

The TREC LiveQA shared task organized by Yahoo and Emory University started a series of evaluation campaigns for non-factoid question answering. The task is to develop a live question answering system to answer real user questions, that are posted to Yahoo! Answers community question answering website. Most of the approaches from the TREC LiveQA 2015 and 2016 combined similar question retrieval and web search techniques [162, 206, 224]. Answers to similar questions are very effective for answering new questions [48, 162]. However, when a CQA archive does not have any similar questions, we have to fall back to regular web search. The idea behind the LiveQA 2015 winning system of CMU [206] is to represent each answer with a pair of

phrases: clue and answer text. A clue is a phrase that should be similar to the given question, and the passage that follows should be the answer to this question. The overview of the TREC LiveQA 2015 and 2016 shares tasks and their results can be found in the following reports [2, 3].

In my dissertation I develop an open source question answering system (Section 4.1), which retrieves an answer from a combination of vertical and general web searches. The system achieves state-of-the-art results on TREC LiveQA evaluations, and can be used as a baseline for future research in the area.

2.3 Crowdsourcing for Question Answering

Using the wisdom of a crowd to help users satisfy their information needs has been studied before in the literature. M.Bernstein *et al.* [28] explored the use of crowdsourcing for an offline preparation of answers to tail search queries. Log mining techniques were used to identify potential question-answer fragment pairs, which were then processed by the crowd to generate the final answer. This offline procedure allows a search engine to increase the coverage of direct answers to user questions. In contrast, the focus of my dissertation is on online question answering, which requires fast responses to users, who are unlikely to wait more than a minute. Another related work is targeting a different domain, namely SQL queries. The CrowdDB system [74] is an SQL-like processing system for queries, that cannot be answered by machines only. In CrowdDB human input is used to collect missing data, perform computationally difficult functions or matching against the query. In [17] authors explored efficient ways to combine human input for multiple choice questions from the "Who wants to be a millionaire?" TV show. In this scenario going with the majority for complex questions is not effective, and certain answerer confidence weighting schemas can improve the results.

CrowdSearcher platform of [38] proposes to use crowds as a data source in the search process, which connects a searcher with the information available from the users of multiple different social platforms.

Many works have used crowdsourcing to get a valuable information that could guide an automated system for some complex tasks. For example, entity resolution system of [218] asks questions to crowd workers to improve the results accuracy. Using crowdsourcing for relevance judgments has been studied extensively in the information retrieval community, *e.g.,*, [11, 12, 80] to name a few. The focus in these works is on document relevance and the quality of crowdsourced judgments. Whereas in my dissertation I investigate the ability of a crowd to quickly assess the quality of the answers in a nearly real-time setting. The use of crowdsourcing in IR is not limited to relevance judgments. The work of [85] explores crowdsourcing for query formulation task, which could also be used inside an IR-based question answering system. Matthew Lease *et al.* [121] provides a good overview of different applications of crowdsourcing in information retrieval.

Crowdsourcing is usually associated with offline data collection, which requires a significant amount of time. Its application to (near) real-time scenarios poses certain additional challenges. [27] introduced the retainer model for recruiting synchronous crowds for interactive real-time tasks and showed their effectiveness on the best single image and creative generation tasks. VizWiz mobile application of [31] uses a similar strategy to quickly answer visual questions. The works of [98, 99, 120] showed how multiple workers can sit behind a conversational agent named Chorus, where human input is used to propose and vote on responses. Another use of a crowd for maintaining a dialog is presented in [29], who let the crowd handle difficult cases when a system was not able to automatically retrieve a good response from the database of Twitter data.

In my Ph.D. dissertation I show a successful example of integration of a

crowdsourcing module into a real-time QA system (Section 4.2). This work shows that even regular workers without certain domain expertise can provide feedback, which a QA system can use to re-rank and improve its response to the user questions.

2.4 Interactions between Users and QA Systems

Most of the research in question answering have focused on improving the core answer retrieval functionality. However, it is important to look into question answering from a users perspective, which requires analyzing and improving interactions patterns and interfaces. I refer readers to a deep book by Ryen White [219], which focuses on user interactions with search systems. In this section, I describe some of the existing research on improving user experience with certain assistance techniques and studying search in a more natural conversational setting. Several studies have focused on learning more about user satisfaction with personal assistants, *e.g.,* [113, 130, 143].

2.4.1 User Assistance in Information Retrieval

There has been a considerable amount of work on user assistance for general web search and improving user experience with feedback, suggestions, and hints. Results of the study in [225] demonstrate that in 59.5% of the cases users need help to refine their searches or to construct search statements. Individual term [159] or query suggestions [30, 47, 108] are among the most popular techniques for helping users to augment their queries. The study in Diane Kelly *et al.* [111] demonstrated that users prefer query suggestions over term relevance feedback, and that good manually designed suggestions improve retrieval performance. Query suggestion methods usually use search

logs to extract queries that are similar to the query of interest and work better for popular information needs [30]. Query suggestions can also have a learning effect. Harvey *et al.* [86] demonstrated, that users can learn formulate better queries by observing high-quality query suggestions. And search by itself is a learning experience [199].

When query or term suggestions are not available, it is still possible to help users by providing potentially useful search hints. An adaptive tool providing tactical suggestions was presented in [117], and users reported overall satisfaction with its automatic non-intrusive advice. Modern search engines have many features that are not typically used by an average user but can be very useful in particular situations as shown in [140]. The study demonstrated the potential effectiveness and teaching effect of hints. In my dissertation, rather than suggesting to use certain advanced search tools, I explore the effectiveness of *strategic* search hints, designed to suggest a strategy a user can adapt to solve a difficult information question.

2.4.2 Conversational Search and Question Answering

The topic of chatbots and conversational answer seeking has recently become quite popular. F.Radlinski and N.Craswell [153] defined a set of required properties and designed a theoretical model of interactions in a conversational search. M.Iyyer *et al.* [103] released a dataset for conversational question answering, which was built by a converting complex multi-part questions from WikiTables dataset to a sequence of related questions using crowdsourcing. Authors identified major challenges in this data as resolving references to previously mentioned entities and semantic matching.

Much work has been done in the area of comparing user interactions with a human and a computer. There are varying opinions on the subject. Edwards *et al.* [64] found no significant differences in how Twitter users treated a

social bot, whether it was perceived as a human or not. In turn, Clément and Guitton [54] report that the way bots are perceived varies with the role they play. They found that "invasive" Wikipedia bots received more "polarizing" feedback – both positive and negative – compared to the bots that carried out "silent helper" functions. The similar result is reported by Murgia *et al.* [141] – Stackoverflow bot receives more negative feedback for false answers when its identity as an automatic program is revealed. Another work by Aharoni and Fridlund [6] reports mixed results from participants who underwent a mock interview with a human and an automatic system. The authors report that there were no explicit differences in the interviewer perception described by the participants, although the authors noticed significant differences in people's behavior – when talking to a human interviewer they made greater effort to speak, smiled more, and were more affected by a rejection. In a study by Luger and Sellen [132], 14 people were interviewed about their experience with an intelligent assistant that they use in their daily life. The authors report on people's experiences, expectations, discuss scenarios of successes and failures of conversational agents. They report that the most frequent types of tasks are relatively simple – weather updates and checking reminders.

Early QA studies considered users the sole proactive part asking refining questions and clarifying on system's response [59]. QA with a more active system's role was investigated within the complex interactive QA (ciQA) TREC track: assessors provided additional information in various forms to live QA systems as a follow-up to initial inquiry; systems produced updated answers upon interactive sessions [58]. The track outcomes were mixed: interactive phase degraded initial results in most cases; evaluation design was found not quite appropriate for interactive QA.

Kotov and Zhai [116] introduced a concept of *question-guided search*, which can be seen as a variant of query suggestion scenario: in response to an ini-

tial query the user is presented with a list of natural language questions that reflect possible aspects of the information need behind the query. Each such question had a ready to show answer, which a user would see if his intent matched the intent of a suggested question. Tang *et al.* [190] proposed a method for refining questions generation, which consists of two steps: 1) refinement terms are extracted from a set of similar questions retrieved from a question archive; 2) terms are clustered using a WordNet-like thesaurus. Cluster type (such as *location* or *food*) defines the question template to be used. Sajjad *et al.* [160] described a framework for search over a collection of items with textual descriptions exemplified with xbox avatar assets (appearance features, clothes, and other belongings). Multiple textual descriptions for each item were gathered via crowdsourcing; attribute–value pairs were extracted subsequently. In online phase, intermediate search results are analyzed and yes/no questions about attributes and values are generated sequentially in order to bisect the result set and finally come to the sought item. Gangadharaiah and Narayanaswamy [78] elaborated a similar approach to search results refinement through clarification questions. The authors considered customer support scenario using forum data. In offline phase noun phrases, attribute–value pairs and action tuples are extracted from forum collection. In online phase answers to automatically generated questions help reduce the set of candidate answers.

The ability to ask clarification questions is one of the key desired components of conversational search systems [153], and can be used for multiple tasks, *e.g.,* to resolve anaphora and coreferences [152]. In spoken dialog systems, clarification questions can be used to resolve speech recognition uncertainty, either of individual words or of whole utterances [183]. Kato et al. [109] investigated clarification questions in the context of an enterprise Q&A instant messaging in the software domain. Analysis has shown that about one-third of all dialogues have clarification requests; 8.2% of all ut-

terances in the log are related to clarifications. The authors developed a question classifier that prompted the asker to provide clarifications in case the request was identified as underspecified. Pengwei Wang *et al.* [212] used a set of shopping question-answer pairs to extract [subject, predicate, answer, condition] quadruples from a set of shopping question-answer pairs using question pattern mining and clustering techniques. The extractions can then be used to either answer new user questions, or trigger clarifications if the condition entity is missing and needs to be asked about.

The results described in Chapter 5 contributes to the research in conversational search in two aspects. The user study, described in Section 5.1, sheds some lights on what features do people expect a conversational search system to have and what aspects do existing commercial systems lack so far. These results provide the insights and formulate the directions for future research in the area. And one of such aspects, that a conversational system should have, is an ability to ask clarification questions. In Section 5.3 of my dissertation I describe the results of analyzing a dataset from the StackExchange CQA website, and demonstrate feasibility of generating clarification questions automatically.

Chapter 3

Combining Data Sources for Factoid Question Answering

The majority of user web searches are entity-centric, *i.e.,* looking for some information or transacting (*e.g.,* buying, downloading) on entities [148]. Questions that are asking about certain entities (*e.g.,* person, movie, product *etc.*) or their attributes (*e.g.,* date or number) are usually referred to as factoid questions.

Factual information exists in many various formats: natural language statements, tables, question-answer (QnA) pairs, structured databases (DB) and knowledge bases (KB), images and other multimedia resources. These information can be helpful for answering various user questions. Moreover, due to the differences in nature of these data sources, their pros and cons are often complimentary, and therefore their combination would have a synergistic effect. In particular, text document collections [115] and structured knowledge bases [197] are very useful for automatic factoid question answering. Natural language text is relatively easy to match against user questions, especially given the redundancy of the information in large text collections, such as the web [126]. On the contrary, for knowledge bases, translating a question into

one of the structured query languages can be very challenging [22]. Moreover, text encodes all kinds of factual knowledge, whereas knowledge base schemes are often very limited, and a set of objects, predicates and facts are far from being complete [62]. According to D.Wimalasuriya and D.Dou [222], about 80% of the information contained in business documents is stated in natural language, *i.e.,* unstructured format. However, text fragments include a very limited amount of information about the mentioned entities, which complicates the reasoning and often require certain prediction models to be built [124]. Knowledge bases on the other hand aggregate all the information around entities and allow very complicated queries using special languages such as SPARQL. Therefore, an idea to combine unstructured text and structured knowledge bases for joint question answering is very appealing and some existing research demonstrated its potential [21, 65, 68, 72, 186].

This chapter describes multiple different approaches I developed to combine the information available in different unstructured, semi-structured and structured data sources for factoid question answering. First, Section 3.1 details an approach for relation extraction from question-answer pairs for knowledge base completion. Unlike existing techniques, which operate either on individual sentences or on structured data like infoboxes or tables, our method exploits the relationship between question and answer sentences, and increases the volume of extracted information, eventually helping with the KB incompleteness problem. Next, Section 3.2 describes the *Text2KB* model for improving knowledge base question answering using a variety of available unstructured and semi-structured data, such as document search results, question-answer archives, and semantically annotated document collections. Finally, in Section 3.3, I propose *EviNets*: a unified framework for factoid question answering, which jointly processes and aggregates evidence from various structured and unstructured data sources. *EviNets* is a memory-based neural network architecture, which embeds unstructured text

and structured knowledge base triples into a low-dimensional space, which is used to perform relevance matching and aggregation to improve answer selection.

In summary, the contributions of this chapter are:

- A novel approach for information extraction from question-answer pairs, which exploits the known relation between the question and answer sentences to infer the relations between entities, mentioned in them. This work was published in NAACL 2015 Student Research Workshop paper titled "Relation extraction from community generated question-answer pairs" [168].

- *Text2KB*: an approach for using various text data to improve precision and recall of knowledge base question answering. It was presented as SIGIR 2016 full paper "When a Knowledge Base Is Not Enough: Question Answering over Knowledge Bases with External Text Data" [166].

- *EviNets*: Neural network framework for scoring and aggregating evidence from structured and unstructured data sources in support for answer candidates. The paper describing EviNets was presented as an ACL 2017 short paper titled "EviNets: Neural Networks for Combining Evidence Signals for Factoid Question Answering" [167].

3.1   Relation Extraction from Question-Answer Pairs

Knowledge Bases were found to be quite effective in answering some of the users' factual information needs [197]. However, KBs are inherently incomplete, *i.e.,* a lot of information is simply missing even from the largest existing knowledge bases. According to Dong et al [62], 71% of people in Freebase

have no known place of birth, and 75% have no known nationality. One approach to bridge this knowledge gap is automatic knowledge extraction from other data sources, *e.g.,* natural language sentence [4, 83, 107, 138], tables [44], *etc.* In this section, I focus on yet another source of information: question-answer pairs.

CQA websites, such as Yahoo! Answers[1], Answers.com[2], Quora[3] *etc.,* have gained a lot of popularity in the recent years, and their archives store hundreds of millions of user questions along with answers provided by the community. Many users' information needs are not unique and arise again and again, which makes it possible to reuse the information to answer new questions [174]. This idea makes CQA data attractive for knowledge base population. Although some of the facts mentioned in QnA pairs can also be found in some other text documents, another part might be unique (*e.g.,* in Clueweb[4] about 10% of entity pairs with existing Freebase relations mentioned in Yahoo! Answers documents cannot be found in other documents [168]). Existing relation extraction techniques face some challenges when applied to CQA data, *i.e.,* they typically consider sentences independently and ignore the discourse of a QnA pair text. However, frequently, it is impossible to understand the answer without knowing the question. For example, sometimes users simply give the answer to the question without stating it in a narrative sentence (*e.g.,* "*What does "xoxo" stand for? Hugs and kisses.*"), or the provided answer might contain ellipsis, *i.e.,* some important information is omitted (*e.g.,* "*What is the capital city of Bolivia? Sucre is the legal capital, though the government sits in La Paz*").

In my dissertation I propose a novel model for relation extraction from CQA data, that uses the discourse of a QnA pair to extract facts between

---

[1]http://answers.yahoo.com/

[2]http://www.answers.com

[3]http://quora.com

[4]http://www.lemurproject.org/clueweb12/

entities mentioned in the question and answer sentences. The conducted experiments confirm that many of such facts cannot be extracted by existing sentence-based techniques and thus it is beneficial to combine their outputs with the output of our model.

More formally, the target problem is relation extraction from QnA data, which is a collection of (q, a) pairs, where q is a question text (can contain multiple sentences) and a is the corresponding answer text (can also contain multiple sentences). By relation instance, r we mean an ordered binary relation between *subject* and *object* entities, which is commonly represented as $[subject, predicate, object]$ triple. For example, the fact that Brad Pitt married Angelina Jolie can be represented as [Brad Pitt, married_to, Angelina Jolie]. In this work we use Freebase, an open schema-based KB, where all entities and predicates come from the fixed alphabets E and P correspondingly. Let e_1 and e_2 be entities that are mentioned together in a text (*e.g.,* in a sentence, or e_1 in a question and e_2 in the corresponding answer), we will call such an entity pair with the corresponding context a mention. The same pair of entities can be mentioned multiple times within the corpus, and for all mentions $i = 1, ..., n$ the goal is to predict the expressed predicate ($z_i \in P$) or to say that none applies ($z_i = \emptyset$). Individual mention predictions $z_1, ..., z_n$ are combined to infer a set of relations $\mathbf{y} = \{y_i \in P\}$ between the entities e_1 and e_2.

3.1.1 Relation Extraction Models

Models for relation extraction from QnA data incorporates the topic of the question and can be represented as a graphical model (Figure 3.1). Each mention of a pair of entities is represented by a set of mention-based features x_i and question-based features x_q. A multinomial latent variable z_i represents a relation (or none) expressed in the mention and depends on the features
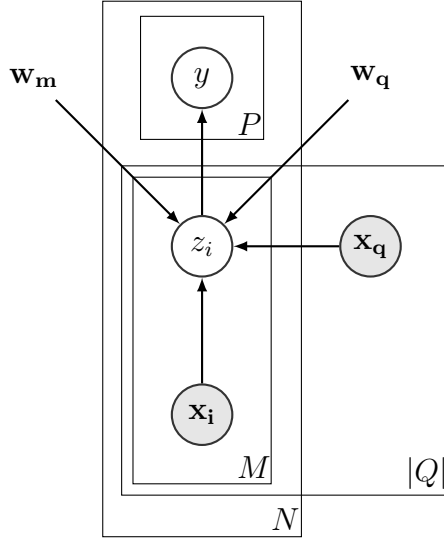
Figure 3.1: QnA-based relation extraction model plate diagram. N - number of different entity pairs, M - number of mentions of an entity pair, $|Q|$ - number of questions where an entity pair is mentioned, $\mathbf{x_i}$ and $\mathbf{x_q}$ - mention-based and question-based features, $\mathbf{w_m}$ and $\mathbf{w_q}$ - corresponding feature weights, latent variables z_i - relation expressed in an entity pair mention, latent variables y - relations between entity pair.

and a set of weights w_m for mention-based and w_q for question-based features:

$$\hat{z}_i = \underset{z \in P \cup \emptyset}{arg\,max}\ p(z_i|x_q, x_i, w_q, w_m)$$

To estimate this variable we use L2-regularized multinomial logistic regression model, trained using the distant supervision approach for relation extraction [138], in which mentions of entity pairs related in Freebase are treated as positive instances for the corresponding predicates, and negative examples are sampled from mentions of entity pairs which are not related by any of the

predicates of interest. Finally, to predict a set of possible relations $\mathbf{y}$ between the pair of entities we take logical OR of individual mention variables z_i, *i.e.,* $y_p = \vee_{i=1}^{M}[z_i = p, p \in P]$, where M is the number of mentions of this pair of entities.

Sentence-based baseline model

Existing sentence-based relation extraction models can be applied to individual sentences of a QnA pair and will work well for complete statements, *e.g.,* "Who did Brad Pitt marry? Brad Pitt and Angelina Jolie married at secret ceremony ...". In sentence-based scenario, when the set of question-based features is empty, the above model corresponds to the Mintz++ baseline described in [188], which was shown to be superior to the original model of [138], is easier to train than some other state of the art distant supervision models and produces comparable results.

Sentence-based model with question features

In many cases, an answer statement is hard to interpret correctly without knowing the corresponding question. To give the baseline model some knowledge about the question, we include question features (Table 3.1), which are based on dependency tree and surface patterns of a question sentence. This information can help the model to account for the question topic and improve predictions in some ambiguous situations.

QnA-based model

The QnA model for relation extraction is inspired by the observation, that often an answer sentence do not mention one of the entities at all, *e.g.,,* "*When was Isaac Newton born? December 25, 1642 Woolsthorpe, England*". To tackle this situation we make the following assumption about the discourse

Sentence-based model	
Dependency path between entities	`[PERSON]→nsubjpass(born)tmod←[DATE]`
Surface pattern	`[PERSON] be/VBD born/VBN [DATE]`
Question features for sentence-based model	
Question template	`when [PERSON] born`
Dependecy path from a verb to the question word	`(when)→advmod(born)`
Question word + dependency tree root	`when+born`
QnA-based model	
Question template + answer entity type	`Q: when [PERSON] born A:[DATE]`
Dependency path from question word to entity	`Q:(when)→advmod(born)nsubj←[PERSON]`
and answer entity to the answer tree root	`A: (born)tmod←[DATE]`
Question word, dependency root and answer pattern	`Q: when+born A:born [DATE]`

Table 3.1: Examples of features used for relation extraction for "*When was Mariah Carey born? Mariah Carey was born 27 March 1970*".

of a QnA pair: an entity mentioned in a question is related to entities in the corresponding answer and the context of both mentions can be used to infer the relation predicate. Our QnA-based relation extraction model takes an entity from a question sentence and entity from the answer as a candidate relation mention, represents it with a set of features (Table 3.1) and predicts a possible relation between them similar to sentence-based models. The features are conjunctions of various dependency tree and surface patterns of a question and answer sentences, designed to capture their topics and relation.

	Y!A	WA
Number of QnA pairs	3.8M	19.6M
Average question length (in chars)	56.67	47.03
Average answer length (in chars)	335.82	24.24
Percent of QnA pairs with answers that do not have any verbs	8.8%	18.9%
Percent of QnA pairs with at least one pair of entities related in Freebase	11.7%	27.5%
Percent of relations between entity pairs in question sentences only	1.6 %	3.1%
Percent of relations between entity pairs in question and answer sentences only	28.1%	46.4%
Percent of relations between entity pairs in answer sentences only	38.6%	12.0%

Table 3.2: Statistics of Yahoo! Answers and WikiAnswers datasets, used in our experiments on relation extraction from question-answer pairs.

3.1.2   Experiments

For experiments we used 2 publicly available CQA datasets: Yahoo! Answers WebScope L6 Comprehensive Questions dataset[5] and a crawl of WikiAnswers[6] collected by A.Fader et al.[69]. The Yahoo! Answers dataset contains 4,483,032 questions (3,894,644 in English) with the corresponding answers collected on 10/25/2007. The crawl of WikiAnswers has 30,370,994 question clusters, tagged by WikiAnswers users as paraphrases, and only 3,386,256 of them have answers. From these clusters, we used all possible pairs of questions and corresponding answers (19,629,443 pairs in total).

For each QnA pair we applied tokenization, sentence detection, named entity tagging, parsing and coreference resolution from Stanford CoreNLP [135]. Our cascade entity linking approach is similar to [51] and considered all noun phrase and named entity mentions as candidates. First, all named entity mentions are looked up in Freebase names and aliases dictionary. The

---

[5]http://webscope.sandbox.yahoo.com/catalog.php?datatype=l
[6]http://wiki.answers.com/

next two stages attempt to match mention text with the dictionary of English Wikipedia concepts [182] and its normalized version. Finally for named entity mentions we try spelling correction using Freebase entity names dictionary. We did not disambiguate entities and instead took top-5 ids for each coreference cluster (using the $p(entity|phrase)$ score from the dictionary or number of existing Freebase triples). All pairs of entities (or entity and date) in a QnA pair that are directly related[7] in Freebase were annotated with the corresponding relations.

Table 3.2 gives some statistics on the datasets used in this work. The analysis of answers that do not have any verbs shows that ~8.8% of all QnA pairs do not state the predicate in the answer text. The percentage is higher for WikiAnswers, which has shorter answers on average. Unfortunately, for many QnA pairs, we were unable to find relations between the mentioned entities (for many of them no or few entities were resolved to Freebase). Among those QnA pairs, where some relation was annotated, we looked at the location of related entities. In Yahoo! Answers dataset 38.6% (12.0% for WikiAnswers) of related entities are mentioned in answer sentences and can potentially be extracted by sentence-based model, and 28.1% (46.4% for WikiAnswers) between entities mentioned in question and answer sentences, which are not available to the baseline model and our goal is to extract some of them.

For our experiments, we use a subset of 29 Freebase predicates that have enough unique instances annotated in our corpus, *e.g.,* date of birth, profession, nationality, education institution, date of death, disease symptoms and treatments, book author, artist album, *etc.* We train and test the models on each dataset separately. Each corpus is randomly split for training (75%) and testing (25%). Knowledge base facts are also split into training and

---

[7]We also consider some paths that come through a mediator node, *e.g.,* `/people/person/spouse_s./people/marriage/spouse`

	Yahoo! Answers			WikiAnswers		
	QnA	Sent.	Comb.	QnA	Sent.	Comb.
F-1 score	0.219	0.276	0.310	0.277	0.297	0.332
Number of correct extractions	3229	5900	7428	2804	2288	3779
Correct triples not extracted by other model	20.5%	56.5%	-	39.4%	25.8%	-

Table 3.3: Results for relation extraction of QnA-, sentence-based and combined models on Yahoo! Answers and WikiAnswers datasets.

testing sets (50% each). QnA and sentence-based models predict labels for each entity pair mention, and we aggregate mention predictions by taking the maximum score for each predicate. We do the same aggregation to produce a combination of QnA- and sentence-based models, *i.e.,*, all extractions produced by the models are combined and if there are multiple extractions of the same fact we take the maximum score as the final confidence. The precision and recall of extractions are evaluated on a test set of Freebase triples, *i.e.,* an extracted triple is considered correct if it belongs to the test set of Freebase triples, which are not used for training (triples used for training are simply ignored). Note, that this only provides a lower bound on the model performance as some of the predicted facts can be correct and simply missing in Freebase.

Figure 3.2 shows Precision-Recall curves for QnA-based and sentence-based baseline models and some numeric results are given in Table 3.3. As 100% recall we took all pairs of entities that can be extracted by either model. It is important to note, that since some entity pairs occur exclusively inside the answer sentences and some in pairs of the question and answer sentences, none of the individual models is capable of achieving 100% recall, and maximum possible recalls for QnA- and sentence-based models are different.
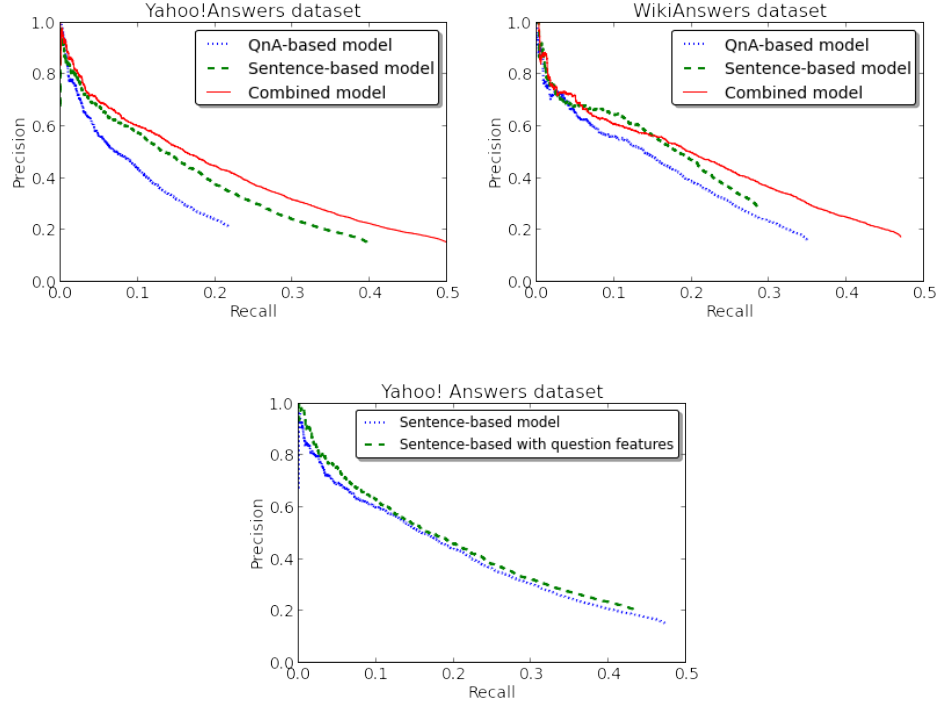
Figure 3.2: Precision-Recall curves for QnA-based vs sentence-based models and sentence-based model with and without question features.

Results demonstrate that from 20.5% to 39.4% of correct triples extracted by the QnA-based model are not extracted by the baseline model, and the combination of both models is able to achieve higher precision and recall. Unfortunately, comparison of the sentence-based model with and without question-based features (Figure 3.2) did not show a significant difference.

3.1.3 Analysis and Discussion

To get an idea of typical problems of the QnA-based model we sampled and manually judged extracted high confidence examples that are not present in Freebase (and thus are considered incorrect for precision-recall analysis).

The major reason (40%) of false positive extractions is errors in entity linking. For example: *"Who is Tim O'Brien? He was born in Austin on October 1, 1946"*. The model was able to correctly extract [Tim O'Brien, date_of_birth, October 1, 1946], however, Tim O'Brien was linked to a wrong person. In a number of cases (16%) our discourse model turns out to be too simple and fails for answers, that mention numerous additional information, *e.g., "How old is Madonna really? ...Cher was born on 20 May 1946 which makes her older that Madonna..."*. A possible solution would be to either restrict QnA-based model to cases when no additional information is present or design a better discourse model with deeper analysis of the answer sentence and its predicates and arguments. Some mistakes are due to distant supervision errors, for example for the music.composition.composer predicate our model extracts singers as well as composers (which are in many cases the same).

Of course, there are a number of cases, when our extractions are indeed correct but are either missing (33%) or contradicting with Freebase (8%). An example of an extracted fact, that is missing in Freebase is *"Who is Wole Soyinka? He studied at the University College, Ibadan(1952-1954) and the University of Leeds (1954-1957)"*, and [Wole Soyinka, institution, University of Leeds] is currently not present in Freebase. Contradictions with Freebase occur because of different precision levels ("pianist" vs "jazz pianist", city vs county, *etc.*), different calendars used for dates or "incorrect" information provided by the user. An example, when existing and extracted relation instance are different in precision is:*"Who is Edward Van Vleck? Edward Van Vleck was a mathematician born in Middletown, Connecticut"* we extract [Edward Van Vleck, place_of_birth, Middletown], however, the Freebase currently has the USA as his place of birth.

The problem of "incorrect" information provided in the answer is very interesting and worth special attention. It has been studied in CQA research,

e.g., [170], and an example of such QnA pair is: "*Who is Chandrababu Naidu? Nara Chandra Babu Naidu (born April 20, 1951)*". Other authoritative resources on the Web give April 20, 1950, as Chandrababu Naidu's date of birth. This raises a question of trust to the provided answer and expertise of the answerer. Many questions on CQA websites belong to the medical domain, *e.g.,* people asking advice on different health related topics. How much can we trust the answers provided to extract them into the knowledge base? We leave this question to the future work.

Finally, we have seen that only a small fraction of available QnA pairs was annotated with existing Freebase relations, which shows a possible limitation of Freebase schema. A promising direction for future work is the automatic extraction of new predicates, which users are interested in and which can be useful to answer more future questions.

3.2 Text2KB: Augmenting Knowledge Base Question Answering with External Text Data

Existing relation extraction tools are not perfect, in particular, due to recall losses a lot of information is left behind. Moreover, extractions contain a certain level of incorrect information due to precision losses. Therefore, by applying relation extraction we are lowering the upper bound of the performance of an underlying question answering system. An alternative approach is to keep the information in its raw unstructured format and design a way to use it along with KB. In this section, I describe a novel factoid question answering system, that utilizes available textual resources to improve different stages of knowledge base question answering (KBQA). This work was presented as a full paper at SIGIR 2016 conference [166].

KBQA systems must address three challenges, namely, question entity identification (to anchor the query process); candidate answer generation; and candidate ranking. We will show that these challenges can be alleviated by the appropriate use of external textual data. Entity identification seeds the answer search process, and therefore the performance of the whole system greatly depends on this stage [234]. The question text is often quite short, may contain typos and other problems, that complicate entity linking. Existing approaches are usually based on dictionaries that contain entity names, aliases and some other phrases, used to refer to the entities [182]. These dictionaries are noisy and incomplete, *e.g.,* to answer the question *"what year did tut became king?"* a system needs to detect a mention *"tut"*, which refers to the entity `Tutankhamun`. If a dictionary does not contain a mapping *"tut"* $\rightarrow$ `Tutankhamun`, as happens for one of the state-of-the-art systems, it will not be able to answer the question correctly. Such less popular name variations are often used along with full names inside text documents, for example, to avoid repetitions. Therefore, we propose to look into web search results to find variations of question entity names, which can be easier to link to a KB (Figure 3.3). This idea has been shown effective in entity linking for web search queries[8] [57].

After question entities have been identified, answer candidates need to be generated and ranked to select the best answer. A candidate query includes one or multiple triple patterns with predicates, corresponding to words and phrases in the question. Existing knowledge base question answering approaches [20, 22, 23, 24, 35, 235] rely on a lexicon, learned from manually labeled training data, and supported by additional resources, such as question paraphrases [23] and weakly labeled sentences from a large text collection [236]. Such training data tends to be small compared to the number of different predicates in a KB, and therefore the coverage of these lexicons is

---

[8]http://web-ngram.research.microsoft.com/ERD2014/
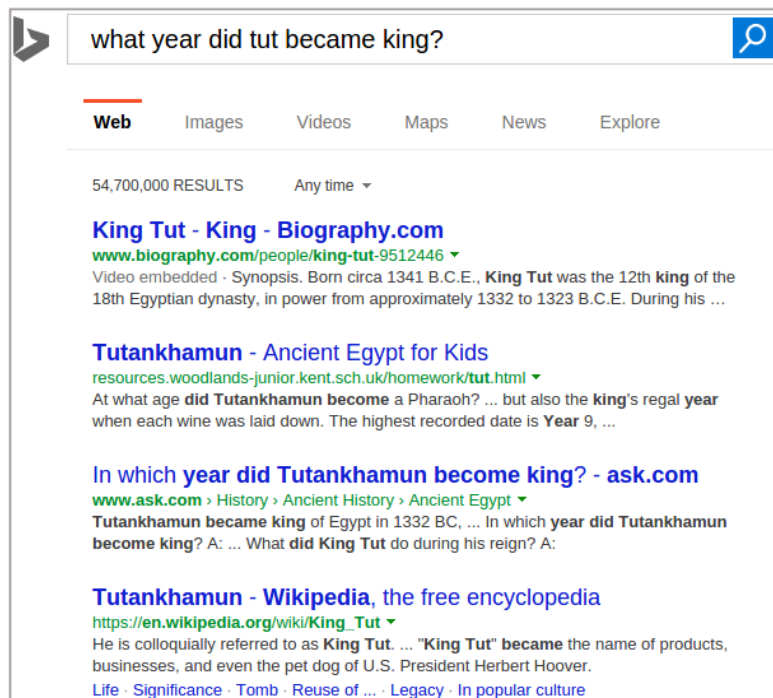
Figure 3.3: Web search results for the question *"what year did tut became king?"*, which mention both the full name of the king and the correct answer to the question.

limited. By our estimate, in a popular WebQuestions KBQA dataset [22], the answers to ∼5.5% of test questions (112 out of 2032) involve a predicate that does not appear as a ground truth in the training set. For example, the RDF triple [`Bigos, food.dish.type_of_dish1, Stew`] answers the question *"what are bigos?"*, but no other examples in the training set involve this predicate. In addition, a lexicon needs to cover all different ways a predicate can be asked about. For example, questions *"who did jon gosselin cheat with?"* and *"who is the woman that john edwards had an affair with?"* are answered by the same KB predicate but use different language. Therefore, the presence of the first question in a training set may not help to answer the

second question. On the other hand, traditional Text-QA systems benefit from the redundancy of the information on the Web, where the same facts are stated multiple times in many different ways [126]. This increases the chances of a good lexical match between a question and answer statements, which makes even some relatively simple counting-based techniques quite effective [41]. The model I developed adapts these ideas from text-based question answering for KBQA.

The general architecture and an example use case of Text2KB is presented on Figure 3.4. Text2KB is based on the information extraction approach to knowledge base question answering [236], in particular, it extends the Aqqu system of H.Bast et al. [20], which is one of the best performing open source KBQA system on the WebQuestions dataset. The left part of the Figure 3.4 describes a typical architecture of IE-based KBQA systems, and the right part introduces additional external text data sources, namely Web search results, community question answering (CQA) data, and a collection of documents with detected KB entity mentions. First, I describe the main stages of the information extraction approach to knowledge base question answering using Aqqu, the baseline system, as an example.

3.2.1 Baseline Approach

The first stage of the knowledge base question answering process is the identification of question topic entities, which are used as sources for the answer search process. For concreteness, consider a question from the WebQuestions dataset *"who is the woman that john edwards had an affair with?"*. Here, the entity `John Edwards` with Freebase id `/m/01651q` is the main question entity. However, Freebase contains millions of entities and it can be difficult to identify the topical ones (*e.g.,* entities `Woman` and `Affair` are also present in Freebase), or to disambiguate and choose between `John Edwards` a politician
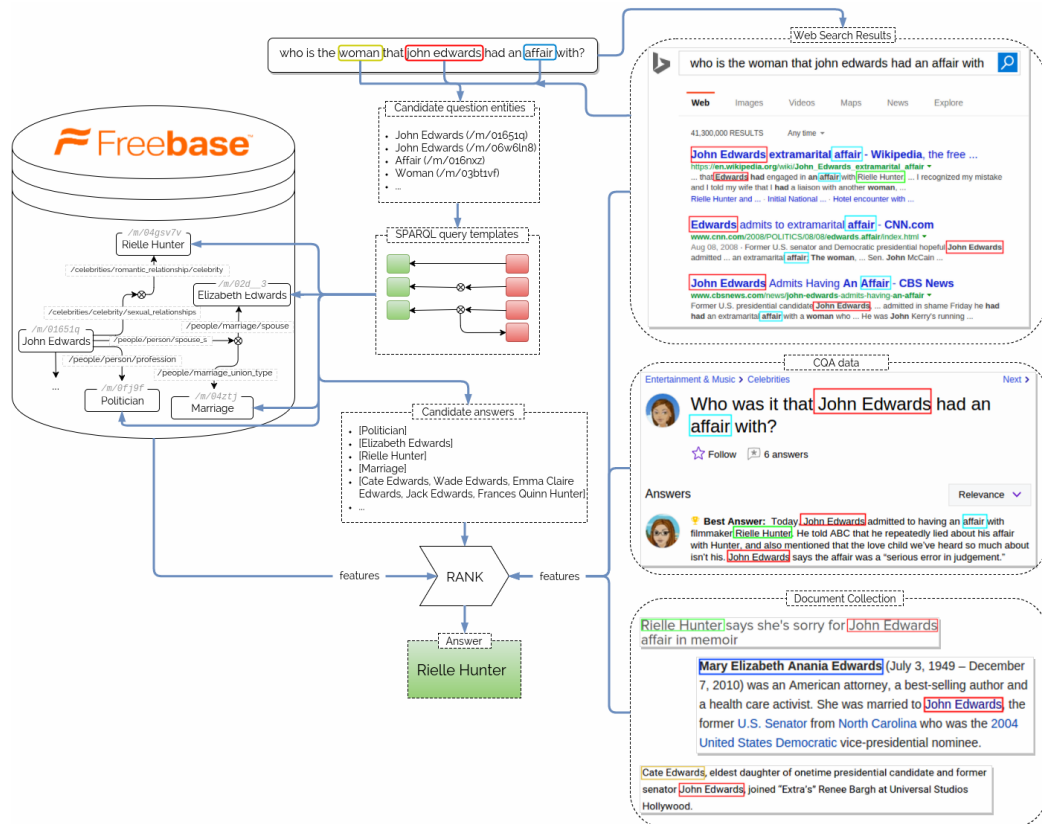
Figure 3.4: The architecture of the Text2KB Question Answering system.

(/m/01641q), an American racing driver (/m/06zs089) and other people with the same name. Aqqu considers all spans of question words under certain conditions on part of speech tags and uses an entity names lexicon [182] to map phrases to potential entities. Most reported systems, including Aqqu, do not disambiguate entities at this stage, but rather keep a set of candidates along with some information about their popularities (*e.g.,* number of mentions in the collection), and mention scores $p(entity|mention\ text)$.

At the next stage, SPARQL query candidates are generated by exploring the neighborhood of the question topic entities using a predefined set of query templates. Each query template has question entities, predicates and answer

placeholders. The majority of the answers in the WebQuestions dataset can be covered by just 3 templates (q_entity - question entity, a_entity - answer entity, cvt_node - Freebase mediator node, which represents tuples with more than 2 arguments):

```
SELECT DISTINCT ?a_entity {
    <q_entity> <predicate> ?a_entity .
}
```

```
SELECT DISTINCT ?a_entity {
    <q_entity> <predicate_1> ?cvt_node .
    ?cvt_node <predicate_2> ?a_entity .
}
```

```
SELECT DISTINCT ?a_entity {
    <q_entity_1> <predicate_1> ?cvt_node .
    ?cvt_node <predicate_2> <q_entity_2> .
    ?cvt_node <predicate_3> ?a_entity .
}
```

The first template retrieves a set of entities that are directly connected to the given question entity via a certain predicate. The second template accounts for the presence of a mediator node, that groups together arguments of a multi-argument relation. And the last template looks for cases when a question also mentions another argument of a multi-argument relation, *e.g.,* Captain Kirk and Star Trek for the question *"who played captain kirk in star trek movie?"*.

Each query candidate is represented with a set of features, that includes the scores for linked question entities, various scores for matching between question term n-grams and query predicates, the size of the results list, *etc.* The final stage of the question answering process is filtering and ranking. The Aqqu system employs a pairwise learning-to-rank model, trained on part of

the dataset. For each pair of candidate answers, Aqqu creates an instance, which contains 3 groups of features: features of the first, the second candidate in the pair and the differences between the corresponding features of the candidates. Specifically, a Random Forest model is used in the provided Aqqu implementation. A pair where the first candidate is better than the second belongs to class +1, and -1 otherwise. To reduce the number of pairs for the final ranking, Aqqu includes a simplified linear filtering model, which is trained to detect incorrect answers with high precision.

In Text2KB we also introduced a couple of extensions to the original Aqqu system, which does not involve external text data. We noticed that since Aqqu does not use information about the answer entity Freebase types, in many cases it returns an answer that is incompatible with the question: *e.g.,* state instead of county *etc.* Therefore, we trained a model to return a score that measures compatibility between the question and answer entities, based on the entity notable types and question uni- and bi-grams as features, similar to Aqqu's relations score model. A second extension introduced a new date range query template, which helps solve cases like *"what team did david beckham play for in 2011?"*, where we need to look at the ranges of dates to determine whether an answer candidate satisfies the question.

```
SELECT DISTINCT ?a_entity {
    <q_entity_1> <predicate_1> ?cvt_node .
    ?cvt_node <from_predicate> ?date_from .
    ?cvt_node <to_predicate> ?date_to .
    ?cvt_node <predicate_2> ?a_entity .
    FILTER ( <question_date> >= ?date_from AND
             <question_date> <= ?date_to )
}
```

3.2.2   Text2KB Model

Text2KB improves the baseline knowledge base question answering model by utilizing textual resources on various stages of the pipeline. Our model uses web search results to improve question analysis, *i.e.,* identify question topic, and extract additional features to support generated answer candidates. Question-answer pairs from CQA archives are used to learn associations between question words and KB predicates, and score candidate SPARQL queries. Finally, a text collection annotated with KB entity mentions is used to build a language model for entity pairs, and generate answer ranking features.

Web search results for question understanding and answer ranking. Traditional TextQA systems rely on search results to retrieve relevant documents, which are then used to extract answers to users' questions. Relevant search results mention question entities multiple times and in various forms, which can be helpful for entity linking [57]. Furthermore, retrieved document set often contains multiple occurrences of the answer, which can be a strong signal for candidate ranking [126].

To obtain related web search results, Text2KB issues the question as a query to a search engine[9], extracts top 10 result snippets and the corresponding documents. Next, Text2KB uses Aqqu entity linking module to detect KB entity mentions in both snippets and documents.

Question text provides only a limited context for entity disambiguation and linking; additionally, the entity name can be misspelled or an uncommon variation used. This complicates the task of entity identification, which is the foundation of KB question answering process. Fortunately, web search results help with these problems, as they usually contain multiple mentions

---

[9]In our experiments, we use the Bing Web Search API https://www.microsoft.com/cognitive-services/en-us/bing-web-search-api and local Wikipedia search using Lucene

of the same entities and provide more context for disambiguation. Text2KB uses the search result snippets to *expand* the set of detected question entities. More specifically, we count the frequencies of each entity mentioned in search snippets, and most popular ones with names similar to some of the question terms are added to the list of topical entities. The goal of this similarity condition is to keep only entities that are likely mentioned in the question text and filter out related, but different entities. To estimate the similarity between a name and question tokens, we use Jaro-Winkler string distance. An entity is added to the list of question entities if at least one of its tokens e_t has high similarity with one of the question tokens q_t excluding stopwords ($Stop$):

$$max_{e_t \in M \backslash Stop, q_t \in Q \backslash Stop} 1 - dist(e_t, q_t) \geq 0.8$$

The information stored in KBs can also be present in other formats, *e.g.,* text statements. For example, on Figure 3.3 multiple search snippets mention the date when Tutankhamun became a king. TextQA systems use such passages to extract answer to users' questions. However, text may not provide sufficient context information about the mentioned entities, and systems have to infer the useful details, *e.g.,* entity types, which can be problematic [239]. On the other hand, KBQA systems can utilize all the available KB knowledge about the entities in a candidate answer and would benefit from additional text-based information to improve ranking. More specifically, Text2KB proceeds as follows (full list of features is given in Table 3.6):

1. Precompute term and entity IDFs. We used Google n-grams corpus to approximate terms IDF by collection frequencies and available ClueWeb Freebase entity annotations[10] to compute entity IDF scores.

2. Each snippet s_i and document d_i are represented by two TF-IDF vectors of lowercased tokens (t_{s_i} and t_{d_i}) and mentioned entities (e_{s_i} and

---

[10]http://lemurproject.org/clueweb09/FACC1/

e_{d_i}).

3. In addition, vectors of all snippets and all documents are merged together to form combined token and entity vectors: $t_{\cup s_i}$, $t_{\cup d_i}$, $e_{\cup s_i}$ and $e_{\cup d_i}$.

4. Each answer candidate a_j is also represented as TF-IDF vector of terms (from entity names), and entities: t_{a_j} and e_{a_j}

5. Cosine similarities between answer and each of 10 snippet and document vectors are computed: $\cos(t_{s_i}, t_{a_j})$, $\cos(t_{d_i}, t_{a_j})$ and $\cos(e_{s_i}, e_{a_j})$, $\cos(e_{d_i}, e_{a_j})$. We use the average score and the maximum score as features.

6. We also compute answer similarities with the combined snippet and document vectors: $\cos(t_{\cup s_i}, t_{a_j})$, $\cos(e_{\cup s_i}, e_{a_j})$, $\cos(t_{\cup d_i}, t_{a_j})$, $\cos(e_{\cup d_i}, e_{a_j})$.

CQA data for matching questions to predicates. Recall that a major challenge in KBQA is that natural language questions do not easily map to entities and predicates in a KB. An established approach for this task is supervised machine learning, which requires labeled examples of questions and the corresponding answers to learn this mapping, which can be expensive to construct. Researchers have proposed to use weakly supervised methods to extend a lexicon with mappings learned from *single sentence statements* mentioning entity pairs in a large corpus [236]. However, the language used in questions to query about a certain predicate may differ from the language used in statements. In Section 3.1 we demonstrated how distant supervision can be applied to question-answer pairs from CQA archives for a related task of information extraction for knowledge base completion. In a similar way, we use weakly labeled collection of question-answer pairs to compute *associations* between question terms and predicates to *extend* system's lexicon (Figure 3.5). We emphasize that this data does not replace the mappings
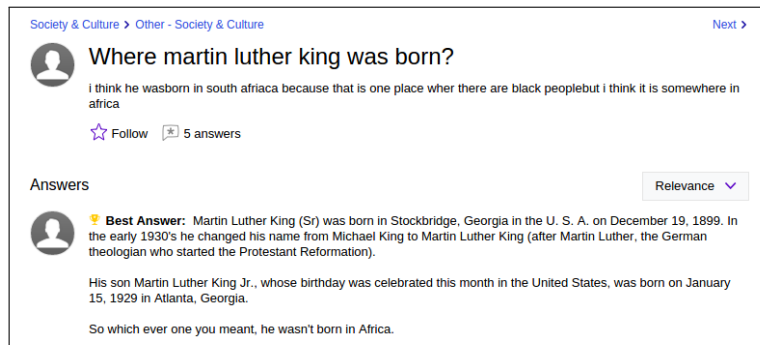
Figure 3.5: Example of a question and answer pair from Yahoo! Answers CQA website.

learned from single sentence statements, which are already used by our baseline system, but rather introduces the new ones learned from the CQA data.

For our experiments, we use 4.4M questions from Yahoo! WebScope L6 dataset[11]. Question and answer texts were run through an entity linker, that detected mentions of Freebase entities. Next, we use distant supervision assumption to label each question-answer pair with predicates between entities mentioned in the question and in the answer. These labels are used to learn associations between question terms and predicates by computing pointwise mutual information scores (PMI) for each term-predicate pair. Examples of scores for some terms are given in Table 3.4.

In Text2KB we evaluate candidate answer predicates by using the association (*e.g.,* PMI) scores between predicates and the question terms (missing pairs are given a score of 0). The minimum, average and maximum of these values are used as features to represent a candidate answer. Such associations data can be sparse, we also use pretrained word2vec word embeddings[12]. We compute predicate embeddings by taking a weighted average of term vectors from predicate's PMI table. Each term vector is weighted by its PMI value

---

[11]https://webscope.sandbox.yahoo.com/

[12]https://code.google.com/p/word2vec/

Term	Predicate	PMI
born	`people.person.date_of_birth`	3.67
	`people.person.date_of_death`	2.73
	`location.location.people_born_here`	1.60
kill	`people.deceased_person.cause_of_death`	1.70
	`book.book.characters`	1.55
currency	`location.country.currency_formerly_used`	5.55
	`location.country.currency_used`	3.54
school	`education.school.school_district`	4.14
	`people.education.institution`	1.70
	`sports.school_sports_team.school`	1.69
illness	`medicine.symptom.symptom_of`	2.11
	`medicine.decease.causes`	1.68
	`medicine.disease.treatments`	1.59
win	`sports.sports_team.championships`	4.11
	`sports.sports_league.championship`	3.79

Table 3.4: Examples of term-predicate pairs with high PMI scores, computed using distant supervision from a CQA collection.

(terms with a negative score are skipped). Then, we compute cosine similarities between predicate vector and each of the question term vectors and take their minimum, average, maximum as features. Finally, we average embeddings of question terms and compute its cosine similarity with the predicate vector (full list of features is given in Table 3.6).

Estimating entity associations using a semantically annotated text collection. A key step for ranking candidate answers is to estimate whether the question and answer entities are related in a way asked in the question. Existing KBQA approaches usually focus on scoring the mappings between

Entity 1	Entity 2	Frequent terms
	Rielle Hunter	campaign, affair, mistress, child, former ...
	Cate Edwards	daughter, former, senator, courthouse, greensboro, eldest ...
John Edwards	Elizabeth Edwards	wife, hunter, campaign, affair, cancer, rielle, husband ...
	Frances Quinn	daughter, john, rielle, father, child, former, paternity...

Table 3.5: Example of entity pairs along with the most popular terms mentioned around the entities in ClueWeb12 collection.

question phrases and KB concepts from a candidate SPARQL query. However, textual data can provide another angle on the problem, as question and answer entities are likely to be mentioned together somewhere in text passages. For example, in the bottom right corner of Figure 3.4 we can see some passages that mention a pair of people, and the context of these mentions explains the nature of the relationships. This data can be viewed as additional edges in a KB, which connect pairs of entities and have associated language models, estimated from text phrases, that mention these entities. Such edges do not have to coincide with the existing KB edges and can connect arbitrary pairs of entities, that are mentioned together in the text, therefore extend the KB.

We use the ClueWeb12 corpus with existing Freebase entity annotations and count different terms that occur in the context of a mention of a pair of different entities (we only consider mentions within 200 characters of each other). To compute this unigram language model we use the terms separating the entities, as well as the terms within a small window (*e.g.,* 100 characters) before and after the entity mentions. A small sample of this data is presented in Table 3.5.

We use this data to compute candidate ranking features as follows. Consider question words Q and an answer candidate, which contains a question entity e_1 and one or more answer entities e_2. For each answer candidate, we compute a language model score:

$$p(Q|e_1, e_2) = \prod_{t \in Q} p(t|e_1, e_2)$$

and use the minimum, average and maximum over all answer entities as features. To address the sparsity problem, we again use embeddings, *i.e.*, for each entity pair a weighted (by counts) average embedding vector of terms is computed and minimum, average and maximum cosine similarities between these vectors and question token embeddings are used as features (full list of features is given in Table 3.6).

Internal text data to enrich entity representation. In addition to external text data, many knowledge bases, including Freebase, contain text data as well, *e.g.,* Freebase includes a description paragraph from Wikipedia for many of its entities. These text fragments provide a general description of entities, which may include information relevant to the question [186]. For completeness, we include them in our system as well. Each entity description is represented by a vector of tokens, and a vector of mentioned entities. We compute cosine similarities between token and entity vectors of the question and description of each of the answers and use the minimum, average and maximum of the scores as features.

The final list of text-based features used in Text2KB model is presented in Table 3.6.

3.2.3   Experimental Results

This section reports the experimental setup, including the dataset and metrics, as well as the main methods compared for evaluating the performance

Data Source	Feature
Search results (`Wiki` or `Web`)	- number of entity mentions in search results
	- max and average tf-idf cosine similarity between answer and search snippets/documents
	- max and average embeddings cosine similarity between question tokens and search snippets/documents
CQA data (`CQA`)	- sum, average, minimum and maximum PMI scores between question tokens and answer predicates
	- sum, average, minimum and maximum embeddings cosine similarity scores between question tokens and PMI-weighted answer predicates tokens
Text collection (`CL`)	- min, max and average entity-pair language model score for question topic and answer entities
	- min, max and average entity-pair embeddings score for question topic and answer entities

Table 3.6: The list of text-based features used in the Text2KB model.

of our Text2KB system. Additionally, we describe a series of ablation studies to analyze the contribution of different system components.

Methods Compared. We compare our system, Text2KB, to state-of-the-art approaches, notably:

- **Aqqu**: a state-of-the-art baseline KBQA system [20], described in Section 3.2.1.
- **Text2KB(Web search)**: Our Text2KB system, using the Bing search engine API over the Web.

- **Text2KB(Wikipedia search)**: Our Text2KB system, using the standard Lucene search engine over the February 2016 snapshot of the English Wikipedia, in order to validate our system without the potential "black-box" effects of relying on a commercial Web search engine (Bing) and changing corpus (Web).
- **STAGG**: One of the best[13] current KBQA systems [191] as measured on the WebQuestions dataset.

Additionally, other previously published results on WebQuestions are included to provide context for the improvements introduced by our Text2KB system.

Datasets. I followed the standard evaluation procedure for the WebQuestions dataset and used the original 70-30% train-test split (3,778 training and 2,032 test instances). Within the training split, 10% was set aside for validation to tune the model parameters and only the best-performing set of parameters selected on the validation data was used to report the results on the official test split.

Evaluation Metrics. Recent papers using the WebQuestions dataset have primarily used the average F1-score as the main evaluation metric, defined as: $avg\ F1 = \frac{1}{|Q|} \sum_{q \in Q} f1(a_q^*, a_q)$

$$f1(a_q^*, a_q) = 2\frac{precision(a_q^*, a_q)recall(a_q^*, a_q)}{precision(a_q^*, a_q) + recall(a_q^*, a_q)}$$

$precision(a_q^*, a_q) = \frac{|a_q^* \cap a_q|}{|a_q|}$ and $recall(a_q^*, a_q) = \frac{|a_q^* \cap a_q|}{|a_q^*|}$, a_q^* and a_q are correct and given answers to the question q, which can be lists of entities. Additionally, I report average precision and recall, to gain a better understanding of the trade-offs achieved by different methods.

---

[13]It was the best result published before summer 2016, *i.e.,* the camera-ready version of my paper, describing Text2KB system.

System	$\overline{Recall}$	$\overline{Precision}$	F1 of $\overline{P}$ & $\overline{R}$	$\overline{F1}$
OpenQA [69]	-	-	-	0.35
YodaQA [21]	-	-	-	0.343
Jacana [236]	0.458	0.517	0.486	0.330
SemPre [22]	0.413	0.480	0.444	0.357
Subgraph Embed [35]	-	-	0.432	0.392
ParaSemPre [23]	0.466	0.405	0.433	0.399
Kitt AI [234]	0.545	0.526	0.535	0.443
AgendaIL [24]	0.557	0.505	0.530	0.497
DepLambda [157]	0.611	0.490	0.544	0.503
STAGG [239]	0.607	0.528	0.565	0.525
Textual Evidence[226]	-	-	-	0.533
FMN [104]	**0.649**	**0.552**	**0.597**	**0.557**
Aqqu (baseline) [20]	0.604	0.498	0.546	0.494
Text2KB$_{WikiSearch}$	$\mathbf{0.632}^{*}_{(+4.6\%)}$	0.498	$0.557^{*}_{(+2.0\%)}$	$0.514^{*}_{(+4.0\%)}$
Text2KB$_{WebSearch}$	$\mathbf{0.635}^{*}_{(+5.1\%)}$	$0.506^{*}_{(+1.6\%)}$	$\mathbf{0.563}^{*}_{(+3.1\%)}$	$\mathbf{0.522}^{*}_{(+5.7\%)}$

Table 3.7: Average performance metrics of the Text2KB system on WebQuestions dataset compared to the existing approaches. The differences of scores marked * from the baseline Aqqu system are significant with p-value < 0.01.

Main Results. The results of existing approaches and our Text2KB system are presented in Table 3.7. We should note, that text-based QA systems typically return a ranked list of answers, whereas many answers on WebQuestions dataset are lists, which complicates the comparison between KBQA and text-based systems. The result reported for YodaQA system is the F1 score at position 1. As we can see, Text2KB significantly improves over the baseline system.

Data Source and Features Contribution. To analyze the contribution

of the features and data sources I introduced, I report results from a series of ablation studies. For convenience, I introduce the following short-hand notations for different components of our system:

- `T` - notable type score model as a ranking feature
- `DF` - date range filter-based query template
- `WebEnt` - using web search result snippets for question entity identification
- `WikiEnt` - using wikipedia search result snippets for question entity identification
- `Web` - using web search results for feature generation
- `Wiki` - using wikipedia search results for feature generation
- `CQA` - using CQA-based [question term, KB predicate] PMI scores for feature generation
- `CW` - features, computed from entity pairs language model, estimated on ClueWeb

In the results table I will use the notation +<comp> for a system with a certain component added, and -<comp> when it is removed. For example, the baseline system will be denoted as "`Aqqu`". The same system with additional date range filter query templates and notable types score model is denoted as "`Aqqu +DF+T`", which represents the same system as "`Text2KB -WebEnt-Web-CQA-CL`" (we will call it Text2KB (base)). Our full system "`Text2KB`" can be also denoted as "`Aqqu +DF+T+WebEnt+Web+CQA+CL`".

First, I analyze the improvements introduced by different components of the system (Table 3.8). As we can see, additional date range filters and notable types model (`Aqqu+DF+T`) are responsible for an increased recall and a drop in precision compared to the baseline model. Features generated from Wikipedia search results, CQA data and ClueWeb entity pair language models (`+Wiki+CQA+CL`) improve average F1 by 0.007 (+1.4%) compared to
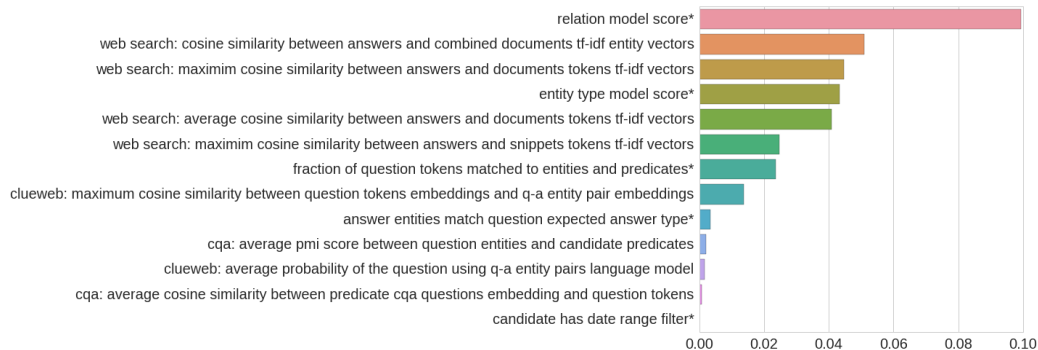
Figure 3.6: A plot of Gini importances of different features of our answer ranking random forest model (features marked * are not text-based and are provided for comparison).

the base model, adding entity linking using Wikipedia search results improves results even more (+3%).

Web search results (`+Web+CQA+CL`) turned out to be more helpful than Wikipedia results (`+Wiki+CQA+CL`), which is natural since Wikipedia is a subset of the web. This was one of the reasons we did not combine Wikipedia and Web search together. Finally, entity linking and all text-based features combined achieves an even higher score, proving that their contributions are independent.

I now analyze the contribution of the different data sources. I will remove a group of web search, CQA or Clueweb-based features and see how the performance of the whole system changes (Table 3.9). As we can see, all data sources have an impact on the system performance, and web search results based features provide the most useful signal for answer ranking.

Figure 3.6 plots a subset of features ranked by their Gini index-based importance scores. The figure supports the observation that web search results features are the most useful, however, other text data sources also contribute to the improvement.

System	Recall	Prec	F1
Aqqu	0.604	0.498	0.494
Text2KB (base) = Aqqu+DF+T	0.617	0.481	0.499
+Wiki+CQA+CL	0.623	0.487	0.506
+WikiEnt +Wiki+CQA+CL	0.632	0.498	0.514
+WebEnt	0.627	0.492	0.508
+Web+CQA+CL	0.634	0.497	0.514
+WebEnt +Web+CQA+CL	0.635	0.506	0.522

Table 3.8: Average Recall, Precision (Prec), and F1 of Aqqu and Text2KB system with and without different components. +A means that a component A is added to the Text2KB (base) system.

In summary, Text2KB significantly outperforms the baseline system, and each of the introduced components contributes to this improvement. Web search results data turned out to be the most useful resource, and it significantly improves the quality by helping with question entity identification and candidate ranking. Next, I analyze the system performance in more detail and investigate factors for future extension.

3.2.4   Analysis

I now investigate how Text2KB compares to other systems on the same benchmark; then, I investigate in depth the different error modes, which helps identify the areas of most substantial future improvements.

I took an existing KBQA system and demonstrated that by combining evidence from a knowledge base and external text resources we can boost the performance. A reasonable question is whether the same approach will be helpful for other systems, *e.g.,* the best system at the moment of our paper publication – STAGG [191]. STAGG differs from the baseline system Aqqu

System	Recall	Prec	F1
`Text2KB` (Web search)	0.635	0.506	0.522
`Text2KB -Web`	0.633	0.496	0.513
`Text2KB -CQA`	0.642	0.499	0.519
`Text2KB -CL`	0.644	0.505	0.523
`Text2KB -CQA-CL`	0.642	0.503	0.522
`Text2KB -Web-CQA`	0.631	0.498	0.514
`Text2KB -Web-CL`	0.622	0.493	0.508

Table 3.9: Average Recall, Precision (Prec), and F1 of Text2KB with and without features based on web search results, CQA data and ClueWeb collection.

in the components: entity linking algorithm, a set of query templates and ranking methods. Therefore, my approach is "orthogonal" to these improvements and should be helpful for STAGG as well. To support this claim, I made an experiment to combine answers of STAGG and Text2KB. One of the advantages of the former is its set of filters, that restricts list results to entities of certain type, gender, *etc.* Therefore, I combined answers of STAGG and Text2KB using a simple heuristic: I chose to use the answer returned by STAGG if the number of answer entities is less than in the Text2KB answer, otherwise, I used the answer of Text2KB. Table 3.10 gives the results of the experiment, and as we can see the combination achieves a slightly better average F1 score. Alternatively, we can look at the Oracle combination of the systems, which always selects the answer with the higher F1. As we can see such a combination results in a performance of 0.606, which is much higher than either of the systems.

As I mentioned earlier, answers to 112 of the test questions in the WebQuestions dataset involve predicates that were not observed in the training

System	avg F1
Text2KB	0.522
STAGG [191]	0.525
Text2KB + STAGG	0.532 (+1.3 %)
Text2KB + STAGG (Oracle)	0.606 (+15.4 %)

Table 3.10: Average F1 for combinations of Text2KB and STAGG using a simple heuristic based on the length of the answer list and Oracle upper bound.
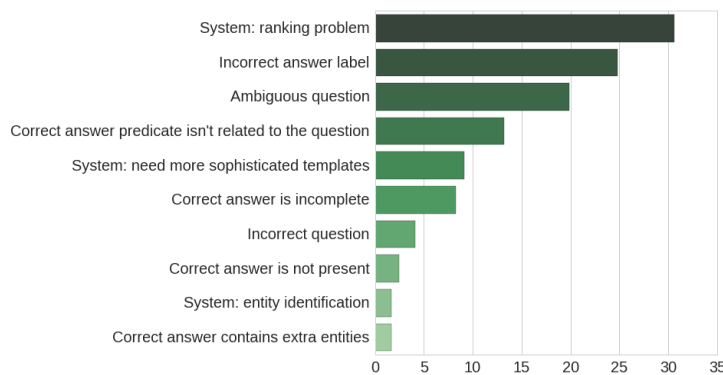
Figure 3.7: Distribution of problems with questions, where Text2KB returns an answer with F1<1.

set, which may be a problem for approaches that rely on a trained lexicon. I evaluated both systems on these questions, and indeed the performance is very low, *i.e.,* the average F1 score of Text2KB is 0.1640 compared to 0.1199 for STAGG[14].

To get better insights into the problems that remain, I collected 1219 questions for which Text2KB did not return a completely correct answer, *i.e.,* F1 score < 1. I manually looked through a couple of hundreds of these examples

---

[14]Unfortunately, the number of questions is too low to show statistical significance (p-value=0.16) of the difference.

and grouped the problems into several clusters (Figure 3.7).

As we can see candidate ranking is still the major problem, and it accounts for $\sim 31\%$ of the cases. The second problem is incorrect ground truth labels (almost 25% of reported errors). Another set of questions has incomplete or overcomplete ground truth answer list. Typical examples are questions asking for a list of movies, books, landmarks, *etc.* The ground truth answer usually contains ~ 10 entities, whereas the full list is often much larger. This seems to be an artifact of the labeling process, where the answer was selected from the Freebase entity profile page, which shows only a sample of 10 entities, while the rest are hidden behind the "N values total" link. About 20% of the questions are ambiguous, *i.e.,* questions have no strict 1-1 correspondence with any of the predicates and can be answered by multiple ones without any obvious preferences. For example, the question *"what did hayes do?"* can be answered by profession, occupied position or some other achievements. Another problem is when there is no predicate that answers the question. For example, the question *"what do people in france like to do for fun?"* does not have a good match among the facts stored in Freebase. The ground truth entity `Cycling` comes from the list Olympic sport competitions country participated[15].

Text2KB components were quite effective in resolving some of the problems. Web search results helped identify the right question topical entity in a number of cases, *e.g., "what did romo do?"* mentions only the last name of the Dallas Cowboys quarterback and the baseline system were unable to map it to the right entity. Web search results provides more than enough evidence that *"romo"* refers to `Tony Romo`. However, there is a number of loses, introduced by added unrelated entities. For example, the entity `I Love Lucy` was added for the question *"what was lucille ball?"*, because the term *lucy* had high similarity with *lucille*. A portion of these problems can be fixed by a bet-

---

[15]`olympics.olympic_participating_country.athletes`

ter entity linking strategy, *e.g.,* [57]. An interesting example, when external text resources improved the performance is the question *"what ship did darwin sail around the world?"*. This is actually a hard question because the ship entity is connected to the `Charles Darwin` entity through the "knownFor" predicate along with some other entities like `Natural selection`. Thus, the predicate itself is not related to the question, but nevertheless, the name of the ship `HMS Beagle` is mentioned multiple times in the web search results, and entity pair model computed from ClueWeb also has high scores for the terms "ship" and "world".

There are several major reasons for the loses, introduced by features based on external text resources. Some entities often mentioned together and therefore one of them gets high values of co-occurrence features. For example, the baseline system answered the question *"when did tony romo got drafted?"* correctly, but since `Tony Romo` is often followed by `Dallas Cowboys`, Text2KB ranked the team name higher. Another common problem with our features is an artifact of entity linking, which works better for names and often skips abstract entities, like professions. For example, the correct answer to the question *"what did jesse owens won?"* is an entity with the name `Associated Press Male Athlete of the Year`, which is rarely mentioned or it is hard to find such mentions. Some problems were introduced by a combination of components. For example, for *"where buddha come from?"* a topical entity `Buddhism` was introduced from search results, and it generated `Gautama Buddha` as one of the answer candidates. This answer was ranked the highest due to a large number of mentions in the search results.

In summary, in this section, I demonstrated that unstructured text resources can be effectively utilized for knowledge base question answering to improve query understanding, candidate answer generation and ranking. Textual resources can help KBQA system mitigate the problems of matching

between knowledge base entities and predicates and textual representation of the question.

Unfortunately, Text2KB does not help with the problem of knowledge base incompleteness, *i.e.,* my system will not be able to respond to the question, which refers to an entity, a predicate or a fact, which is missing in a KB. Section 3.3 describes a neural network framework, that naturally combines evidence of different nature for factoid question answering.

3.3 EviNets: Joint Model for Text and Knowledge Base Question Answering

A critical task for question answering is the final answer selection stage, which has to combine multiple signals available about each answer candidate. Most of the recent works in QA have focused on the problem of semantic matching between a question and candidate answer sentences [88, 155, 231]. The datasets used in these works, such as Answer Sentence Selection Dataset [211] and WikiQA [233], typically contain a relatively small set of sentences, and the task is to select those that state the answer to the question. However, for many questions, a single sentence does not provide sufficient information, and it may not be reliable in isolation. At the same time, the redundancy of information in large corpora, such as the Web, has been shown useful to improve information retrieval approaches to QA [53].

One approach for joint representation of diverse information is embedding into a low-dimensional space, *i.e.,* as achieved by various neural network architectures. In particular, Memory Networks [185] and their extensions [136] use embeddings to represent relevant data as memories, and summarize them into a single vector, therefore losing information about answers provenances.

In this section, I describe *EviNets*, a novel neural network architecture for
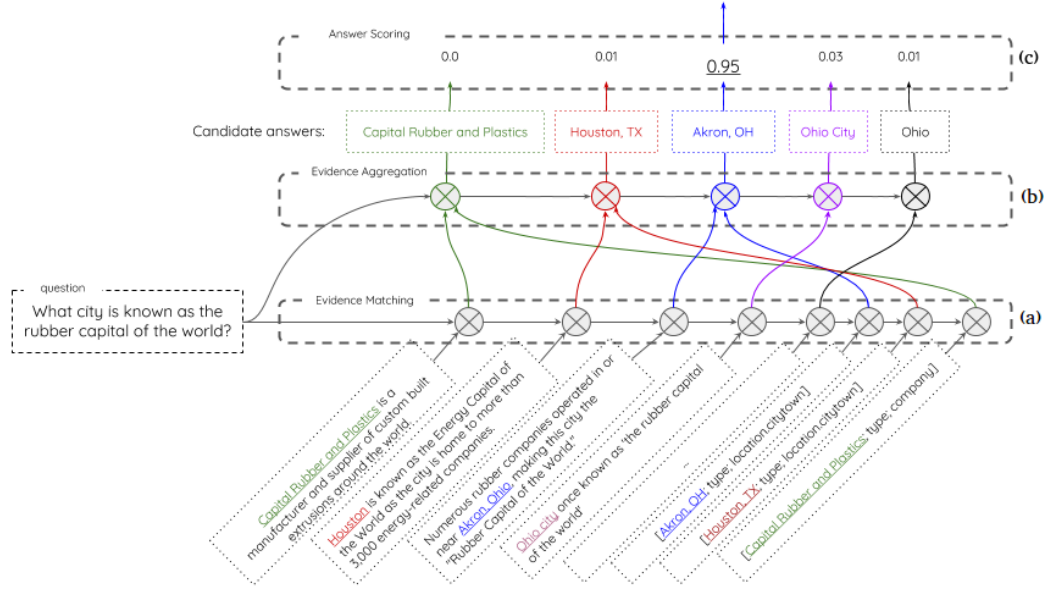
Figure 3.8: The architecture of EviNets Neural Networks for combining textual and KB evidence in factoid question answering.

factoid question answering, which provides a unified framework for aggregating evidence, supporting answer candidates. Given a question, *EviNets* retrieves a set of relevant pieces of information, *e.g.,* sentences from a corpora or knowledge base triples, and extracts mentioned entities as candidate answers. All the evidence signals are then embedded into the same vector space, scored and aggregated using multiple strategies for each answer candidate. Experiments on the TREC QA, WikiMovies and new Yahoo! Answers datasets demonstrate the effectiveness of the proposed approach, and its ability to handle both unstructured text and structured KB triples as evidence. This work was published as a short paper at the Annual Meeting of the Association for Computational Linguistics 2017 [167].

answer score

fully-connected network

aggregated evidence

(c) Answer scoring

(b) Evidence aggregation

$\max_e\{w_e|mention(a,e)\}$

$avg_e\{w_e|mention(a,e)\}$

$(\frac{1}{N}\sum_e w_e|\sum_{a \cdot a} a_{emb} | mention(e,a)|) \cdot a_{emb}$

answer

Akron, OH

mention(a, e)

1	1	1	1
0	1	0	1
0	0	1	0
0	0	0	1

a_{emb} ◯◯◯◯◯◯◯◯

softmax $w_e = \frac{exp(match(q,e))}{\sum_{e'} exp(match(q,e'))}$

$match(q, e^{(i)}) = q_{emb}^T \cdot e_{emb}^{(i)}$

$match(q, e^{(i)}) = q_{emb}^T \cdot e_{emb}^{(i)}$

(a) Evidence matching

q_{emb} ◯◯◯◯◯◯◯◯

BoW averaging

$e_{emb}^{(i)}$ ◯◯◯◯◯◯◯◯

BoW averaging

Other evidence

$q_{emb,t}$

$e_{emb,t}^{(i)}$

question

What city is known as the rubber capital of the world?

evidence

Numerous rubber companies operated in or near Akron, Ohio, making this city the "Rubber Capital of the World."

Figure 3.9: Layer-wise structure of the EviNets Neural Networks framework for factoid question answering. Evidence matching (a), aggregation (b) and answer scoring (c) stages correspond to those in Figure 3.8.

3.3.1 Model and Architecture

The high level architecture of *EviNets* is illustrated in Figure 3.8. For a given question, we extract potentially relevant information, *e.g.,* sentences from documents retrieved from text corpora using a search system. Next, we can use an entity linking system, such as TagMe [70], to identify entities mentioned in the extracted information, which become candidate answers. *EviNets* can further incorporate additional supporting evidence, *e.g.,* textual description of candidate answer entities, and potentially useful KB triples, such as types [186]. Finally, question, answer candidates and supporting evidence are given as input to the *EviNets* neural network.

Let us denote a question by q, and $\{q_t \in R^{|V|}\}$, as a one-hot encoding of its tokens from a fixed vocabulary V. a_i is a candidate answer from the set A, and we will assume, that each answer is represented as a single entity. For each question, we have a fixed set $E = E_{text} \cup E_{KB}$ of evidence statements $e^{(i)}, i = 1..M$, and their tokens $e_t^{(i)}$. A boolean function $mention : A \times E \to \{0, 1\}$ provides the information about which answer candidates are mentioned in which evidences. Individual tokens $q_t, a_i, e_t^{(i)}$ are translated into the embedding space using a matrix $W_{D \times |V|}$, where D is the dimension of the embeddings, *i.e.,* $q_{emb,t} = Wq_t$, $a_{emb,i} = Wa_t$ and $e_{emb,t}^{(i)} = We_t^{(i)}$. In the experiments, I use the same matrix for questions, evidence, and answers. KB entities are considered to be individual tokens, while predicates and type names are tokenized into constituent words, *i.e.,* by splitting on underscore and dot characters for Freebase predicates. A layer-wise architecture of *EviNets* is displayed on Figure 3.9. The evidence matching module ((a) on Figures 3.8 and 3.9) estimates the relevance of each statement in the memory, and computes their weights using the softmax function. The evidence aggregation module (b) uses multiple ways to compute the aggregated statistics of evidences, mentioning each answer candidate. Finally, the answer scoring module (c) uses a fully connected network to predict a score for each answer candidate. EviNets selects the answer with the highest score as the final response.

Evidence Matching Module

Evidence matching is responsible for estimating the relevance of each of the pieces of evidence to the question, i.e., $w_e = softmax(match(q, e))$. The function *match(q, e)* can be implemented using any of the recently proposed semantic similarity estimation architectures[16]. One of the simplest approaches is to average question and each evidence token embed-

---

[16]e.g., see the ACL Wiki on Question_Answering_(State_of_the_art).

dings and score the similarity using the dot product: $q_{emb} = \frac{1}{L_q}\sum_t q_{emb,t}$ and $e_{emb}^{(i)} = \frac{1}{L_e}\sum_t e_{emb,t}^{(i)}$ and $match(q, e^{(i)}) = q_{emb}^T \cdot e_{emb}^{(i)}$.

Evidence Aggregation Module

Evidence Feature	Description
Maximum evidence score mentioning the answer	$\max_e\{w_e|mention(a,e)\}, e \in E, E_{text}$ or E_{KB}
Average evidence score mentioning the answer	$avg_e\{w_e|mention(a,e)\}, e \in E, E_{text}$ or E_{KB}
Sum of evidence scores mentioning the answer	$\sum_e\{w_e|mention(a,e)\}, e \in E, E_{text}$ or E_{KB}
Number of mentions	$\sum_e\{1|mention(a,e)\}, e \in E_{text}$
Weighted memory similarity to the question	$(\frac{1}{M}\sum_i w_e e_{emb}^{(i)}) \cdot q_{emb}$
Weighted memory similarity to the answer [185]	$(\frac{1}{M}\sum_i w_e e_{emb}^{(i)}) \cdot a_{emb}$ or $R^T(\frac{1}{M}\sum_i w_e e_{emb}^{(i)} + q_{emb}) \cdot a_{emb}$, where $R_{D\times D}$ is a rotation matrix
Weighted memory answer mentions similarity to the answer [136]	$(\frac{1}{M}\sum_e w_e[\sum_a a_{emb} \mid mention(e,a)]) \cdot a_{emb}$

Table 3.11: Signals used in *EviNets* to aggregate evidence in support for each of the answer candidates a.

After all the evidence signals have been scored, *EviNets* aggregate the support for each answer candidate. Table 3.11 summarizes the aggregation features used. With these features, *EviNets* captures different aspects, *i.e.,* how well individual sentences match the question, how frequently the candidate is mentioned and how well a set of answer evidences covers the information requested in the question.

Dataset	Example Questions
TREC QA	Where is the highest point in Japan?
1236 train	What is the coldest place on earth?
202 test	Who was the first U.S. president to appear on TV?
WikiMovies	what films did Ira Sachs write?
96185 train	what films does Claude Akins appear in?
10000 dev	the movie Victim starred who?
9952 test	what type of film is Midnight Run?
Yahoo! Answers	What is Elvis's hairstyle called?
1898 train	Who is this kid in Mars Attacks?
271 dev	who invented denim jeans?
542 test	who's the woman on the progressive.com commercials?

Table 3.12: Statistics of the TREC QA, WikiMovies and Yahoo! Answers factoid datasets.

Answer Scoring Module

Finally, *EviNets* uses the aggregated signals to predict the answer scores, to rank them, and to return the best candidate as the final answer to the question. For this purpose, we use two fully-connected neural network layers with the ReLU activation function, with 32 and 8 hidden units respectively. The model was trained end-to-end by optimizing the cross entropy loss function using the Adam algorithm [112].

3.3.2 Experimental Evaluation

To test our framework we used TREC QA [186], WikiMovies [136] benchmarks and the new Yahoo! Answers dataset[17] derived from factoid questions posted on the CQA website (Table 3.12). In all experiments, embeddings

---

[17]available for research purposes at http://ir.mathcs.emory.edu/software-data/

were initialized with 300-dimensional vectors pre-trained with Glove [147]. Embeddings for multi-word entity names were obtained by averaging the word vectors of constituent words.

Baselines

As baselines for different experiments depending on availability and specifics of a dataset we considered the following methods:

- IR-based QA systems: *AskMSR* [41] and *AskMSR+* [195], which select the best answer based on the frequency of entity mentions in retrieved text snippets.
- KBQA systems: *SemPre* [22] and *Aqqu* [20], which identify possible topic entities of the question, and select the answer from the candidates in the neighborhood of these entities in a KB.
- Hybrid system *QuASE* [186] detects mentions of knowledge base entities in text passages, and uses the types and description information from the KB to support answer selection.
- Hybrid system *Text2KB* [166], which uses textual resources to improve different stages of the KBQA pipeline, described in Section 3.2.
- Memory Networks: *MemN2N* [185] and *KV MemN2N* [136] represent relevant information with embeddings, and summarize the memories into a single vector using the soft attention mechanism. Additionally, KV MemN2N splits memories into key-value pairs, where keys are used for matching against the question, and values are used to summarize the memories.

TREC QA dataset

The TREC QA dataset is composed of factoid questions, which can be answered with an entity, and were used in TREC 8-12 question answering

Method	P	R	F1
SemPre	0.157	0.104	0.125
Text2KB	0.287	0.287	0.288
AskMSR+	0.493	0.490	0.491
QuASE (text)	0.550	0.550	0.550
QuASE (text+kb)	0.579	**0.579**	**0.579**
MemN2N	0.333	0.328	0.330
KV MemN2N	0.517	0.500	0.508
EviNets (text)	0.580	0.560	0.569
EviNets (text+kb)	**0.585**	0.564	0.574

Table 3.13: Precision, Recall and F1 of KB- and Text-based question answering methods on the TREC QA dataset. The improvements over the Key-Value memory networks are statistically significant at p-value < 0.01.

tracks. Similarly to [186] we used web search (using the Microsoft Bing Web Search API) to retrieve top 50 documents, parsed them, extracted sentences and ranked them using tf-idf similarity to the question. To compare our results with the existing state-of-the-art, we used the same set of candidate entities as used by the QuASE model. We note that the extracted evidence differs between the models, and we were unable to match some of the candidates to our sentences. For text+kb experiment, just as QuASE, we used entity descriptions and types from Freebase knowledge base.

Table 3.13 summarizes the results. *EviNets* achieves competitive results on the dataset, beating KV MemN2N by 13% in F1 score, and, unlike QuASE, does not rely on expensive feature engineering and does not require any external resources to train.

WikiMovies dataset

The WikiMovies dataset contains questions in the movies domain along with relevant Wikipedia passages and the OMDb knowledge base. Since KVMemN2N already achieves an almost perfect result answering the questions using the KB, we focus on using the provided movie articles from Wikipedia. We followed the preprocessing procedures described in [136]. Unlike TREC QA, where there are often multiple relevant supporting pieces of evidence, answers in the WikiMovies dataset usually have a single relevant sentence, which, however, mentions multiple entities. To help the model distinguish the correct answer, and explore its abilities to encode structured and unstructured data, we generated additional *entity type* triples. For example, if an entity e appears as an object of the predicate `directed_by` in OMDb, we added the `[e, type, director]` triple. As baselines, we used MemN2N and KV MemN2N models, and the results are presented in Table 3.14. As we can see, with the same setup using individual sentences as evidence/memories *EviNets* significantly outperforms the KV MemN2N model by 27%. Moreover, the proposed approach can effectively incorporate additional entity type RDF triples, and significantly improve the performance over the text-only version. It is important to emphasize that the best-reported results of memory networks were obtained using *entity-centered windows* as memories, which requires special pre-processing and increases the number of memories. Additionally, these models used *all* of the KB entities as candidate answers, whereas *EviNets* relies only on the mentioned ones, which is a more scalable scenario for open-domain question answering, where it is not realistic to score millions of candidate answers in real-time.

Method	Accuracy
MemN2N (wiki windows)	0.699*
KV MemN2N (wiki windows)	0.762*
AskMSR (entities)	0.314
KV MemN2N (wiki sent)	0.524
EviNets (wiki sent)	0.616
EviNets (wiki sent + entity types)	**0.667**

Table 3.14: Accuracy of EviNets and baseline models on the WikiMovies dataset. The results marked * are obtained using a different setup, *i.e.,* they use pre-processed entity window memories, and the whole set of entities as candidates.

New Yahoo! Answers factoid questions dataset

Yahoo! recently released a dataset with search queries, which lead to clicks on factoid Yahoo! Answers questions, identified as questions with the best answer containing less than 3 words and a Wikipedia page as the specified source of information[18]. This dataset contains 15K queries, which correspond to 4725 unique Yahoo! Answers questions (Table 3.12). We took these questions, and mapped answers to KB entities using the TagMe entity linking library [70]. We filtered out questions, for which no answer entities with a good confidence[19] were identified, *e.g.,* date answers, and randomly split the rest into training, development and test sets, with 2711 questions in total. Similarly to the TREC QA experiments, we extracted textual evidence using Bing Web Search API, by retrieving top 50 relevant documents, extracting the main content blocks, and splitting them into sentences. We applied the TagMe entity linker to the extracted sentences, and considered all entities

---

[18]L27 dataset https://webscope.sandbox.yahoo.com

[19]A minimum ρ score of 0.2 from TagMe was required.

Method	P	R	F1
Aqqu	0.116	0.117	0.116
Text2KB	0.170	0.170	0.170
AskMSR (entities)	0.175	0.319	0.226
MemN2N	0.072	0.131	0.092
KV MemN2N	0.126	0.228	0.162
EviNets (text)	0.210	0.383	0.271
EviNets (text+kb)	**0.226**	**0.409**	**0.291**
Oracle	0.622	1.0	0.767

Table 3.15: Precision, Recall and F1 of different methods on Yahoo! Answers factoid QA dataset. The Oracle performance assumes candidate answers are ranked perfectly and is bound by the performance of the initial retrieval step.

of mentions with the confidence score above the 0.2 threshold as candidate answers. For candidate entities we also retrieved relevant KB triples, such as entity types and descriptions, which extended the original pool of evidences.

Table 3.15 summarizes the results of *EviNets* and some baseline methods on the created Yahoo! Answers dataset. As we can see, knowledge base data is not enough to answer most of these questions, and a state-of-the-art KBQA system Aqqu gets only 0.116 precision. Adding textual data helps significantly, and Text2KB improves the precision to 0.17, which roughly matches the results of the AskMSR system, that ranks candidate entities by their popularity in the retrieved documents. EviNets significantly improves over the baseline approaches, beating AskMSR by 28% and KV MemN2N by almost 80% in F1 score. Using text along with KB evidence gave higher performance metrics, boosting F1 from 0.271 to 0.291.

In the above-mentioned experiments we used bag-of-words representation for questions and evidence, and a reasonable question is whether more com-

Method	P	R	F1
EviNets (text+kb): BOW	0.226	0.409	0.291
EviNets (text+kb): biLSTM	0.196	0.356	0.252

Table 3.16: Precision, Recall and F1 of EviNets with bag-of-words and bidirectional LSTM representations of questions and evidence.

plicated methods could achieve higher results. Table 3.16 compares bag-of-words representation with bidirectional LSTM [207]. On Yahoo! Answers dataset BOW representation showed better performance, which is most likely due to a relatively low size of the dataset for the variety of questions present there.

3.3.3   Discussion

EviNets, described in this section, is a neural network for question answering, which encodes and aggregates multiple evidence signals to select answers. Experiments on TREC QA, WikiMovies and Yahoo! Answers datasets demonstrate that *EviNets* can be trained end-to-end to use both the available textual and knowledge base information. EviNets improves over the baselines, both in cases when there are many or just a few relevant pieces of evidence, by helping build an aggregate picture and distinguish between candidates, mentioned together in a relevant memory, as is the case for WikiMovies dataset. The results of the experiments also demonstrate that EviNets can incorporate signals from different data sources, *e.g.,* adding KB triples helps to improve the performance over text-only setup. As a limitation of the approach and a direction for future research, *EviNets* could be extended to support dynamic evidence retrieval, which would allow retrieving additional answer candidates and evidence as needed.

3.4   Summary

This Chapter introduced several approaches for combining unstructured, semi-structured and structured data sources to improve factoid question answering. Relation extraction from question-answer pairs aims at filling some gaps in KB fact coverage. The experiments show, that we can use distant supervision to extract factual knowledge from community question answering archives and increase the recall of the existing sentence-based relation extraction techniques. However, extraction techniques are not perfect and suffer from both precision and recall losses. As an alternative strategy, we can use semantic annotations of entity mentions in a text to connect knowledge base and textual data. Such annotation allows to quickly find relevant textual resources and improve KBQA methods, as demonstrated by Text2KB model, or use both data sources together, as pieces of supporting evidence for generated answer candidates. Diverse information can be mapped into the embedding space and aggregated together with a neural network architecture, such as *EviNets*.

Factoid questions represent just a part of user information needs. Many problems require a more elaborate response, such as a sentence, list of instructions or, in general, a passage of text. Such questions are usually referred to as non-factoid questions and they will be the focus of Chapter 4.

Chapter 4

Improving Non-factoid Question Answering

Factoid questions studied in Chapter 3 represent only a fraction of user information needs, and there are many other types of questions, which cannot be answered with entity names or dates. The variety of user information needs is reflected in different types of questions, that people post to community question answering websites [84, 100, 131]. Such questions usually require a longer response, *e.g.,* a paragraph of text, list of instructions, *etc.* For the majority of such questions, modern search engines still return the "10 blue links", and delegate the task of digging into the information and extracting relevant pieces of knowledge to the user, which can be quite time-consuming. In this Chapter, I focus on improving question answering for such generic information needs.

Previous research on non-factoid question answering either focused on a small subset of questions (*e.g.,* definition questions [92]), or considered this as a problem of ranking existing answers in CQA archives, which can be reused to answer new questions [48, 174]. To advance the research in the area of automatic question answering for a general class of user information

needs in 2015 TREC started a series of LiveQA evaluation campaigns[1]. The task of TREC LiveQA track is to develop a real-time system to answer real user questions, that are posted live to Yahoo! Answers[2] community question answering platform.

This chapter describes the methods and ideas I implemented in a system, that participated in 2015 and 2016 versions of the track. The system uses a combination of semi-structured information, *i.e.,* question-answer pairs from different CQA platforms, with unstructured information, which can be extracted from regular web documents. The former strategy of retrieving similar previously posted questions was shown to be quite effective [48, 174], as it allows a system to return a naturally looking answer in cases when a good match was found. However, many similar questions are formulated differently, which complicates the retrieval problem, additionally, many incoming information needs are still unique and there are simply no similar questions in the archive. In this case a system can extract its answer from potentially relevant passages of regular web documents. In Section 4.1 I will describe the architecture of *EmoryQA*: the system I developed to participate in the TREC LiveQA shared task.

Despite the indisputable improvements in automatic answer selection[3], the analysis of results of TREC LiveQA 2015 task demonstrated, that we still have a big gap in performance between human and automatic question answering. Mistakes, that were made by trained machine learning models for answer passage selection, could be easily spotted by a human in fractions of a second even when a person does not have enough expertise in the area of the question. To build on this observation, we looked into how an automatic system can integrate crowdsourcing to improve its performance. More

---

[1]http://trec-liveqa.org

[2]http://answers.yahoo.com/

[3]http://aclweb.org/aclwiki/index.php?title=Question_Answering_(State_of_the_art)

specifically, we propose to use feedback of a crowd of workers to extend the answers pool and obtain quality labels for generated answer candidates. Section 4.2 describes the design of the crowdsourcing component and the results of TREC LiveQA 2016, which demonstrated its effectiveness for near real-time question answering.

In summary, the main contributions of this chapter include:

- An open-source state-of-the-art automatic question answering system for a general class of user information needs, evaluated at TREC LiveQA 2015 and 2016 tracks. The architecture of the system and evaluation results were published in TREC conference proceedings [162, 165].

- A novel hybrid question answering system, that incorporates a crowd-sourcing module, but still operates in near real-time, and significantly improves performance over the pure automatic approach. This work was presented as a full paper at HCOMP 2016 [164].

4.1 Ranking Answers and Web Passages for Non-factoid Question Answering

In this section, I describe the architecture of *EmoryQA* (Figure 4.1), the automatic question answering system, which combines semi-structured data from CQA archives and unstructured web document passages to cover a variety of questions that users have.

Community Question Answering websites became quite popular and millions of users post their questions there and hope to receive a response from the community. The questions on these websites typically consist of question title, body, and category. Since CQA archives accumulated a lot of question data, we chose this question format for our development, similar to the setup
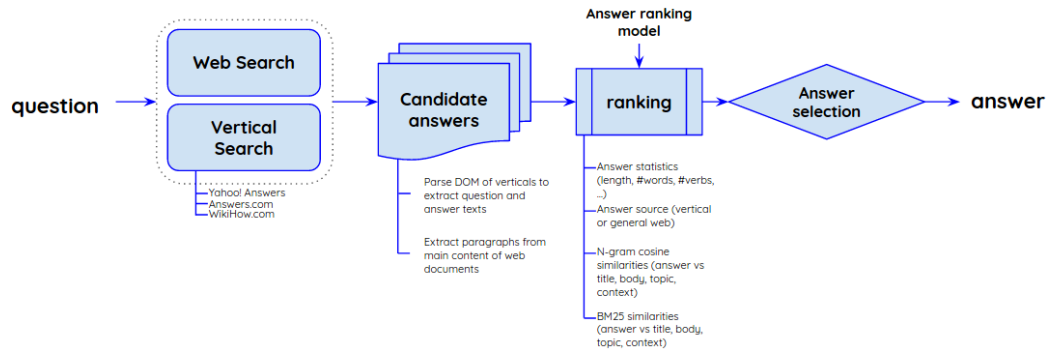
Figure 4.1: Architecture of the EmoryQA non-factoid question answering system, participated in TREC LiveQA shared task.

proposed for the TREC LiveQA shared task. Below you can see an example question from a CQA website:

Question category: Astronomy & Space
Question title: Why do people claim the Earth is not the center of the universe?
Question body: Clearly the sun and moon are moving around the Earth otherwise we would not have night and day.

People often have similar tasks and situations which pose the same questions, and therefore community question answering platforms receive many similar questions, which can be answered in a similar way. Researchers have found out, that a good reply to similar questions can be reused to answer new user questions [48, 174], and a number of approaches have been proposed to rank candidate answer passages [75, 171, 180, 187, 232]. *EmoryQA* builds on these works and includes components to retrieve a set of candidate answers from a number of community question answering platforms, such as Yahoo! Answers [4], Answers.com [5] and WikiHow [6]. Besides some frequent

---

[4]http://answers.yahoo.com/

[5]http://answers.com/

[6]http://wikihow.com/

questions, there is always a long tail of requests, which are either unique or phrased differently from those that were previously submitted to a CQA website [28]. To help a user with such questions, *EmoryQA* includes a general web search module, which extracts passages from regular web pages to extend the pool of candidate answers.

All candidate answers, extracted from either CQA verticals or regular web documents, are ranked together by a trained answer ranking model, and *EmoryQA* returns the top scoring passage as the final answer to the question. Next, I will describe candidate generation and ranking modules in more detail.

4.1.1   Candidate Answer Generation

When the *EmoryQA* system receives a user question, it first generates a set of candidate answers from CQA vertical and regular web search data sources. User questions vary in language [5] and the level of details, therefore, to increase the recall and retrieve as many relevant results as possible, for each question we generate multiple search queries:

- Question title, which most often captures the gist of the question
- Two longest question sentences (detected by the presence of the question word at the beginning or question mark at the end of a sentence) from the title and body of the question. In some cases, the real user question is hidden inside the body, while the title just provides the overall topic of the question.
- Concatenation of the question word, verbs and top-5 terms from the question title by inverse document frequency[7].

---

[7]IDF of terms are estimated using Google N-gram corpus: https://catalog.ldc.upenn.edu/LDC2006T13

For CQA verticals, *EmoryQA* issues the queries to the built-in search interfaces of Yahoo! Answers, Answers.com and WikiHow.com and extracts top-10 similar questions with the corresponding answers, posted by the community, and adds them to the candidates pool. For regular web search, we rely on the Bing Web Search API[8], which we query to retrieve top-10 relevant documents and extract paragraphs of text from their main content, as detected by a method based on [114].

In addition to the candidate answers themselves, *EmoryQA* extracts certain meta-data, that helps to estimate the relevance of a passage to the current question. For regular web page paragraphs, it is useful to know the topic of the page (*e.g.,* its title) and the context (such as text that immediately precedes the paragraph in the document), as shown by Di Wang and Eric Nyberg in [206]. For CQA answers, our system stores the text of the corresponding question title, body, and category. For convenience, we will refer to this question title and web page title as *"answer topic"*, while the body of the retrieved question and the preceding text block for web candidates as *"answer context"*.

4.1.2   Candidate ranking

EmoryQA represents each candidate answer with a set of features (Table 4.1). To predict the quality of the accumulated candidates and select the best answer we use a trained learning-to-rank model, which sorts the answers and selects the top response as the final answer to the question. There are multiple ways to train such a ranking model depending on the type of data available.

---

[8]https://datamarket.azure.com/dataset/bing/searchweb

Answer statistics
— Length in chars, words and sentences
— Average number of words per sentence
— Fraction of non-alphanumeric characters
— Number of question marks
— Number of verbs
Answer source
— Binary feature for each of the search verticals: Web, Yahoo! Answers, Answers.com, WikiHow.com
N-gram matches
— Cosine similarities using uni-, bi- and tri-gram representations of the question title and/or body, and answer text, topic or context
— The lengths of longest spans of matched terms between question title and/or body, and answer text, topic or context
Information Retrieval score
— BM25 scores between question title and/or body, and answer text, topic or context

Table 4.1: The list of candidate answer ranking features used by the EmoryQA non-factoid question answering system.

Training answer ranking model using unlabeled CQA data

The problem of learning to rank for information retrieval usually assumes the presence of labeled data, that provides some kind of partial order of documents for a set of queries [129]. For question answering that would mean that one needs to provide relevance labels for passages, which can be retrieved for a question. This process is more expensive than for regular document search, since there are many more passages than documents, and a QA system is not even restricted to return a continuous text from a single document.
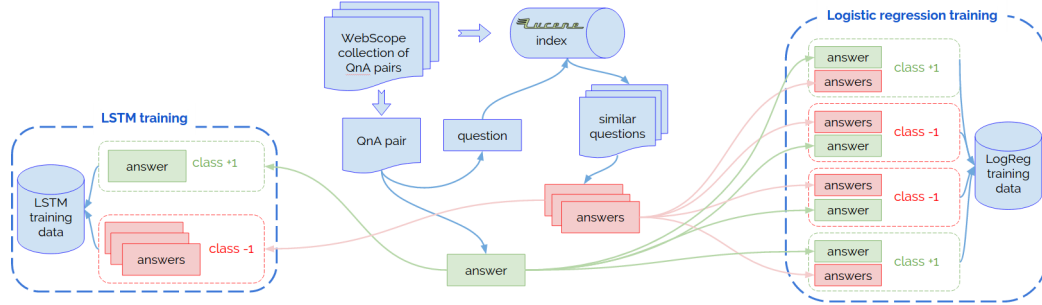
Figure 4.2: Dataset generation workflow for training logistic regression and LSTM answer ranking models used in EmoryQA system participated in TREC LiveQA 2015.

When no explicit labeled data is available, we can use implicit information available in CQA archives, *e.g.,* selected best answers. The version of *EmoryQA*, developed to participate in TREC LiveQA 2015, had two trained models: LSTM recurrent neural network based model, used as one of the features for the final logistic regression model that scores all candidates and selects the best one as the answer. To train these models I used WebScope Yahoo! Answers dataset[9], and the process of building the training datasets is explained on Figure 4.2.

LSTM model. Deep learning models had a huge success in image and speech problems and showed very promising results in natural language processing and question answering, *e.g.,* [242, 207] to name a few. Long Short-Term Memory (LSTM) [94] is a particular architecture of recurrent neural networks that helps with the vanishing gradients problems. The model reads the question and answer tokens and produces a probability score based on a vector representation of a QnA pair. Figure 4.3 shows the structure of the model.

Question (title with the body) and answer texts are tokenized, punctuation

---

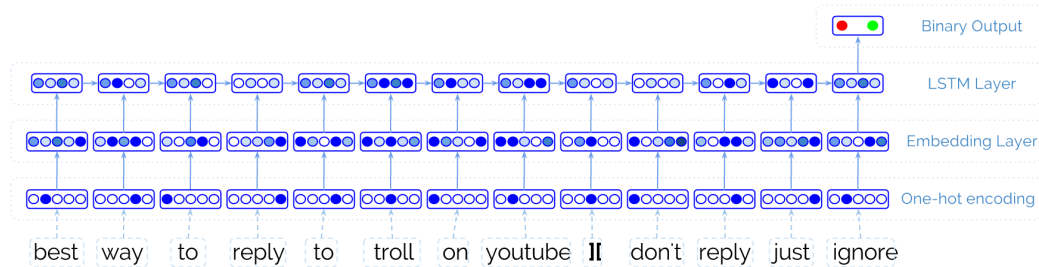[9]https://webscope.sandbox.yahoo.com/catalog.php?datatype=l

Figure 4.3: LSTM model for answer scoring used in EmoryQA system, which participated in TREC LiveQA 2015 shared task. The example shows a QnA pair where the question is "Best way to reply to trolls on youtube?" and the answer is "Do not reply, just ignore".

characters are removed and for each token lowercased lemma is taken. The sequences are limited to 100 elements and concatenated through a sentinel separator character so the model could learn where the question ends and the answer starts. The hidden state of the model after the whole sequence is processed is used by logistic regression unit to output a probability, that a candidate answers the question well.

The model was trained in a pointwise learning-to-rank fashion [129], *i.e.,* we trained the model to distinguish between the selected best answer and some negative examples. Random negative examples would be too unrelated to the current question, therefore I chose to use answers to similar questions only. All QnA pairs were indexed with Lucene[10] and similar questions were retrieved using the built-in BM25 retrieval model. For each question and correct answer pair from the dataset 10 similar questions were retrieved and the corresponding answers were used as negative examples for training, even though some of them can indeed be relevant to the original question.

The model was implemented using the Keras[11] library. I used an embedding

---

[10]https://lucene.apache.org/

[11]http://keras.io

and hidden layers of dimension 128 and the vocabulary size of 1M words. The model was trained using the Adam optimization technique [112] with mini batches of 200 instances for 100 epochs.

Logistic regression model. The final model that ranks all answer candidates is a linear L2-regularized logistic regression model. To train the model I used a different from LSTM model split of QnA pairs from Yahoo! Answers WebScope dataset. For each question, the corresponding "best answer" is taken as the correct one. To get a sample of negative examples Lucene index is used again and answers to 10 most similar questions are retrieved. Different from LSTM model training, here I took a pairwise approach for learning-to-rank and generated training examples from pairs of different answers to the same question, where one answer is the correct one. That is, let the current question be Q, its "correct" answer A^*, and retrieved candidates $A_1, ..., A_n$. Each candidate is represented with a set of features: $f(Q, A^*)$, $f(Q, A_1)$, ..., $f(Q, A_n)$. For each $i = 1..n$ we create two training instances, i.e. class 1: $\langle A^*, A_i \rangle$ and class -1: $\langle A_i, A^* \rangle$. Each such instance is represented with pairwise differences of features, e.g. $\langle A^*, A_i \rangle : f_{pair}(Q, \langle A^*, A_i \rangle) = f(Q, A^*) - f(Q, A_i)$. The trained model is linear, therefore if $w(f(Q, A^*) - f(Q, A_i)) > 0$ then $wf(Q, A^*) > wf(Q, A_i)$ and we can rank candidates by the score produced by the model, i.e. $wf(Q, A_i)$.

Learning to rank answers with graded relevance data

The approach described in the previous section works well, but the automatic labeling introduces a certain level of noise. When we have a set of candidate answers with specified graded relevance, as was the case for TREC LiveQA 2016, it is possible to utilize this cleaner data to train an answer ranking model. Therefore, in TREC LiveQA 2016 for *EmoryQA* we used the listwise approach to learning-to-rank and trained a LambdaMART model [42], which gave the best results on the development set. This model was trained using

the RankLib library[12] on the data from the previous year TREC LiveQA task[13]. This data includes 1087 questions with answers provided by the participants, each of which was rated on a scale from 1(bad) to 4(excellent) by professional NIST assessors.

4.1.3 Evaluation

The experimental evaluation of our *EmoryQA* non-factoid question answering system was done on TREC LiveQA 2015 and 2016 tasks. The task was to build a live question answering system to respond to user questions, which were sampled from the live stream of questions posted to the Yahoo! Answers community question answering website by its users. Each input question consisted of a short question title, body, and category. A QA system had to provide an answer of 1000 characters or less within a 1 minute period using any available data source. A reader can refer to [2, 3] for more details on TREC LiveQA 2015 and 2016 results and analysis.

During the evaluation periods, each system received 1,087 and 1,088 questions correspondingly, and responses were recorded by the organizers. The answers to the questions were judged by the organizers on a scale:

4: Excellent - a significant amount of useful information, fully answers the question.

3: Good - partially answers the question.

2: Fair - marginally useful information.

1: Bad - contains no useful information for the question.

-2 - the answer is unreadable (only 15 answers from all runs were judged as unreadable).

The official performance metrics used for the tasks are:

---

[12]https://sourceforge.net/p/lemur/wiki/RankLib/

[13]https://sites.google.com/site/trecliveqa2016/liveqa-qrels-2015

	avg score (0-3)	succ@2+	succ@3+	succ@4+	prec@2+	prec@3+	prec@4+
Results from TREC LiveQA 2016							
$HUMAN_{qual}$	1.561	0.655	0.530	0.375	0.855	0.692	0.490
$HUMAN_{speed}$	1.440	0.656	0.482	0.302	0.784	0.576	0.362
1. *EmoryCRQA*	1.260	0.620	0.421	0.220	0.644	0.438	0.228
2. CMU OAQA	1.155	0.561	0.395	0.199	0.596	0.420	0.212
3. **EmoryQA**	1.054	0.519	0.355	0.180	0.530	0.362	0.184
Avg results	0.643	0.329	0.212	0.104	0.422	0.271	0.131
Results from TREC LiveQA 2015							
1. CMUOAQA	1.081	0.532	0.359	0.190	0.543	0.367	0.179
2. ecnucs	0.677	0.367	0.224	0.086	0.401	0.245	0.094
3. NUDTMDP1	0.670	0.353	0.210	0.107	0.369	0.219	0.111
...							
7. **EmoryQA**	0.608	0.332	0.190	0.086	0.408	0.233	0.106
Avg results	0.467	0.262	0.146	0.060	0.284	0.159	0.065

Table 4.2: Top results of the TREC LiveQA 2015 and 2016 shared tasks. EmoryQA is the described fully automatic question answering system. EmoryCRQA is a system with the integrated crowdsourcing module, described in Section 4.2.

- **avg-score(0-3)**: average score over all questions, where scores are translated to 0-3 range (1 is subtracted from each judgment). This metric considers "Bad", unreadable answers and unanswered questions, as scored 0.
- **succ@i+**: success at i+ metrics measures the fraction of answers with score i or greater (i=1..4).
- **prec@i+**: precision at i+ measures the number of questions with score i or greater (i=2..4) divided by the number of answered questions.

Table 4.2 provides the results of the challenge in 2015 and 2016. The full

results can be found in the official overview reports [2, 3]. As we can see, *EmoryQA* achieves competitive results during both 2015 and 2016 evaluation campaigns. Improvements made in 2016, *i.e.,* additional data sources, list-wise learning to rank model, trained on data from previous year task, helped to improve the average answer score by $\approx 70\%$.

Yahoo! Answers (qual) and Yahoo! Answers (speed) are answers, collected by the organizers from the original questions after a week from the postings. The *speed* answers are those submitted chronologically first, while *qual* were selected as best answers by the asker, or by Yahoo's quality scoring algorithm. As we can see, the quality of community answers are still far better, than those of QA systems. In particular almost $\sim 50\%$ of the questions received a perfect answers from Yahoo! Answers community, compared to $\sim 22\%$ from the winning system. However, $\sim 20\%$ of the questions did not receive any response from the community, even though human had the whole week to respond, which stresses an importance of developing automatic approaches even more.

The distribution of answer quality scores and the difference in precision between community and QA systems show, that the later often returns an answer, which is not relevant to the question. In addition, a quick error analysis also revealed that automatic systems often have trouble recognizing non-relevant passages, which are easily detected by a human even without domain expertise. The next section describes *EmoryCRQA*, the winning approach from TREC LiveQA 2016, which utilizes crowdsourcing inside a real-time question answering system, and significantly improves performance over the fully automatic approach.

4.2   CRQA: Crowd-powered Real-time Automatic Question Answering System

As we have seen in the previous section, existing question answering systems are still far from being able to handle every human question. Answers, submitted by the community users for TREC LiveQA 2016 questions, are $\approx 35\%$ better than the answers from the best performing fully automatic system: CMU OAQA (Figure 4.2). One way to overcome the above-mentioned challenges in complex question answering is to develop a hybrid human-computer question answering system, which could consult a crowd of workers in order to generate a good response to the user question. This section first describes the initial analysis of the feasibility of obtaining different types of feedback from a crowd in real-time. Then, I present *EmoryCRQA*, a crowd-powered, near real-time automated question answering system for complex informational tasks, that incorporates a crowdsourcing module for augmenting and validating the candidate answers.

More specifically, in this section we answer the following questions:

1. Can crowdsourcing be used to judge the quality of answers to non-factoid questions under a time limit?

2. Is it possible to use crowdsourcing to collect answers to real user questions under a time limit?

3. How does the quality of crowdsourced answers to non-factoid questions compare to original CQA answers, and to automatic answers from TREC LiveQA systems?

4. Can crowdsourcing be used to improve the performance of a near real-time automated question answering system?

5. What is the relative contribution of candidate answer ratings and answers provided by the workers to the overall question answering performance?

6. What are the trade-offs in performance, cost, and scalability of using crowdsourcing for real-time question answering?

Part of the described results was published at Human-Computer Question Answering workshop at NAACL 2016 conference [169], and another appeared as a full paper titled "CRQA: Crowd-powered Real-time Automated Question Answering System" on HCOMP 2016 conference [164].

4.2.1 Evaluating crowdsourcing for question answering

In this section I explore two ways crowdsourcing can assist a question answering system that operates in (near) real-time: by providing answer *validation*, which could be used to filter or re-rank the candidate answers, and by *creating* the answer candidates directly. To test the hypothesis that crowd workers can quickly provide reliable feedback we conducted a series of crowdsourcing experiments using the Amazon Mechanical Turk platform[14]. I used questions from the TREC LiveQA 2015 shared task, along with the systems answers, rated by the NIST assessors[15]. The questions for the task were selected by the organizers from the live stream of questions posted to the Yahoo! Answers CQA platform on the day of the challenge (August 31, 2015). For these questions, we also crawled their community answers, that were eventually posted on Yahoo! Answers[16].

Answer validation experiment

To check if crowdsourcing can be used to judge the quality of answers under a time limit, I asked workers to rate answers to a sample of 100 questions using the official TREC rating scale:

---

[14]http://mturk.com

[15]https://sites.google.com/site/trecliveqa2016/liveqa-qrels-2015

[16]As the answer we took the one selected as the "Best answer" by the author of the question or by the community.

1. Bad — contains no useful information
2. Fair — marginally useful information
3. Good — partially answers the question
4. Excellent — fully answers the question

We chose to display 3 answers for a question, which were generated by three of the top-10 automatic systems from TREC LiveQA 2015 evaluation [2]. To study the effect of time pressure on the quality of judgments we split participants into two groups. One group made their assessments with a 1-minute countdown timer shown to them, while the other could complete the task without worrying about a time limit. Within each group, we assigned three different workers per question, and the workers were compensated at a rate of $0.05 per question for this task.

The interface for collecting answer ratings is illustrated in Figure 4.4a[17]. On top of the screen, workers were shown the instructions on the task, and question and answers were hidden at this time. They were instructed to read the question, read the answers, and rate each answer's quality on a scale from 1 (Bad) to 4 (Excellent), and finally, choose a subset of candidates that best answer the question. Upon clicking a button to indicate that they were done reading the instructions, the question, a 60-second countdown timer and 3 answers to the question appeared on the screen. At the 15 second mark, the timer color changed from green to red. In the experiments without time pressure, the timer was hidden, but we still tracked the time it took for the workers to complete the task.

At the end, we collected 6 ratings (3 with and 3 without time pressure) for each of three answers for a sample of 100 questions, which makes it a total of 1800 judgments. Each answer also has an official NIST assessor rating on the same scale. Figure 4.5 shows the correlation between the official NIST

---

[17]The screenshots show the final state of the form, as we describe later in this sections fields were unhidden step-by-step for proper timing of reading, answering and validation.

(a) Answer validation form   (b) Answer crowdsourcing form

Figure 4.4: User Interface for the answer quality judgment experiment using real-time crowdsourcing.

assessor relevance judgments and the ratings provided by our workers. The Pearson correlation between the scores is $\rho = 0.52$. The distribution of scores shows that official assessors were very strict and assigned many extreme scores of 1 or 4, whereas mechanical turk workers preferred intermediate 2s and 3s. The results did not show any significant differences between experiments with and without time pressure. Figure 4.6 shows that even
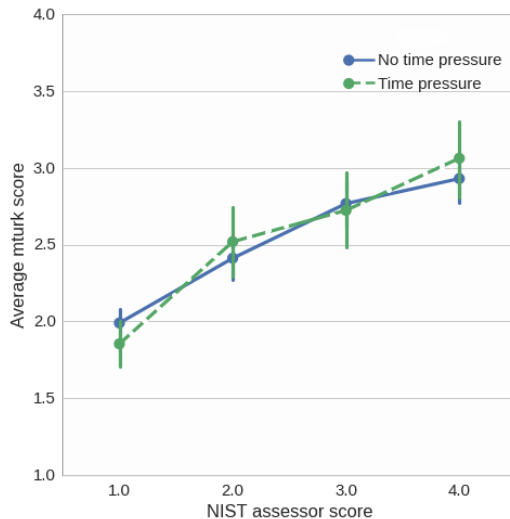
Figure 4.5: Correlation between NIST assessor scores and crowdsourced ratings with and without time limit on the work time for answers from a sample of 100 questions from TREC LiveQA 2015 task.

though the median time to rate all three answers is around 22-25 seconds in both experiments, the upper bound is significantly lower in the experiment with the time pressure.

Therefore, we conclude that in general we can trust crowdsourced ratings, and on average one minute is enough to judge the quality of three answers to CQA questions.

Answer generation experiment

In another experiment, designed to check whether crowd workers can provide an answer to a given question within a limited amount of time, we asked different workers to answer the questions from TREC LiveQA 2015. I split the workers into two groups and displayed a one-minute countdown timer for one of them. We left a grace period and let the workers submit their answers after the timer had run out. The workers received a $0.10 compensation for
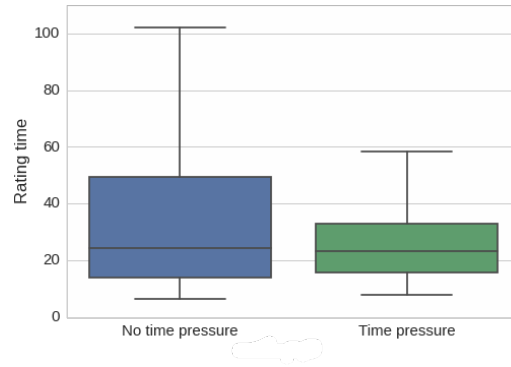
Figure 4.6: Box plot of answer rating time by workers on Amazon Mechanical Turk platform with and without time pressure.

each answer. The form for answer crowdsourcing is shown in Figure 4.4b, and similar to the answer rating form, it starts with a set of instructions for the task. We let the users browse the internet if they were not familiar with the topic or could not answer the question themselves. To prevent them from finding the original question on Yahoo! Answers, we included a link to Google search engine with a date filter enabled[18]. Using this link, workers could search the web as it was on 8/30/2015, before TREC LiveQA 2015 questions were posted and therefore workers were in the same conditions as automatic systems on the day of challenge[19]. Initially, the question was hidden for proper accounting of question-reading and answering times. Upon clicking a button to indicate that they were done reading the instructions, a question appeared along with a button, which needed to be clicked to indicate that they were done reading the question. After that, the answering form appears, it contained four fields:

1. Does the question make sense: "yes" or "no" to see if the question was

---

[18]https://www.google.com/webhp?tbs=cdr:1,cd_max:8/30/2015

[19]The ranking of search results could be different on the day of the challenge and for our workers

    comprehensible

2. Are you familiar with the topic: A yes or no question to evaluate whether the worker has had prior knowledge regarding the question topic

3. Answer: the field to be used for the user's answer to the given question

4. Source: the source used to find the answer: URL of a webpage or NA if the worker used his own expertise

At the end, we collected 6 answers (3 with and without time pressure) for each of the 1087 LiveQA'15 questions. Since we have answers from different sources, let's introduce the following notations:

- *Yahoo! Answers* - answers eventually posted by users on Yahoo! Answers for the original questions
- *Crowd* - answers collected from Mechanical Turk workers without time pressure
- *Crowd-time* - answers collected from Mechanical Turk workers with one minute time pressure
- *LiveQA winner* - answers from the TREC LiveQA'15 winning system

Table 4.3 summarizes some statistics on the answers. The first thing to notice is that, unlike CQA websites, where some questions are left unanswered, by paying the crowd workers we were able to get at least one answer for all LiveQA questions (after filtering "No answer" and "I do not know" kind of responses). The length of the answers, provided by Mechanical turk users is lower, and time pressure forces users to be even more concise. The majority of workers ($\sim 90\%$) did not use the web search and provided answers based on their experience, opinions and common knowledge.

Statistic	Y!A	mTurk	mTurk-time	LiveQA'15 winner
% answered	78.7%	100.0%	100.0%	97.8%
Length (chars)	354.96	190.83	126.65	790.41
Length (words)	64.54	34.16	22.82	137.23

Table 4.3: Statistics of different types of answers for Yahoo! Answers questions.
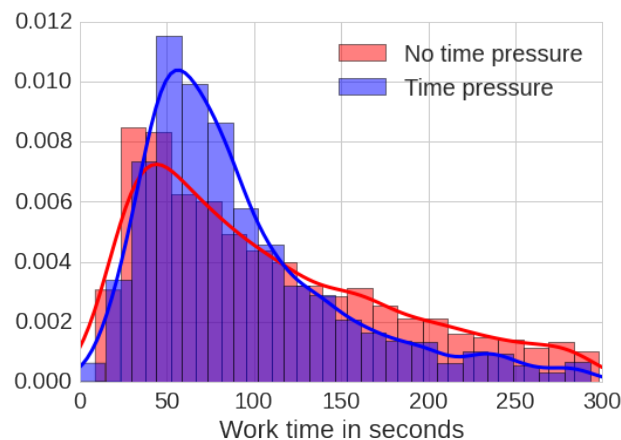
Figure 4.7: Distribution of answering times for experiments with and without time pressure.

From Figure 4.7 we can see that adding time pressure shifts the distribution of answering times[20]. The tail of longer work times for no time limit experiment becomes thin with time restrictions and the distribution peaks around one minute.

---

[20]We had separate timers for reading the instructions, the question, and writing the answer, the inclusion of instruction-reading time is why the total time could be more than 1 minute
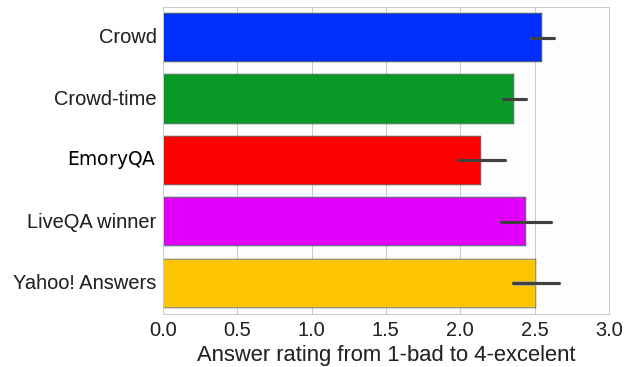
Figure 4.8: Average scores of different types of answers to Yahoo! Answers questions.

Answer quality comparison

Finally, to compare the quality of the collected answers with the automatic system and CQA responses we pooled together the crowdsourced answers, the answers from the winning and *EmoryQA* systems from LiveQA'15, and the original answers crawled from Yahoo! Answers. We took a sample of 100 questions and repeated the answer rating experiment on this data. Each answer was judged by 3 different workers (without time pressure), and their scores were averaged. Figure 4.8 displays the plot with average score for answers from different sources. Quite surprisingly the quality of collected answers turned out to be comparable to those of CQA website users. Average rating of answers produced by the winning TREC LiveQA system is also pretty close to human answers. Finally, as expected, time pressure had its negative effect on the quality, however, it is still significantly better than the quality of *EmoryQA* answers.

Analysis of the score distribution (Figure 4.9) sheds some light on the nature of the problems with automatic and human answers. The automatic systems generate non-relevant answers ($score = 1$) more often than human, either because the systems fail to retrieve relevant information or to distin-
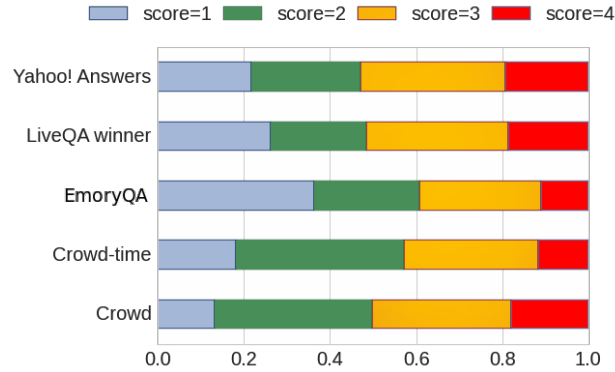
Figure 4.9: Distribution of scores for different types of answers to Yahoo! Answers questions.

guish between useful and non-useful answer candidates. However, by having a larger information store, *e.g.,* the Web, automated QA systems can often find a perfect answer ($score = 4$), while crowd workers tend to give generally useful, but less perfect responses ($score = 2, 3$).

Our results suggest that the "crowd" can quickly give a reasonable answer to most CQA questions. However, some questions require a certain expertise, which a common crowd worker might not possess. One idea to tackle this challenge is to design a QA information support system, which a worker can use to help them find additional information. For example, in our experiment, we let workers use a web search to find answers if they were unfamiliar with the topic; more effective search interfaces may be helpful.

4.2.2   System Design

The findings described in the previous section were used to implement our *EmoryCRQA* system (or simply CRQA), which stands for Crowd-powered Real-time Question Answering. CRQA integrates a crowdsourcing module into an automated question answering system within an overall learning-to-
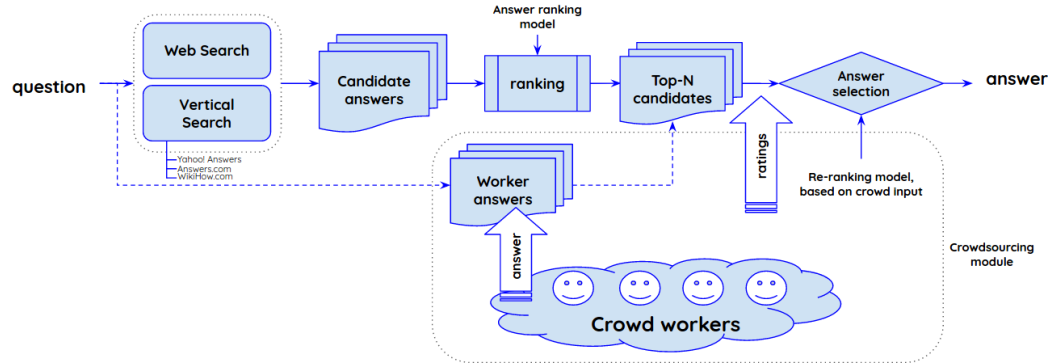
Figure 4.10: The architecture of our Crowd-powered Real-time Question Answering system, that uses crowdsourcing to augment a list of automatically extracted candidate answers and to rate their quality.

rank framework for selecting answers to complex questions. I report extensive experiments of stress-testing the CRQA system, by participating in the TREC LiveQA 2016 evaluation challenge, which provided a realistic evaluation setup.

The high-level architecture is presented in Figure 4.10. The automated part of the CRQA system is based on EmoryQA system, described in Section 4.1. The crowdsourcing module is designed to overcome two of the most common problems of the automated QA approaches: lack of good candidate answers and ranking errors. More particularly, CRQA asks crowd workers to provide answers to the given questions if they can, and additionally rate the quality of candidate answers, generated by the automated system. After the candidate answers are generated, instead of returning the final answer, as EmoryQA does, in CRQA we send the question and top-7 ranked candidates to the crowd workers and wait for the responses. I chose to give 7 answers based on the average number of rated answers per minute in the preliminary studies. Figure 4.11 presents the user interface of our crowdsourcing module.

The overall algorithm for obtaining crowd input for real-time question an-

Figure 4.11: User Interface for workers in our Crowd-Powered Question Answering system.

swering is the following:

1. When the system receives a question, it is posted to the workers, who will have 50 seconds to provide their input

2. Workers are asked to write an answer if they can provide one (it is optional)

3. Otherwise they are waiting for the answer candidates to arrive

4. When the system is done with generating and ranking candidates it posts top-7 scoring answers to the workers for the rating (which usually leaves $\sim$ 35 seconds for rating)

5. Workers receive a list of answers[21] and rate them until the timer runs off. Each answer is rated on a scale from 1 to 4, using the official TREC LiveQA rating scale:

    • 1 — Bad: contains no useful information
    • 2 — Fair: marginally useful information

---

[21]Answers submitted by workers are also sent for ratings to all workers except the author

- 3 — Good: partially answers the question
- 4 — Excellent: fully answers the question

6. The interface displays 3 answers at a time when an answer gets rated, it disappears and its place is taken by another answer from the pool. The interface displays only the first 300 characters of the answer, which was experimentally shown to be enough on average to make a good judgment. Full answer can be revealed upon clicking the "show all" link.

7. When the timer runs off, the question and all the answers disappear, and workers wait for the next question

Instructions

1. This HIT will last exactly 15 minutes

2. Your HIT will only be submitted after these 15 minutes

3. In this period of time, you will receive some questions, that came from real users on the Internet

4. Each question has a time limit after which it will disappear and you will need to want for the next one

5. If you know the answer to the question, please type it in the corresponding box

6. At some point, several candidate answers will appear at the bottom of the page

7. Please rate them on a scale from 1 (bad) to 4 (excellent)

8. Do not close the browser or reload the page as this will reset your assignment.

Table 4.4: EmoryCRQA crowdsourcing task instructions, displayed to the user when she first gets to the task.

To hire the workers we used Amazon Mechanical Turk platform[22]. Since

---

[22]http://mturk.com

the question answering system needs to provide a near real-time response whenever it receives a question, we adapted the "retainer" model for real-time crowdsourcing, inspired by the success of this model reported in previous works [27, 31]. Specifically for TREC LiveQA 2016 task, to obtain an even distribution of workers over the 24-hour period, we posted 10 tasks every 15 minutes, and they expired after the next set of tasks became available. Since not all assignments were accepted right away, the number of workers for each question varied and could be greater than 10. When a worker first gets to our crowdsourcing interface, she is shown task instructions (Table 4.4) and asked to wait for the questions to arrive. The workers were paid $1.00 for the whole 15 minutes task, no matter how many questions they got[23].

Answer-based

— The length of the answer

— Source of the answer (Crowd, Web, Yahoo! Answers, Answers.com or WikiHow.com)

— Original rank of the candidate answer or -1 for answers provided by the crowd workers

Worker ratings

— Number of ratings provided

— Minimum, maximum, median and average ratings

Table 4.5: The list of features used for answer re-ranking based on crowdsourcing input in EmoryCRQA question answering system.

The last stage in CRQA is answer re-ranking, which aggregates all the information received from the crowdsourcing and produces the final answer to the question. The input of the re-ranking module is a set of candidate

---

[23]In TREC LiveQA task questions are sent to the systems one by one, therefore there is no concurrency, however, the delays between the questions are possible.

answers with quality ratings provided by the crowd workers. Candidates can include the answers posted by the workers, which might also be rated if workers had enough time to do that. To re-rank the answers we trained a gradient boosting regression trees (GBRT) model [76]. To build this model we used a training set of questions with answers generated by our system. The quality of each answer was manually assessed using the official LiveQA scale from 1 (bad) to 4 (excellent). The features, used for answer re-ranking are listed in Table 4.5. CRQA sorts the candidates by the quality score predicted by the model, and returns the top candidate as the final answer.

4.2.3   Experiments

We now describe the experimental setup used to evaluate the performance of CRQA and other methods for near real-time question answering.

The experimental evaluation of our CRQA system was done on the official run of TREC LiveQA 2016 shared task, which happened on May 31, 2016. All participating systems were running for 24 hours and received questions sampled from the live (real-time) stream of questions, posted by real users to Yahoo! Answers platform. In total, each system received 1,088 questions, and system responses were recorded by the organizers.

Overall statistics are provided in Table 4.6. As we can see, on average workers were able to provide at least one answer for each question, and the provided answers got about 6 ratings.

The official TREC LiveQA 2016 evaluation results for our system are presented in Table 4.7. EmoryCRQA achieved the highest scores of all participating systems, and improved results of fully automatic setup (EmoryQA) by 20%. Unfortunately, there is a noticeable gap in performance between community responses and answers of our system.

To perform a full analysis of the system, we used traditional (batch-mode)

Name	Value
Number of questions received	1088
Number of completed assignments (15 mins each)	889
Average number of questions per assignment	11.44
Total cost per question	$0.81
Average number of answers provided by workers	1.25
Average number of ratings per answer	6.25

Table 4.6: Aggregate statistics of the crowdsourcing tasks submitted during TREC LiveQA 2016 shared task run.

crowdsourcing to obtain the quality labels for all answer candidates that were given to the workers during the task, as well as the answers provided by the workers. In addition, on June 2, two days after the TREC LiveQA challenge has completed, I crawled the current answers provided by the community for the questions, used for the task. All the answers for each question were randomly shuffled and rated on a scale from 1 (bad) to 4 (excellent) by workers hired on Amazon Mechanical Turk. As we have shown in the previous section, crowdsourced labels correlate well with the official ratings, provided by the professional NIST assessors. Each answer was labeled by 3 different workers, and we averaged the scores to get the final quality labels for the candidates.

We compared CRQA system against several baselines:

- *EmoryQA*: automated QA system described in Section 4.1.
- *Re-ranking by score*: a simplified version of the EmoryCRQA re-ranking model, which select the answer with the highest average ratings, provided by the crowd workers.
- *Yahoo Answers*: traditional, non-real-time community question answering site (Yahoo! Answers), from which the challenge question orig-

	avg score (0-3)	succ@2+	succ@3+	succ@4+	prec@2+	prec@3+	prec@4+
Results from TREC LiveQA 2016							
$HUMAN_{qual}$	1.561	0.655	0.530	0.375	0.855	0.692	0.490
$HUMAN_{speed}$	1.440	0.656	0.482	0.302	0.784	0.576	0.362
1. **EmoryCRQA**	1.260	0.620	0.421	0.220	0.644	0.438	0.228
- crowd-answers only	1.298*	0.206	0.128	0.072	0.659	0.410	0.230
2. CMU OAQA	1.155	0.561	0.395	0.199	0.596	0.420	0.212
3. EmoryQA	1.054	0.519	0.355	0.180	0.530	0.362	0.184
Avg results	0.643	0.329	0.212	0.104	0.422	0.271	0.131

Table 4.7: Official evaluation results of Emory CRQA system from TREC LiveQA 2016 shared task. Full results table is available in [3]. *Unlike other systems, average score for "crowd-answers only" experiment was normalized by the number of answers rather than questions.

> inated. The answers were collected two days after the challenge, thus allowing the Yahoo Answers community extra two days to collect the answers through traditional (community-based) crowdsourcing.

To evaluate the methods we used the metrics proposed by the organizers of the LiveQA task (Section 4.1.3). Table 4.8 summarizes the performance of the baselines and our system. As we can see, the average score and precision of answers generated by CRQA system are higher than the baseline ranking and even community answers on the Yahoo! Answers platform. However, Yahoo! Answers community answers have a higher percentage of *"4 (excellent)"* scores. Figure 4.12 shows the distribution of scores for the original system ranking, our crowdsourcing system and Yahoo! Answers. Two peaks on the distribution of scores from Yahoo! Answers community suggest, that there are essentially two kinds of responses: non-useful (*e.g.,* spam) or excellent that fully answers the question. In addition, around 20% of the questions did

Method	$\overline{score}$	$\overline{prec}$	s@2+	s@3+	p@2+	p@3+
EmoryQA	2.321	2.357	0.697	0.297	0.708	0.302
Re-ranking by score	2.416	2.421	0.745	0.319	0.747	0.320
Yahoo! Answers	2.229	2.503	0.656	0.375	0.737	**0.421**
EmoryCRQA	**2.550**	**2.556**	**0.799**	**0.402**	**0.800**	0.402
without worker answers	2.432	2.470	0.750	0.348	0.762	0.354
without worker ratings	2.459	2.463	0.759	0.354	0.760	0.355

Table 4.8: Evaluation of the baselines and system answers quality based on the ratings of answers obtained via crowdsourcing. The scores are averaged over 100 different 50:50 splits of 1088 questions into the training and test set. The differences between average score and precision of CRQA and the original ranking are significant at p-value < 0.01.

not get any answer from the community. Automatically generated answers, on the contrary, are rarely empty, but on average provide only marginally relevant information, which often does not answer the questions, and therefore rated *"2 (fair)"*. The introduction of the crowdsourcing module allowed CRQA to cover an additional couple of percents of the questions, for which the automated system was not able to generate any candidates, as well as select better candidates when it was possible using crowd ratings.

4.2.4 Analysis and Discussion

In this section, we will analyze some of the results of our experiments and discuss their implications.

Worker answers vs ratings. First, let's look at the contribution of additional answers and answer ratings provided by the workers. These two types of contributions are complimentary to each other and attempt to solve different problems. Table 4.8 shows the performance of our question an-
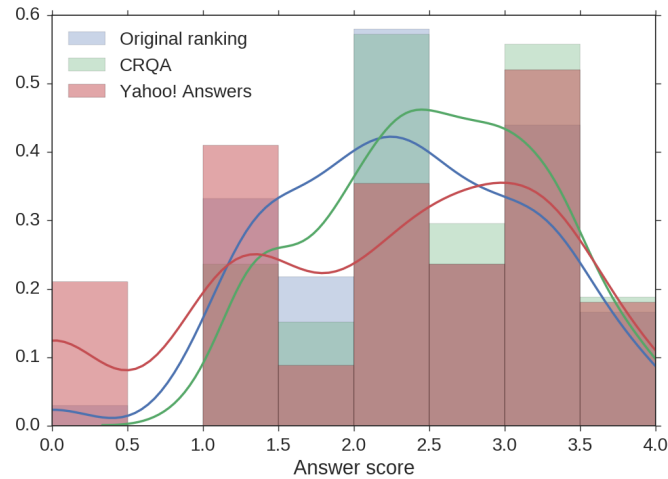
Figure 4.12: Histogram and kernel density estimation of answer scores for original candidate ranking, CPQA model re-ranking and Yahoo! Answers answers.

swering system using each of these types of feedback independently. The results demonstrate that both answers and ratings have a positive effect on the performance. Even with limited time, workers were able to reliably rate candidate answers, which helped the system to select a better final answer and improve the model precision. However, this method does not help the system in cases, when it was not able to generate a good candidate in the first place, therefore using ratings only has lower average answer score than using worker generated answers. By asking the crowd to provide a response if they can answer the question, CRQA covers this gap, which is important as in a real scenario even a fair answer would probably be better for the user than no answer at all. Of course, given limited time and the fact that a random worker might not possess an expertise required to answer the question, such answers do not always perfectly answer the question. We also looked at official evaluation results for answers, produced by the crowd and returned

as the final response by CRQA (Table 4.7). Crowd-generated answers accounted for $\sim 30\%$ of all system responses, and their average quality was 1.298, which is better than the quality of fully automatic EmoryQA system, but significantly lower than scores of community generated responses.

Table 4.9 gives some examples of worker-generated answers with low and high-quality scores.

Selection of answer candidate for rating. We have seen that crowd workers are able to provide reliable answer ratings, which can be used to re-rank them and select a better final response. However, since a system usual has hundreds or thousands of candidate passages, the capacity of crowd-sourcing is limited. We chose to show top-7 answers according to the trained learning-to-rank model, however, the order in which the answers are shown can also have a strong effect on the system performance, because the answers are typically rated one by one in the order they are displayed on the screen. Our system included two strategies for answer ordering: random or according to their ranking score. The former strategy provides a uniform coverage for all the answers selected for rating, while the later puts more emphasis on the currently top scoring candidates. We randomly selected one of the strategies for each user and question. To analyze the performance of each of the strategies we compute the average score of answers, generated using the corresponding ratings. The average score for answers generating when candidates are shuffled is 2.508, and it is 2.539 when the candidates are sorted according to their model ranking score. This suggests, that it is beneficial to allocate more of the worker's attention on the top scoring candidate answers.

Cost analysis. The results of our experiments clearly demonstrated that crowdsourcing can improve the performance of near real-time question answering system. The next reasonable question is what is the price of this improvement. In our study, we paid workers $1.00 per single 15 minutes task, and each 15 minutes we had 10 assignments, which translates to $15.00

Question	Answer	Score
Is Gotu Kola a good herb for mental health? How long does it take to work??	yes	1.66
Can I write any number on line number 5 of a W2? would like to set up my W2 were I get the most out of my paycheck...	W2	1.33
...i randomly asked my mother why when I lived with you in your home country a man that was our neighbour used to call me his daughter...?	yes	1.0
Is it bad not wanting to visit your family?	It is nt bad. Just be honest with them. They may be upset but they should understand	3.0
Any health concerns with whey protein?...	As long as you use it as directed, there should not be any major problems. You may want to consult your doctor just in case.	3.0
Foot pain unable to walk? Hi so today woke with some pain, I am able to put weight on my heel with no problem or pain...	Possible gout in your foot, also possible you may have strained it during the previous day.	3.0
What is a good remedy/medicine for stomach aches? Specifically ones caused by stress or anxiety?	Chamomile tea should help	3.66

Table 4.9: Examples of questions, answers and their quality scores, provided by the crowd workers during TREC LiveQA 2016 shared task.

(a) avg-score: Average score per question (b) avg-prec: Average score per answer
(ignoring non-answered questions)

Figure 4.13: Plot showing how the quality of the final answer depends on the number of workers per question.

per 15 minutes. Overall, our experiment cost $0.88 per question, and in this section, we will discuss some ideas to reduce this cost.

First, we will study the effect of the number of workers on the performance of our CRQA system. For this experiment, we randomly sampled a certain percentage of workers and removed all contributions (answers and ratings) of others. Figure 4.13 plots the dependency of the performance of our QA system on the number of workers.

Obviously, more workers mean more reliable answer ratings and more answer candidates, which improves the performance of the question answering system. However, we can observe diminishing returns: the cost per extra gain in performance metrics decreases as the number of workers grows. Half of the overall performance improvement could be achieved with only 3 workers per question, which would save 70% of the costs.

An alternative cost-reduction strategy is selective triggering of crowdsourcing, which would only ask for workers feedback for some of the questions. Such a strategy would be necessary to scale a crowd-powered question an-

swering system to a higher volume of questions. There are multiple different approaches for such selective crowdsourcing: *e.g.,* a system can only ask for crowd contributions if it did not generate enough candidate answers or the predicted quality of the top scoring candidates is low [49, 87]. We leave this questions for the future work, as here we focused on the scenario, proposed by the organizers of the TREC LiveQA shared tasks, where questions arrive one by one and it is possible to utilize crowd input for every question.

To summarize, in the explored real-time QA scenario it is possible to reduce the costs of crowdsourcing by reducing the number of workers, although with some performance losses. Our analysis suggests that paying 30% of the original cost would give 50% of the performance improvement.

4.3   Summary

This chapter described two QA systems aimed at answering a general class of user information needs, often referred to as non-factoid questions. A fully automatic *EmoryQA* system has information retrieval techniques at its basis, *i.e.,* it extracts relevant passages from multiple semi-structured and unstructured sources, represents them with a set of features, uses a learning-to-rank model to sort the candidate answers and selects the top one as the final answer. The system was experimentally tested on TREC LiveQA 2015 and 2016 shared tasks, achieving very competitive results. The source code of the system is available at https://github.com/emory-irlab/liveqa.

The analysis of the TREC LiveQA results revealed that automatic systems often have problems distinguishing between information, that is totally irrelevant and something, that might be of potential use to a user. As a result, compared to community generated responses, automatic QA systems have a higher fraction of answers, rated "bad". Many of these mistakes can easily be spotted by a human, even without a specific domain expertise.

EmoryCRQA extends the fully automatic QA system with a crowdsourcing module, which obtains feedback from a crowd of workers in a form of additional answer candidates and ratings of existing passages, while still operating in near real-time. Results of TREC LiveQA 2016 task confirmed the effectiveness of crowdsourcing in this scenario. The analysis presented in Section 4.2.4 shows that both worker contributed answers and ratings make an equal impact on the overall answer quality. The described CRQA implementation is a promising step towards the efficient and close integration of crowd work and automatic analysis for real-time question answering.

These contributions have laid the groundwork for future research in non-factoid question answering, using various types of semi-structured (QnA pairs), unstructured (web documents) and real-time crowdsourcing data. Together, the proposed methods enabled substantial performance improvements in answering a wider class of user information needs.

Chapter 5

Conversational Question Answering

Chapters 3 and 4 focused on improving the answer retrieval performance for different types of user information needs. However, question answering is not a one-way communication, where a user is only responsible for issuing requests and a system has to generate answers. This setup is quite limited because it does not provide any means for the user to affect the behavior of a question answering system except by issuing new questions. Similarly, it does not allow QA systems to request any additional information from the user, nor learn from the previous interactions.

Proliferation of mobile devices and more "natural" interfaces [89] are changing the way people search for information on the web. Many experts envision that search in the near future will be a dialog between a user and an intelligent assistant, rather than just "ten blue links" in response to a one-shot keyword query.[1] Participants of the SWIRL'2012 workshop foresaw a fusion of traditional IR and dialog systems [10]: "Dialogue would be initiated by the searcher and proactively by the system. The dialogue would be about

---

[1] http://time.com/google-now/

questions and answers, with the aim of refining the understanding of questions and improving the quality of answers." Today we can witness this trend embodied in such products as Amazon Alexa, Google Home, Microsoft Cortana, Apple Siri, and others.

This chapter presents the research towards designing better conversational search interfaces. Section 5.1 describes the user study we performed to learn how people use dialogs for information seeking scenarios and how the experience with modern personal assistants compares to a human-to-human dialog. Sections 5.2 and 5.3 focus on two particular interaction strategies, which allows a QA system to help the user either formulate or clarify their questions.

The contributions of the research described in this chapter are:

- A user study on conversational search, which provides actionable feedback on what people expect from such systems, how the expectations towards an automatic system differ from those towards humans, and what are some of the problems with using existing commercial personal assistants. This work will be presented at CHI 2017 conference [203].

- A study of the effect of strategic hints, that a system might provide to the user, on search experience and success rate for complex informational tasks. This work has been published as a short paper at SIGIR 2014 [163].

- An extensive analysis of clarification questions on community question answering platforms, and a model to predict the subject of a popular type of clarification questions, which shows the potential of such an approach. This results were published in CHIIR 2017 conference proceedings [39].

5.1 Conversational Search With Humans, Wizards, and Chatbots

Chatbots and conversational assistants are becoming increasingly popular. However, for information seeking scenarios, these systems still have very limited conversational abilities, and primarily serve as proxies to existing web search engines. In this section, we ask: what would conversational search look like with a truly intelligent assistant? To begin answering this question empirically, we conduct a user study, in which 21 participants are each given 3 information seeking tasks to solve using a text-based chat interface. To complete each task, participants conversed with three conversational agents: an existing commercial system, a human expert, and a *perceived* experimental automatic system, backed by a human "wizard" behind the curtain. The observations and insights of our study help us understand the aspirations of users and the limitations of the current conversational agents – and to sharpen a frontier of work required to improve conversational assistants for search scenarios.

5.1.1 Motivation

Personal assistants, such as Amazon's Alexa, Google Home, *etc.* are becoming increasingly popular, and people are integrating them in everyday life, *e.g.,* for simple tasks like setting up a timer, checking the calendar, requesting the latest news, a song, etc.[2]. The popularity of text-based chatbots is also on the rise in many areas of the web [71]. Most of them are template-based and are designed to fulfill a single, often monotonous, job [54, 64]. At the same time, a growing proportion of web search queries is formulated as natural language questions [16, 128, 144], which is partially explained by the increasing usage of

---

[2]https://arc.applause.com/2016/09/26/amazon-echo-alexa-use-cases/

voice interfaces [220]. Alas, for information seeking scenarios, existing chat-bots and intelligent assistants are usually implemented as simply a "proxy" to existing web search engines, even though question-answering technology has made dramatic progress handling such question-like queries [195]. Furthermore, conversation provides additional opportunities to improve search quality. For example, a conversational system should be able to ask clarification questions [39] to better identify searcher's intent, and incorporate explicit user feedback [153] – something that is not normally available in a traditional web search scenario. However, before jumping into implementing additional features for conversational search systems, it is important to gain a better understanding what the users' expectations are when interacting with a truly intelligent conversational search agent. It is equally important to anticipate how users might behave when faced with a conversational search system since behavioral feedback is critical for system evaluation and improvements. To this end, we explore the following research questions:

- **RQ1**: What are the main expectations from a conversational search system?
- **RQ2**: What are the differences between human-to-human and human-to-computer conversations?
- **RQ3**: What characteristics prevent existing conversational agents from becoming effective tools for complex information seeking?

As no truly intelligent conversational search systems exist yet, we explore these research questions with a mixture of survey methods and user studies. In the user study, the participants are faced with 3 complex information search tasks, derived from TREC Session track tasks [50]. To eliminate the voice recognition quality variable, we chose to use text messaging as the interface between a participant and conversational systems. We use three different conversational systems answering user requests: an existing com-

mercial intelligent assistant, a human expert and a human disguised as an automatic system. The results of our exploration suggest: (1) people do not have biases against automatic conversational systems, as long as their performance is acceptable; (2) existing conversational assistants are not yet up to the task, *i.e.,* they cannot be effectively used for complex information search tasks; (3) by addressing a few requests from users that we identified, even current search systems might be able to improve their effectiveness and usability, with feasible modifications.

5.1.2   Study design

We recruited 21 participants (graduate and undergraduate students at Emory University), to complete 3 different complex search tasks, taken from the TREC Session track 2014 [50] (Figure 5.1). The participants were asked to use an assigned text messenger-based conversational agent. They were not given any instructions on how to use the agent and therefore were free to interact with it in any way they chose. They were allowed to spend up to 10 minutes working on each task, after which they were asked to move on a topical quiz, consisting of 3 questions, designed for the topic. After seeing the topical quiz questions, the participants were not allowed to talk to the agent anymore. By doing so we ensured that the task stayed exploratory in nature, *i.e.,* the participants did not have a set of predefined points to cover. After completing a topical quiz, the participants filled out a preference questionnaire, where they were asked to rate their experience with the agent, provide feedback about advantages and disadvantages of the agent. After completing all tasks they filled out a final questionnaire. The communication was implemented through the Facebook Messenger interface[3]. Participants used a Facebook account created specifically for the purpose of the study.

---

[3]http://www.messenger.com

> **Topic 10:**
> Suppose you are writing an essay about a tax on "junk food". In your essay, you need to argue whether it's a good idea for a government to tax junk food and high-calorie snacks.

> **Topic 20:**
> You have decided that you want to reduce the use of air conditioning in your house. You've thought that if you could protect the roof being overly hot due to sun exposure, you could keep the house temperature low without the excessive use of air conditioning.

> **Topic 21:**
> Hydropower is considered one of the renewable sources of energy that could replace fossil fuels. Find information about the efficiency of hydropower, the technology behind it and any consequences building hydroelectric dams could have on the environment.

Figure 5.1: Description of the tasks used in the user study on conversational search. All the tasks were obtained from TREC Session track 2014 [50].

Message history was cleared prior to every experiment.

Wizard agent

Our first research question explores human behavior in human-computer communication. There are currently no general purpose intelligent conversational search systems, that we could use for our purposes. Therefore we "faked" one by substituting the backend with a person. However, the participants were told that it was an experimental *automatic* system, thus following

a general Wizard-of-Oz setup. We will be further referring to this system as the Wizard agent, and the person in the backend as the Wizard. The Wizard had done the research about the topics of the 3 tasks prior to the experiment and compiled a broad set of passages covering most of the aspects of each topic. At the time of the experiment, the Wizard tried to find the best passage to reply to the participant's question/comment. However, in cases where such passage could not be found, the Wizard would reply with a passage retrieved from web search, or write a new passage. In case the participant's question or comment was ambiguous, the Wizard was allowed to ask a clarification question to better identify the information need of the participant. Our Wizard agent was allowed to maintain the context of the conversation, respond to vague questions, understand implied concepts, and provide active feedback in form of clarification questions when needed (all of these capabilities do not yet exist in commercial systems). At the same time, by partially restricting the Wizard to a pre-compiled set of passages, we could maintain some consistency of answers between participants, *i.e.,* for the same question any participant would receive the same answer. By analyzing the ways the participants communicated with the Wizard agent, we could gain insights about strategies people use in a human-computer dialogue for solving complex tasks and look for design implications for automatic conversational systems.

Human agent

To answer our second research question, about the differences between human-to-human and human-to-computer communication, we devised our second conversational agent – the Human agent. In this case, the Wizard from the previous setup was still serving as a backend, but the participants were explicitly informed that they were talking to a live person. Another difference was that the Human agent was restricted to the pre-retrieved set of pas-

sages, but was free to slightly reformulate or revise the passages to better respond to the question. By including both the Human and Wizard agents in the study, we were able to maintain a constant level of intelligence for both agents, thus comparing not the accuracy of each agent, but rather the participants' attitude and expectations towards a perceived automatic agent compared to a known human.

Automatic agent

For a comparison with an existing conversational agent, we used the Google Assistant[4] as a backend for our third agent. Every message sent by a participant was forwarded to the Google Assistant app, and the response was forwarded back to the participant. Most of the time, the response consisted of an URL and a text snippet. The participants were told that they were interacting with another experimental conversational system, but were not given any specific information about it. By using a system representative of the state-of-the-art technology, we were able to evaluate its drawbacks, and situations where it failed to respond properly.

5.1.3 Results

After running the study, we analyzed message logs, answers to topical quizzes, and preference questionnaires and found the most popular trends and answers. This section describes our findings in detail.

Overall satisfaction

After completing each task participants rated their overall experience of working with each agent on a 1 to 5 Likert scale. Average ratings for each

---

[4]https://assistant.google.com/

Agent	Human	Wizard	Automatic
Overall satisfaction	4.1	3.8	2.9
Able to find information	1.5	1.3	1.0
Topical quiz success	1.6	1.6	1.3

Table 5.1: Statistics on user satisfaction and success rates with human, wizard and automatic agent in conversational search user study.

agent are shown in Table 5.1. The differences in ratings of Human vs. Automatic systems, and the Wizard vs. Automatic systems were statistically significant ($p < 0.0001$ and $p < 0.0005$ respectively), while the difference between the Human vs. Wizard systems was not significant. In the final questionnaire, after completing all the tasks, participants were asked which system they liked the most. Out of 21 people, 8 people preferred the Human agent, 6 – the Wizard agent, 4 – the Automatic agent, 2 people said they would use the Wizard or the Human depending on their goals, and 1 person said he would choose between the Human and the Automatic agent depending on her goals.

Able to find information

Participants were asked whether they were able to find all the information about the topic they were looking for. We coded each answer on a 0-2 scale (0 - no, I couldn't; 1 - partially; 2 - yes, I found everything I needed). Average results for each agent are shown in Table 5.1.

Topical quiz success

After completing each task participants were also asked 3 questions about the topic. We evaluated those questions on a scale 0-2, where 0 meant no answer, 1 - poor answer, 2 - good answer. On average, participants showed

a similar level of success with each agent. The average user ratings for each agent are shown in Table 5.1.

These results confirm our initial intuition that human-to-human conversation is more natural for the open-ended problem of the complex search task, compared with automatic conversational agents. This could be because people have experience talking to other people, and the results match their initial expectations. On the other hand, for any system that people have no experience with, they have to learn its functionality and ways to interact with it effectively.

We now turn to qualitative results, reporting the comments participants provided in the post-study questionnaire. The participants' comments broke down into the areas of maintaining context, trustworthiness, and social burden.

Maintaining context

Participant 19 (P19): "It didn't use contextual information so there was no way to expand on the previous answer it gave me." Within a conversation, people expect that the main topic of the discussion is maintained, and they tend to ask short questions, omitting the subject, or referring to the subject using pronouns. Formulating a full question takes effort and is unnatural. For the Automatic system, anaphora resolution did not always work, which annoyed the participants. Similarly, when dealing with the Human and Wizard systems, participants pointed out the ease of use, because their partially stated questions were understood, and relevant answers were returned (Figure 5.2).

Figure 5.2: Automatic system (gray background) fails to maintain context, which causes the participant 15 (blue background) to reformulate his question twice.

> I am not an expert, but I read some materials on the topic, so I can try to answer whatever questions you have.

> I'd rather use a web search then 😉

Figure 5.3: A participant prefers web search to talking to a person. Part of a conversation between participant 7 (blue background) and Human agent (gray background).

Trustworthiness of the sources is crucial

P7: "I ... like to be able to verify the credibility of the sources used." Even though the Automatic system did not always respond with a relevant result, it received approval from our participants for providing sources of its answers. Out of 21 participants, 13 people said that being able to access the URL allowed them to assess the trustworthiness of the source and therefore to accept or reject the answer. On the other hand, in spite the Human and Wizard systems returning more relevant results, they were both criticized for not providing the sources (Figure 5.3).

Social burden

P15: "you have to think about social norms, asking too much, being too stupid, not giving them enough time to respond, troubling them." When dealing with the Human system, 4/22 participants reported that they felt uncomfortable talking to a person, thought more about the social norms, were afraid to ask too many questions, were not sure how to start and end a conversation. This additional burden of interacting with humans further motivates research in the area of automated conversational agents as the medium of choice for a notable fraction of use cases (Figure 5.4).
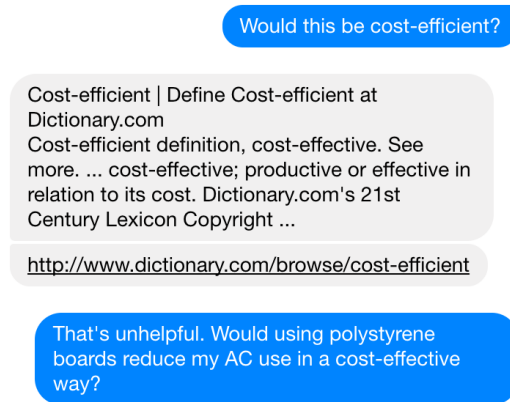
Figure 5.4: Explicit user feedback could be used to recover from failure. Part of a conversation between participant 12 (blue background) and Automatic system (gray background).

5.1.4 Discussion and design implications

Based on our findings we devised a list of recommendations for a conversational agent design, that according to our empirical study will improve user experience significantly.

Context. Maintaining a context of the conversation to enable short questions and comments is crucial to user experience since formulating long sentences each time feels unnatural and takes longer.

Provide sources of answers. Finding relevant and precise answers is important. But trustworthy sources are equally important, and their absence may diminish the credibility of the system. While the Automatic agent supported each answer with an URL, Human and Wizard did not, unless specifically asked.

Use feedback. One crucial difference of conversational setup from web search is the ability of a user to provide the system with explicit feedback. It is likely to contain essential information that may help the system to get back up from failure and improve upon the previous result.

Opinion aggregation. According to the participants, sometimes what is needed is the *experience* of other people in similar situations. A good conversational system should be able to aggregate opinions and present them to the user in a short summary, perhaps explaining each one. Participant 17 said: *"It would be nice if I could see a summarization of different opinions that there exist – from different sources."*

Direct answers vs. expanded information. For this aspect, our participants split into 2 camps: those who prefer getting direct answers to the question provided, and those who prefer also getting a broader context. People from Camp 1 complained that the answers returned by the systems were too long (even for the Wizard and Human), and preferred to have their questions answered directly with minimum extra information. Camp 2, on the other hand, said that they prefer talking to a person, who would recognize their true information need (beyond the immediate question) and provide the relevant information.

In this section, we investigated human behavior when using conversational systems for complex information seeking tasks. We also compared participant behavior when talking to a human expert, vs. a perceived automatic system. We observed that people do not have biases against automatic systems, and are glad to use them as long as their expectations about accuracy were met. However, existing agents often fail to provide a reasonable response, and users often struggle with finding the right way to ask or reformulate a question. In the next section, we will investigate search hints, which a system can provide to its users in order to help them solve complex informational search tasks.

5.2   Search Hints for Complex Informational Tasks

Some informational needs are more complex than others. While existing technologies can handle relatively simple questions pretty well, they might leave users frustrated with their responses for more difficult requests. Bilal and Kirby [32] reported that about half of the participants of their user study felt frustration when searching. Xie and Cool [225] demonstrated that most of the time users have problems with formulating and refining search queries. Besides good retrieval performance, a successful search requires users to possess certain skills. Search skills can be trained, *e.g.,* Google offers a course[5] on improving search efficiency. Although very useful, such courses are time-consuming and detached from real search problems of these particular users. Displaying search hints is another technique that has both learning effect and offers immediate assistance to the user in solving her current search task. Moraveji et al. [140] demonstrated that hints, suggesting certain search engine functionality, help people find answers more quickly, and the effect is retained after a week without hints.

In this section, I explore *strategic* search hints, that are designed to guide a user in solving her search problem. More specifically, for complex search tasks users might find it helpful to choose the divide-and-conquer strategy, *i.e.,* splitting an original difficult question into smaller problems, searching answers to the subtasks and combining them together. Two sets of strategic hints were manually designed: *generic* hints describing the divide-and-conquer strategy in general and *task-specific* hints providing a concrete strategy to solve the current search task. To evaluate the effect of the hints on behavior and search success we conducted a user study with 90 participants. The results of the user study demonstrate that well-designed task-specific

---

[5]http://www.powersearchingwithgoogle.com
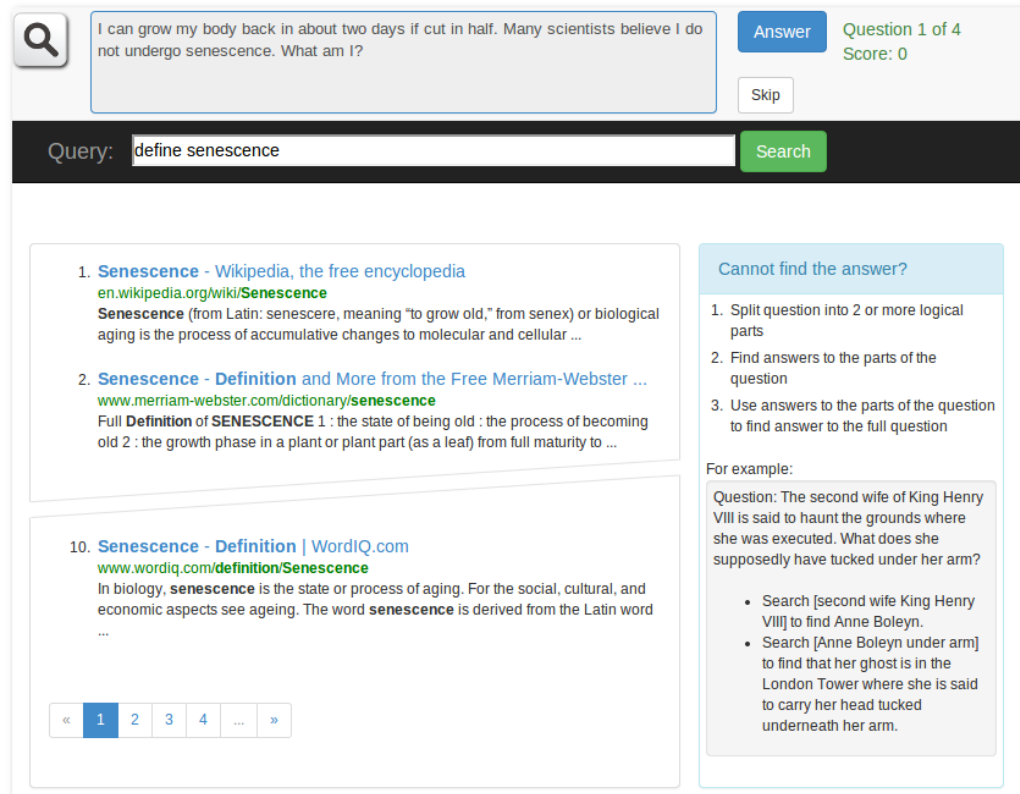
Figure 5.5: The interface of the search game used in the study of the effect of strategic search hints on success in solving complex informational tasks.

hints can improve search success rate. In contrast, generic search hints, which were too general and harder to follow, may have the negative effect on user performance and satisfaction.

5.2.1   User Study

To estimate the effect of strategic search hints on user behavior we conducted a study in a form of a web search game similar to "a Google a Day"[6] and

---

[6]http://www.agoogleaday.com/

	Question	Correct Answer	Specific hints
1	I can grow body back in about two days if cut in half. Many scientists think I do not undergo senescence. What am I?	Senescence means "biological aging". Hydra is considered biologically immortal and regenerates fast.	1. Find what is senescence 2. Find who does not undergo senescence 3. Find who can also regenerate body and choose the one that satisfies both conditions
2	Of the Romans "group of three" gods in the Archaic Triad, which one did not have a Greek counterpart?	Archaic Triad includes Jupiter, Mars, and Quirinus. Among those Quirinus did not have a Greek counterpart.	1. Find the names of the gods from the Archaic triad 2. For each of the gods find a Greek counterpart
3	As George surveyed the "waterless place", he unearthed some very important eggs of what animal?	"Gobi" in Mongolian means "Waterless place". The first whole dinosaur eggs were discovered there in 1923.	1. Find what is the "waterless place" mentioned in the question? 2. Search for important eggs discovery in this "waterless place"
4	If you were in the basin of the Somme River at summers end in 1918, what language would you have had to speak to understand coded British communications?	Cherokee served as code talkers in the Second Battle of the Somme.	1. Find the name of the battle mentioned in the questions 2. Search for which coded communications language was used in this battle

Table 5.2: Search tasks and specific search hints used for user study on the effectiveness of strategic hints for complex informational search tasks.

uFindIt [1]. Participants were hired using Amazon Mechanical Turk[7].

The goal of the web search game was to find answers to several questions with the provided web search interface (Figure 5.5). Players are instructed not to use any external tools. The questions are given one by one and since tasks might be too difficult, a chance to skip a question was provided, although users were instructed that effort put into solving a question will be evaluated. To answer a question each player needs to provide a link to a page containing the answer as well as its text. The answer is automatically verified and a pop-up box notifies a player if the answer is incorrect (since the answer can be formulated differently, the presence of a keyword was checked). A player can then continue searching or skip the question when she gives up. A bonus payment was made to players who answer all questions correctly. We used Bing Search API[8] as a back-end of the game search interface. All search results and clicked documents were cached so users asking the same query or clicking the same page got the same results. At the end of the game, a questionnaire was presented asking for feedback on user satisfaction with the game, prior experience, and other comments.

The tasks for the study were borrowed from the "A Google a Day" questions archive. Such questions are factual, not ambiguous and usually hard to find the answer to a single query, which makes them interesting for user assistance research. We filtered search results to exclude all pages that discuss solutions to "A Google a Day" puzzles. To do this we removed pages that mention a major part of the search question or "a google a day" phrase. To keep users focused throughout the whole game we limited the number of questions to 4. The tasks are described in Table 5.2 and were presented to all participants in the same order to ensure comparable learning effects.

The questions have multiple parts and to solve them it is helpful to search

---

[7]http://www.mturk.com/

[8]https://www.microsoft.com/cognitive-services/en-us/bing-web-search-api

for answers to parts of the questions and then combine them. In one of the previous studies, we observed, that most of the users did not adopt the divide-and-conquer strategy, but kept trying to find the "right" query. We decided to estimate the effect of strategic search hints, suggesting users to adopt the new strategy.

We built 2 sets of strategic hints: *task specific* and *generic*. Task-specific hints described steps of one of the possible solutions to each question (Table 5.2). The second set contained a single hint, which was shown for all tasks. Generic hint described the divide-and-conquer strategy:

1. Split the question into 2 or more logical parts
2. Find answers to the parts of the question
3. Use answers to the parts of the question to find answer to the full question

For example, the question: "The second wife of King Henry VIII is said to haunt the grounds where she was executed. What does she supposedly have tucked under her arm?"

1. Search [second wife King Henry VIII] to find Anne Boleyn.
2. Search [Anne Boleyn under arm] to find that her ghost is in the London Tower where she is said to carry her head tucked underneath her arm.

To control for the learning effect demonstrated in [140], each user was assigned to one of the three groups: (1) users who did not get any hints; (2) users who got task-specific hints; (3) users who got the generic hints.

5.2.2 Results and Discussion

From 199 unique participants, who clicked the HIT on Amazon Mechanical Turk only 90 players finished the game. We further examined all games

manually and filtered out 9 submissions for one of the following reasons: lack of effort (*e.g.,* skipped several tasks after none or a single query) or usage of external resources (*e.g.,* the answer was obtained without submitting any queries or results explored did not contain the answer). Furthermore, 10 players from the group which received hints indicated in the survey that they did not see them, so we filtered out those submissions and finally, we had 71 completed games (29 for no hints, 20 for task-specific hints and 22 for generic hints groups).

Effects of Search Tips on Performance. In order to measure search success rate we looked at the number of questions answered correctly by different groups of users[9]. Figure 5.6a shows that success rate is higher for users who saw task-specific hints compared to users who did not get such assistance. Surprisingly, having the generic hint decreased the success rate, although users could easily ignore a hint they did not like. A possible explanation is: generic hints were harder to follow and users who tried and failed became frustrated and did not restart their searches.

The plot of average time to answer a question on Figure 5.6b does not show an improvement for the task-specific hints group, except for the question 1. Our task-specific hints represent a possible way to solve a problem and there is no guarantee, that it is the fastest one. It is worth noting, that users from the generic search hint group had slightly higher variance in success time, which can probably be explained by the fact that some users were successful in finding the right way to follow the hint and some other users struggled with it much longer. Another insight comes from the number of incorrect attempts users made. Figure 5.6c demonstrates the average number of incorrect answer attempts for all groups of users. Although the variance is high, there is a tendency for users who saw task-specific hints to make fewer

---

[9]Since users were allowed to skip a question we are counting the number of questions that were eventually solved correctly even if a player made some incorrect attempts.

(a) Success rate per task for each group of partici- pants (b) Task completion time for each group of players (c) The number of incorrect submission attempts per question for all groups of users
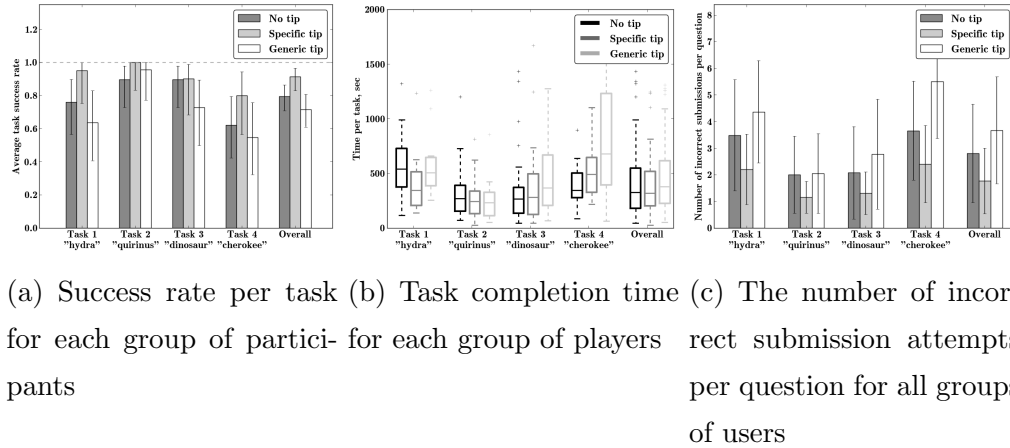
Figure 5.6: Results of the user study on the effectiveness of strategic search tips on search task success rate.

attempts than both other groups. This is not in direct correspondence with time spent on the game. It seems that the users who saw a clear strategy to solve the question were less likely to notice plausible, but incorrect solution. Moreover, we analyzed texts of incorrect answers and can conclude that a big part of incorrect submission are due to users trying all possible options they found on the way, even if these options are clearly wrong.

(a) How did you like the game? (b) How difficult was the game? (c) Were search hints use- ful to you?
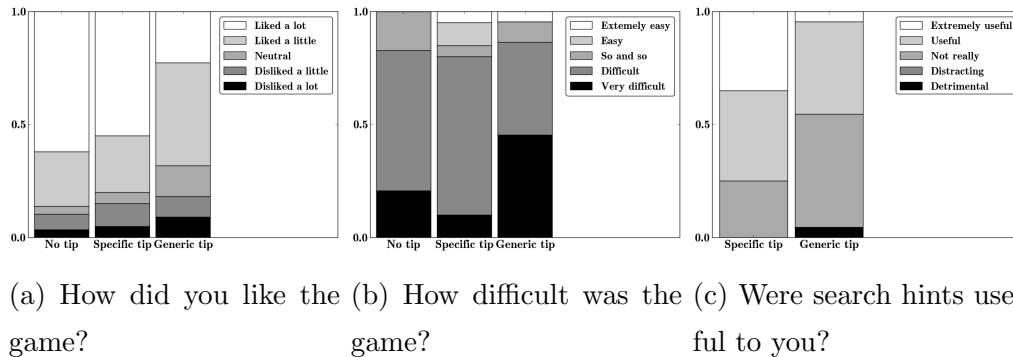
Figure 5.7: Proportions of replies to some of the survey question for each group of users.

We also looked at other search behavior characteristics: number of queries submitted, the number of clicks made, the average length of the queries. The variance in these characteristics was too high to make any speculations regarding their meaning.

Effects of Search Tips on User Experience. Finally, we looked at the surveys filled out by each group of users. Figure 5.7 presents proportions of different answers to three of the questions: "How did you like the game?", "How difficult was the game?" and "Were search hints useful to you?". Surprisingly, user satisfaction with the game was lower for users who saw hints during the game and users who did not get any assistance enjoyed it more. The replies to the question about game difficulty are in agreement with the success rate: users who saw task-specific hints rated difficulty lower than participants who struggled to find the correct answers. The game was very difficult on average, however, some participants from the group who received task-specific hints surprisingly rated it as very easy, which suggests that our hints do help users. This is supported by the answers to the last question on whether hints were helpful (Figure 5.7c).

Summary

In this section, we studied the effect of strategic search hints on user behavior. The conducted user study in a form of a web search game demonstrated the potential of good hints in improving search success rate. However, to be useful, they should be designed carefully. Search hints that are too general can be detrimental to search success. We also find that even searchers who are more effective using specific search hints, feel subjectively less satisfied and engaged than the control group, indicating that search assistance has to be specific and timely if it is to improve the searcher experience.

In addition to providing users with some hints on how to continue their search process, it is important for a QA system to improve question under-

standing skills. Unfortunately, some questions contain certain ambiguity. In conversational settings, a natural thing to do for such requests is to ask a clarification question, which is the focus of the next section of this dissertation.

5.3   Clarifications in Conversational Question Answering

One key capability required to make conversational question answering effective is asking *clarification questions* (CLARQ) proactively, when a user's intent is not clear, which could help the system provide more useful responses. With this in mind, we make the first attempt to examine the clarification questions (CLARQ) that users ask on the Stack Exchange community question answering (CQA) platform. We analyze Stack Exchange data in two domains corresponding to about 300K questions and comments. The contributions of this study are threefold:

- To learn about user behavior associated with CLARQ and about their role in CQA communications. We find that CLARQ are quite common on Stack Exchange, and therefore represent a good source of data for analysis.
- To study the types of CLARQ users ask in different situations. We classify clarification questions into several categories according to their target as well as syntactic patterns, which help define the space of CLARQ for future research;
- To make the first step towards automatic generation of CLARQ: we build a model to predict the subject of a popular type of clarification questions, which shows the potential of such approach for future research.

cover a period of 5.5 years – from July 2010 to January 2016.

We define CLARQ in a straightforward manner: sentences in comments to the initial questions ending with the question mark, provided by the users different from the asker of the initial question, four words and longer. This heuristic is not perfect, as clarification requests can be formulated as a declarative sentence, *e.g., Please provide details...* or question mark can be just missed. At the same time, these interrogative comments may be rhetorical questions, or not on the initial question's subject. Nevertheless, manual inspection showed that this definition of CLARQ is operational and allows extraction of CLARQ with precision acceptable for an exploratory study.

Basic statistics of the two datasets are reported in Table 5.3.

	DIY	*GAMES*
# of questions	20,702	62,511
# of answers	36,580	105,167
# of accepted answers	8,381	40,049
# of comments	87,238	228,074
average question length in words	130.8	86.5
average comment length in words	33.8	25.8
# of comments on questions	37,296	96,247
# of non-asker comments on questions	27,873	72,495
# of comments on questions with '?'	11,040	21,448
CLARQ followed by asker's comments	3,679	8,021
CLARQ followed by post editing	4,270	9,038
CLARQ followed by post editing by asker	1,631	3,772

Table 5.3: Statistics of the Stack Exchange datasets, used for the clarification questions study.

5.3.2   Results

User behavior

As Table 5.3 shows, the presence of CLARQ in CQA is substantial. Many characteristics such as questions/answers, accepted answers/all answers ratios are similar for both domains. Questions and comments in DIY are longer then in GAMES, which is expected: DIY implies richer and more diverse contexts. Askers are engaged in communication even after the initial question is posted: they comment on questions and edit them (however, questions are edited by community members more often). Although there are more comments on questions in DIY, GAMES seem to be somewhat more conversational: askers respond to questions on questions more often. Interestingly, thousands of initial questions are followed by clarifications, and in many cases, these are followed by the original question being edited, presumably in response to the clarification request.

Unfortunately, we see that questions followed by CLARQ do not differ much from questions without any comments in length – a simple assumption that CLARQ are targeted at short underspecified questions does not hold. The hypothesis that questions asked by novice and less experienced community members (based on users' ratings) receive more CLARQ is not supported either. We also did not find any topical specificity of questions with CLARQ based on tags.

Figure 5.9 shows rating distribution of the GAMES users, who ask CLARQ and those who provide accepted answers (i.e., the best answerers in the community). We can observe that distribution for the former group is shifted towards higher scores (DIY exhibits a very similar distribution). However, the users who ask for clarifications provide answers for the initial questions very rarely. This observation suggests that CLARQ in CQA are a form of 'quality control measures' undertaken by most experienced users.
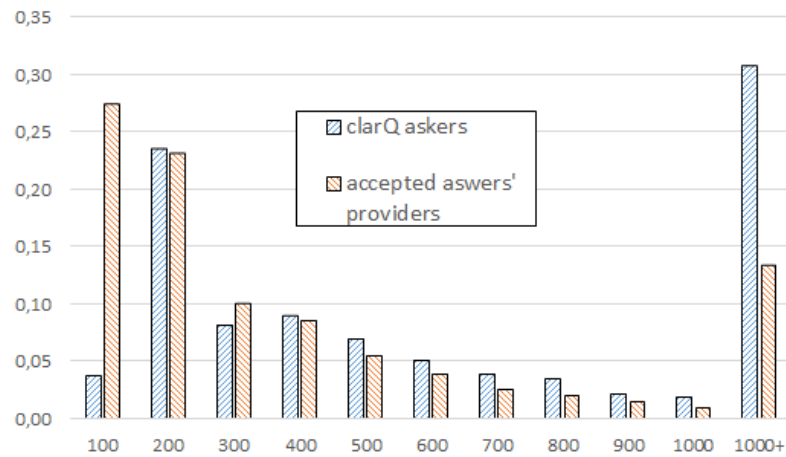
Figure 5.9: Distribution of users' reputation scores in the groups of accepted answers' providers and commentators on questions (GAMES).

Question types and patterns

Category	%	Example
More Info	28.6	What OS are you using?
Check	29.3	Are you on a 64-bit system?
Reason	8.5	What is the reason you want a drip pan?
General	10.2	Can you add more details to this question?
Selection	9.9	Are you using latex or oil based Kilz?
Experience	10.2	Have you tried to update video card drivers?
Not a CLARQ	13.9	Out of curiosity what elo are you?

Table 5.4: The distribution of questions in StackExchange comments by type. Some comments contain several CLARQ of different types, so the sum is more the 100%.

In order to investigate CLARQ breakdown by type, we sampled 294 comments on questions from both domains, and two authors performed manual

Pattern	DIY	GAMES	Example
have you tried	256	1,123	Have you tried reinstalling the game yet?
do you have	592	692	Do you have enough disk space left?
do you mean	248	552	Do you mean a separate tub and shower?
are you sure	206	366	Are you sure you have timber frame construction?
what (is—are) the	558	361	What is the slope of the floor?
what do you	103	284	What do you mean by squeaking?
(are—is) there any	154	147	Are there any airflow ducts in the room already?
can you (post—provide)	204	125	can you post some pictures?
how X (is—are)	290	117	how old is the water heater?
is it a	186	112	is it a constant 18-22 fps?
what (kind—type) of	344	106	What kind of texture is on the wallpaper?
why do you	73	101	Why do you need to run it from the Flash drive?
have you checked	66	98	Have you checked the frequency of the outlets?
is it possible	78	84	Is it possible the tank is just over filling?
do you know	120	64	Do you know the manufacturer of the fixture?

Table 5.5: Question patterns in comments (sorted by frequency in GAMES) from StackExchange dataset.

annotation analogously to [109], see results in Table 5.4. Comments that are not aimed at clarifying the main question contain rhetorical or humorous questions, questions to previous comments, citations of other questions on the platform, etc. Interestingly, the breakdown of CLARQ by type is roughly the same as in [109].

Further, upon examining the sample we identified a list of common three-word question starting patterns, and calculated their frequency in the whole dataset, see Table 5.5. As can be seen from the Table 5.5, some patterns are quite indicative for the clarification type (*e.g., what kind of* corresponds to *More info* category, whereas *have you tried* – to *Experience*). This observation suggests that recognition of clarification question type is a feasible task.

Clarification subject prediction

As we have shown, there are many different kinds of questions that users ask in comments. Many of them address a certain ambiguity present in questions, *e.g., what kind of* questions inquire about a subtype of a mentioned object. These questions are quite common (Table 5.5) and have a simple structure, which makes them a quite appealing target for automatic question generation. The first step for such question generation is to predict the object to ask about. We collected questions, which received at least one *what (kind—type) of* CLARQ in DIY. From these comments and questions, we extracted noun phrases using Stanford CoreNLP parser [135], and kept only those questions that have a common pair of noun phrases in the question and comment. We formulated the task as the noun phrase ranking problem, where the noun phrase from the comment should be placed higher on the list than other noun phrases from the question. Each candidate phrase was represented by the following set of features:

- **prior**: number of times the noun phrase was used in comments (sepa-

rate from the training and test sets)

- **topicality**: number of occurrences of the phrase in the current question (in title and body together)
- **position**: position of the first occurrence of the noun phrase relative to the beginning of the question
- **entropy**: collection-based statistic, computed using all noun phrases that contain the given noun phrase, which estimates the number of different modifications of the current noun phrase object
- **length**: number of words in the noun phrase

To train the ranking model we used a random forest algorithm implemented in the RankLib library[13]. We optimized DCG@10 and Table 5.6 summarizes the performance metrics on 10-fold cross validation. As we can see, even with a limited set of features our model was able to place the true subject of a clarification question above other candidates in 35% of the cases. To study the contributions of different feature groups we conducted a series of experiments to train the model with each group individually. The results in Table 5.6 suggest, that the number of occurrences of a phrase and the position of the first occurrence are strong features, and confirms our intuition that CLARQ are usually asked about the main topic of the question. However, some noun phrases are more ambiguous in general, therefore the prior feature also contributed significantly to the quality of the model.

Overall, our experiment showed promising results for predicting the subject for a certain type of CLARQ. As a next step, our model can be combined with an ambiguity detection classifier, which would trigger clarification as a response from a conversational search agent.

---

[13]https://sourceforge.net/p/lemur/wiki/RankLib

Model	P@1	MAP	RR@10	ERR@10
random	0.077	0.215	0.231	0.015
+ entropy	0.143	0.334	0.350	0.024
+ length	0.148	0.337	0.345	0.024
+ position	0.165	0.335	0.357	0.024
+ prior	0.214	0.402	0.427	0.030
+ topicality	0.319	0.426	0.473	0.032
all features	0.350	0.508	0.549	0.038

Table 5.6: Performance metrics (P@1 – precision at 1, MAP – mean average precision, RR@10 – reciprocal rank at 10, ERR@10 – expected reciprocal rank) of the ranking model for "ambiguous" noun phrase selection problem.

5.3.3 Discussion

As a step towards general-purpose interactive QA system, we analyzed clarification questions asked by CQA users. In particular, we examined user interactions related to CLARQ, as well as the role and place of these questions in CQA. We analyzed a large sample of CLARQ according to their type and identified most common syntactic patterns in a large collection of CLARQ. Finally, we conducted an experiment aimed at automatically detecting the subject of clarification question of a particular type.

Based on our analyses, we can conclude that CLARQ are common in CQA archives, and introduce a valuable resource for user behavior studies and QA research. Clarification questions asked by community members are an important component in maintaining the quality of user-generated content in CQA. Furthermore, we see that clarification questions are quite diverse in topic and style, are highly dependent on context and individual characteristics of the users. However, there are several types of questions and syntactic patterns that are common within each domain. As a first step towards au-

tomatically generating clarification questions, we show promising results on identifying the subject of CLARQ based on a small set of shallow features. Our findings suggest that CQA data may be useful for research in the field of interactive QA.

5.4 Summary

This Chapter described the research I have done on conversational question answering, which gives a system opportunities to exploit a rich set of possible interactions with its users. First, I described the design and implications of the user study we conducted to investigate the current state in conversational search, and looked into what users expect from a dialog agent compared to a human interlocutor, and what is missing from an existing state-of-the-art commercial system. We learned, that users do not have any prejudice towards automatic question answering agents, and for some people they are actually preferable, avoiding certain social norm issues. We identified some directions for future research in order to move existing systems towards user expectations, by providing a diverse set of opinions as well as information sources, improving context maintenance techniques, and learning from user feedback. As the first steps in these directions, I discussed search hints and clarification questions, which can be provided by the system to either help users structure their search process or clarify some of the ambiguities. The results of the conducted research showed that hints tailored to a specific search problem can effectively guide the user through the search process. However, generic hints might actually take away the feeling of control from the user and lower both satisfaction and success rate. To study the phenomenon of clarification questions, we analyzed the data from one of the community question answering platforms and identified different types of clarifications, *e.g.,* "what type of ..." questions, that aim at requesting information on a

particular type of an object mentioned in the question. The model we built to predict the subject of such questions showed a reasonable performance, and demonstrate the potential of this approach for automatic clarification generation.

Together, the research described in this chapter provides desiderata for future developments in conversational search and question answering and shows promising first steps in some of these directions.

Chapter 6

Conclusions

This chapter concludes the dissertation by providing the summary of the findings, limitations, potential future directions, and the main contributions of my work.

6.1   Summary of the Results

In the era of information overload, we have to rely on intelligent systems to help us organize and search the knowledge about the world. The research described in this dissertation aims to improve the technology behind question answering systems, which can sift through the constantly growing piles of information and give the user the needed response. In the next sections, I will describe the main results of my research, and how it helps to get closer to the above-mentioned goal.

6.1.1 Combining KB and Text Data for Factoid Question Answering

When we type a question like "*What is the capital of Peru?*" into a favorite search engine, we can expect to see a direct answer shown on top of search results. However, for the majority of more complex tail questions users still have to browse through retrieved web documents and search for the answer themselves. My Ph.D. research targets the problem of improving both precision and recall of factoid question answering by combining available data sources. Chapter 3 described 3 different approaches to this problem.

Relation extraction became a common tool for transforming knowledge from one format to another, *e.g.,* from natural language text into structured knowledge base triples. Existing approaches target some particular subsets of data, *e.g.,* facts expressed in statements, such as "*The capital of Peru is Lima*". The proposed model for relation extraction from community generated question-answer pairs (Section 3.1) provides an extension of existing techniques to a new domain and helps to extract additional factual triples, therefore increasing the coverage of knowledge bases. Our experiments on Yahoo! Answers and WikiAnswers datasets suggest, that by adding the proposed model, which focuses on entity pairs mentioned in the question and answer texts, we can extract $\approx 20\%...40\%$ more triples combined with existing sentence-based techniques. This means, that using the relation extraction model proposed in my Ph.D. dissertation, from the same data source we will be able to extract more factual knowledge, therefore significantly improving the recall of the relation extraction methods. Extracted triples can be further injected into a knowledge base, *e.g.,* using approaches like Google Knowledge Vault [62], eventually leading to more answered user questions.

Besides increasing the coverage of underlying data, the dissertation proposes a set of techniques to improve the performance of the core question

answering pipeline. *Text2KB* model, described in Section 3.2, brings the power of text-based question answering to KBQA to improve question interpretation, candidate generation and ranking. By identifying mentions of knowledge base concepts in text documents it is possible to use text matching techniques to understand the question topic, relate question terms to KB predicates and better rank the generated answer candidates. The experiments conducted on the WebQuestions benchmark dataset demonstrated, that the proposed techniques can improve the performance of a state-of-the-art KBQA system by $\approx 8\%$. I believe, that this work makes progress in one of the key directions for improving QA systems: integration of different data sources for joint reasoning, and we can expect more researches to be done in this area in the upcoming years.

Finally, the dissertation proposes the *EviNets* neural network architecture, which can aggregate answers supporting evidence from different sources, including text and knowledge base data. By embedding text and KB triples into the same space, the model can estimate the relevance of each statement to the question, and score candidate answers based on all available evidence. The experiments performed on TREC QA, WikiMovies and new developed Yahoo! Answers datasets confirm these expectations and demonstrate that the model indeed can combine KB and text data in a unified framework, improving the performance over existing baseline approaches. EviNets makes the next step from relevance matching techniques towards systems, that make decisions based on all available information. By demonstrating the effectiveness of evidence aggregation for QA performance, I believe this work will motivate future research on designing better methods for combining various sources of information for joint reasoning.

Combined, these techniques allow us to achieve both higher precision and recall in factoid question answering.

6.1.2 Ranking Answer Passages for Non-factoid Question Answering

The challenges in non-factoid question answering are due to a diverse nature and types of these questions and answers. During my Ph.D. studies, I developed a system, that combines vertical search in CQA archives and general web search to retrieve similar question-answer pairs and text passages from relevant documents. The system achieves state-of-the-art performance in TREC LiveQA 2015 and 2016 shared tasks and can be used as a baseline for future experiments. According to the results of TREC LiveQA 2016, for more than half of the questions the system was able to return a relevant response, and for $\approx 20\%$ of the questions it gave a perfect answer. The system is open sourced, and can be used as a baseline for future research on improving the state-of-the-art in non-factoid QA.

The crowdsourcing module, integrated into the *EmoryCRQA* system for obtaining additional answer candidates, and rating of existing ones, allowed us to significantly boost the performance of the fully automatic QA system. This hybrid approach achieved the highest score among participants of the TREC LiveQA 2016 shared task, with the average answer score of 1.260 on the 1-4 Likert scale. This score is only within $\approx 20\%$ of the quality of the community-generated response, which was obtained a week after. The experiment demonstrates, that it is possible to use a crowd of non-expert workers to obtain additional feedback and relevance judgments in real-time, and these signals can significantly improve the performance of an automatic QA system. With crowdsourcing, our system was able to generate a reasonable response to more than 60% of the questions, compared to only 50% for the fully automatic setup. The crowdsourcing expenses can be reduced by limiting the number of workers per single task and selectively requesting feedback for more complex questions only. These findings can be useful

for building hybrid question answering systems, that would rely on human feedback to improve the performance on difficult tasks.

6.1.3 Question Answering in Conversational Setting

While gaining some popularity, personal assistants like Amazon Alexa, Google Home, Microsoft Cortana and Apple Siri, are still mostly used for simple routine tasks, like playing music and checking the news and weather. In Section 5.1 we described a user study, designed to learn what is missing from commercial products to be the major tool for informational tasks. The findings of the study suggest, that users still often prefer search engines because they can offer a variety of information with sources, and give more control over what information a user consumes. The abilities of existing personal assistants often do not allow natural ways of forming questions, like a person would do in a conversation with an expert. For example, the commercial chatbot we tested could not properly maintain the context of the conversation, *e.g.,* resolve pronouns to the previously mentioned topics. However, the participants of the user study expressed their interest in such systems, as they allow to shortcut some social rules and get straight into the information finding while providing concise responses.

In Sections 5.2 and 5.3 the dissertation describes some of the first steps in the discovered directions. For complex search tasks, it is quite important to learn to formulate good search queries, as confirmed by the popularity of query suggestion and other assistance techniques. The user study we conducted tested how users would react and benefit from strategic search hints, which propose a way to split the original difficult search task into simpler intermediate goals. When dealing with a complex multi-step question, a conversational system can report a failure to answer and use hints to suggest next steps to resolve the issue to the user. The results of the user study

reveal, that such hints can be helpful if designed carefully. However, this assistance takes away some satisfaction from the search process.

As some of the user questions are ambiguous, a conversational system should be able to ask clarifications. By analyzing the data from StackExchange CQA platform we can conclude that clarifications are a common phenomenon in human information seeking dialogs. Clarification questions vary by type and form, however, there are some frequent patterns, which can be used to automatically generate questions to resolve ambiguities about objects, mentioned in the question. As a proof of concept, we built a model to predict objects of "what kind of ..." clarification questions. The performance of this model proves the feasibility of the template-based clarification question generation.

6.2   Limitations and Future Work

In this Section, I summarize some of the limitations of the described approaches and propose directions for future research.

Community question answering websites became quite popular and accumulate millions of questions people ask along with suggested answers. Therefore, CQA websites are quite valuable as a knowledge source. Relation extraction approach, proposed in this dissertation, was designed to extract subject-predicate-object triples for a schema-based knowledge base, such as Freebase. However, many of the user questions do not align well with existing schema, which restricts the scope of the model. To resolve this issue, the future work can include developing methods for open information extraction from question-answer pairs, and extracting new predicates for schema-based KBs, *e.g.,* using approaches similar to [83].

Text2KB model, proposed in this dissertation to improve KBQA by employing techniques from text-based question answering, demonstrated its ef-

fectiveness on the WebQuestions benchmark. However, for candidate generation, it mainly relies on KB data, which, as we saw, often does not contain predicates or entities, that user is asking about. The EviNets model was designed to resolve this issue by considering both KB and Text data equally, and score answer candidates based on the support extracted from the various pieces of evidence. One of the limitations of EviNets is that sets of candidate answers and pieces of evidence are "pre-computed" and cannot be extended during the model evaluation. A possible future direction is to extend the evidence pool at run-time and turn to the reinforcement learning techniques for training. In this case, the problem could be cast as a graph search problem, where entity nodes are connected with either retrieved textual evidence, or KB predicates, or both. Another limitation of EviNets is the focus on single entities, whereas answers to many real user questions might be lists, and contain attributes, such as dates, numbers, *etc.* To cover these answer types we can look into existing research on answer extraction, such as the research conducted on the SQuAD dataset [154], where the goal is to extract an answer span from a passage of a Wikipedia article. It is possible to adopt these techniques to generate candidates, and aggregate all the evidence for each of them using EviNets.

Most of the existing approaches to non-factoid question answering, including the system presented in this dissertation, rely on ranking existing passages, extracted from somewhere on the web. The limitation of such an approach is that it does not give a big picture of all the relevant information available out there, suggests an answer, which might not be trustworthy, and/or gives a one-sided opinion on the issue. In the user study on conversational search, we found out that users care about the source of the information, and want to get a diverse set of opinions. While some approaches for answer summarization have recently been proposed [139, 179, 192, 208, 77], the problem is still far from being solved.

The crowdsourcing approach for real-time question answering, proposed in this dissertation, was shown to be effective, but is also quite expensive. EmoryCRQA involved crowdsourcing for every question, which requires maintaining a pool of workers constantly ready to assist. While we presented an analysis to reduce the costs by using less human resources, the future research should look into how to optimally plan question answering and optimize the cost/quality trade-off. A question answering system or personal assistant should turn to crowdsourcing selectively when it was not able to come up with a good response itself. On the other hand, some of the user questions might not be urgent, which allows the system to engage slower, but more reliable community of experts on one of the domain-specific CQA websites. I believe, that future personal assistants should be able to plan and route questions more effectively given the type of question, their time and cost requirements.

The research on chatbots and conversational question answering is at the beginning, and the findings of the presented user study shed some light into the areas the research community should focus on, *e.g.,* methods for maintaining context in a conversation, generating clarification questions, taking positive and negative feedback, *etc.* The limitation of our user study is that we did not consider the voice interface, which adds certain specifics to how a system should present its results [194]. Some of the participants of our user study also raised an interesting point, that web search offers a possibility to browse related information and stumble upon something unexpected. For example, when doing a research into the economics behind hydro-power, a typical web page will cover multiple related topics, such as history, efficiency, environmental concerns, *etc.* In a conversation, responses are typically focused around the asked question, which makes it harder to learn about related topics. It requires a user to ask multiple questions, and actually assumes she is aware of all these aspects beforehand. Research into ways to discover and

present such related information in a conversational search setting might change the user experience significantly, and bring serendipity to the chat and voice interfaces.

In this dissertation, I mainly focused on improving the accuracy of question answering systems. However, there are other metrics, which are quite important to users, *e.g.,* trustworthiness of answers, their completeness in terms of the number of covered subtopics, as well as opinions. All these aspects should be taken into account by the developers of intelligent assistants.

6.3   Contributions and Potential Impact

My Ph.D. dissertation covers a spectrum of aspects related to answering user information needs, primarily focusing on those expressed as natural language questions. The common thread throughout the thesis is the goal of combining and aggregating valuable information from various unstructured (text), semi-structured (question-answer pairs) and structured (Knowledge Base) data sources, as well as direct human contributions (through Crowdsourcing and dialog utterances). Combining such diverse sources of information allowed us to exploit advantages of one source to overcome the disadvantages of the other sources, and improve precision and recall of automatic factoid and non-factoid question answering. I believe, that this idea of information aggregation will be the key component of future intelligent systems, which should be able to reason with all the available knowledge, rather than simply returning the best matching item. The models I developed in my dissertation demonstrate the potential of such an approach and provide some insights for future research in this direction. In particular, I investigated both feature-engineered and feature learning-based (deep learning) methods for evidence aggregation, and introduced EviNets, a principled method for combining diverse sources of evidence for answer ranking in question answering.

In addition to improving the core question answering techniques, it is important to improve the ways a system communicates with users. A conversation or dialog is a natural activity that humans perform on daily basis, and proliferation of mobile personal assistants suggests that conversations might be the next step for information seeking scenarios. In my dissertation, I presented results of of an initial user study of how people use conversations for information seeking tasks, what are the expectations from the personal assistants, and what is currently missing from the commercial systems. I believe that the results and insights gained from this user study will help build the next generations of conversational systems, which will receive wider adoption and will be quite helpful in everyday life. I hope this work will be used by the academic and industry communities as a guide for future research in conversational question answering. Additionally, we explored two particular dialog scenarios for complex informational tasks. Search hints and clarification questions are two possible response types a question answering system can return for complex or ambiguous questions, instead of returning empty or irrelevant answers. Both of these types of interactions integrate nicely into a dialog scenario and enrich the arsenal of tools a conversational system possess in order to help a user satisfy her goals.

Overall, this dissertation describes methods and techniques for improving the performance of question answering systems by combining various user-generated content, including those created on the fly with crowdsourcing or in dialogs with users. These techniques can be integrated into a conversational personal assistant, and the dissertation gives some insights on the expectations and desiderata from such a system, obtained via user studies. The contributions of this dissertation should help develop the next generation of information access systems, that will serve some of more than 1.2 trillion information searches per year [1] for more than 3.5 billion Internet users.

---

[1]http://www.internetlivestats.com/google-search-statistics/

Bibliography

[1] Mikhail Ageev, Qi Guo, Dmitry Lagun, and Eugene Agichtein. Find it if you can: A game for modeling different types of web search success using interaction data. In *Proceedings of the 34th International ACM SIGIR Conference*, pages 345–354, New York, NY, USA, 2011. ACM.

[2] Eugene Agichtein, David Carmel, Donna Harman, Dan Pelleg, and Yuval Pinter. Overview of the trec 2015 liveqa track. In *Proceedings of TREC*, 2015.

[3] Eugene Agichtein, David Carmel, Donna Harman, Dan Pelleg, and Yuval Pinter. Overview of the trec 2016 liveqa track. In *Proceedings of TREC*, 2016.

[4] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL, pages 85–94, New York, NY, USA, 2000. ACM.

[5] Eugene Agichtein, Steve Lawrence, and Luis Gravano. Learning search engine specific query transformations for question answering. In *Proceedings of the Tenth International World Wide Web Conference*, pages 169–178, 2001.

[6] Eyal Aharoni and Alan J Fridlund. Social reactions toward people vs. computers: How mere lables shape interactions. *Computers in human behavior*, 23(5):2175–2189, 2007.

[7] DD Ahn, Valentin Jijkoun, GA Mishne, KE Müller, Maarten de Rijke, and KS Schlobach. Using wikipedia at the trec qa track. In *The Thirteenth Text Retrieval Conference*, 2005.

[8] Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. A neural knowledge language model. *arXiv preprint arXiv:1608.00318*, 2016.

[9] Ali Mohamed Nabil Allam and Mohamed Hassan Haggag. The question answering systems: A survey. *International Journal of Research and Reviews in Information Sciences (IJRRIS)*, 2(3), 2012.

[10] James Allan, Bruce Croft, Alistair Moffat, and Mark Sanderson. Frontiers, challenges, and opportunities for information retrieval. *SIGIR Forum*, 46(1):2–32, 2012.

[11] Omar Alonso and Ricardo Baeza-Yates. Design and implementation of relevance assessments using crowdsourcing. In *Advances in information retrieval*, pages 153–164. Springer, 2011.

[12] Omar Alonso, Daniel E. Rose, and Benjamin Stewart. Crowdsourcing for relevance evaluation. *SIGIR Forum*, 42(2):9–15, November 2008.

[13] Andrea Andrenucci and Eriks Sneiders. Automated question answering: Review of the main approaches. In *Information Technology and Applications, 2005. ICITA 2005. Third International Conference on*, volume 1, pages 514–519. IEEE, 2005.

[14] Ion Androutsopoulos, Graeme D Ritchie, and Peter Thanisch. Natural language interfaces to databases–an introduction. *Natural language engineering*, 1(01):29–81, 1995.

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

[16] Anne Aula, Rehan M Khan, and Zhiwei Guan. How does search behavior change as search becomes more difficult? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 35–44. ACM, 2010.

[17] Bahadir Ismail Aydin, Yavuz Selim Yilmaz, Yaliang Li, Qi Li, Jing Gao, and Murat Demirbas. Crowdsourcing for multiple-choice question answering. In *Proceedings of AAAI*, pages 2946–2953, 2014.

[18] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. *Citeseer*, 2013.

[19] Hannah Bast, Björn Buchhold, Elmar Haussmann, et al. Semantic search on text and knowledge bases. *Foundations and Trends® in Information Retrieval*, 10(2-3):119–271, 2016.

[20] Hannah Bast and Elmar Haussmann. More accurate question answering on freebase. In *Proceedings of CIKM*, 2015.

[21] Petr Baudiš. Yodaqa: a modular question answering system pipeline. In *POSTER 2015-19th International Student Conference on Electrical Engineering*, pages 1156–1165, 2015.

[22] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, 2013.

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

[24] Jonathan Berant and Percy Liang. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558, 2015.

[25] Adam Berger, Rich Caruana, David Cohn, Dayne Freitag, and Vibhu Mittal. Bridging the lexical chasm: statistical approaches to answer-finding. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 192–199. ACM, 2000.

[26] Delphine Bernhard and Iryna Gurevych. Combining lexical semantic resources with question & answer archives for translation-based answer finding. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2*, pages 728–736. ACL, 2009.

[27] Michael S Bernstein, Joel Brandt, Robert C Miller, and David R Karger. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 33–42. ACM, 2011.

[28] Michael S Bernstein, Jaime Teevan, Susan Dumais, Daniel Liebling, and Eric Horvitz. Direct answers for search queries in the long tail. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 237–246. ACM, 2012.

[29] Fumihiro Bessho, Tatsuya Harada, and Yasuo Kuniyoshi. Dialog system using real-time crowdsourcing and twitter large-scale corpus. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL, pages 227–231, 2012.

[30] Sumit Bhatia, Debapriyo Majumdar, and Prasenjit Mitra. Query suggestions in the absence of query logs. In *Proceedings of the 34th International ACM SIGIR Conference*, pages 795–804, New York, NY, USA, 2011.

[31] Jeffrey P Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, et al. Vizwiz: nearly real-time answers to visual questions. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 333–342. ACM, 2010.

[32] Dania Bilal and Joe Kirby. Differences and similarities in information seeking: Children and adults as web users. *Inf. Process. Manage.*, 38(5):649–670, September 2002.

[33] Matthew W Bilotti, Paul Ogilvie, Jamie Callan, and Eric Nyberg. Structured retrieval for question answering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 351–358. ACM, 2007.

[34] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for struc-

turing human knowledge. In *Proceedings of the 2008 ACM International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA, 2008.

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

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

[37] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*, number EPFL-CONF-192344, 2011.

[38] Alessandro Bozzon, Marco Brambilla, and Stefano Ceri. Answering search queries with crowdsearcher. In *Proceedings of the 21st International Conference on World Wide Web*, WWW, pages 1009–1018, New York, NY, USA, 2012.

[39] Pavel Braslavski, Denis Savenkov, Eugene Agichtein, and Alina Dubatovka. What do you mean exactly? analyzing clarification questions in cqa. In *Proceedings of CHIIR*, 2017.

[40] E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. Data-intensive question answering. In *Proceedings of TREC 2001*, January 2001.

[41] Eric Brill, Susan Dumais, and Michele Banko. An analysis of the askmsr question-answering system. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 257–264. Association for Computational Linguistics, 2002.

[42] Chris J.C. Burges. From ranknet to lambdarank to lambdamart: An overview. Technical report, Miscrosoft Research, June 2010.

[43] Davide Buscaldi and Paolo Rosso. Mining knowledge from wikipedia for the question answering task. In *Proceedings of the International Conference on Language Resources and Evaluation*, pages 727–730, 2006.

[44] Michael J. Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. Webtables: Exploring the power of tables on the web. *Proc. VLDB Endow.*, 1(1):538–549, August 2008.

[45] Michael J. Cafarella, Jayant Madhavan, and Alon Halevy. Web-scale extraction of structured data. *SIGMOD Rec.*, 37(4):55–61, March 2009.

[46] Qingqing Cai and Alexander Yates. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013*, pages 423–433, 2013.

[47] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 875–883, New York, NY, USA, 2008.

[48] David Carmel, Menachem Shtalhaim, and Aya Soffer. eresponder: Electronic question responder. In *International Conference on Cooperative Information Systems*, pages 150–161. Springer, 2000.

[49] David Carmel and Elad Yom-Tov. Estimating the query difficulty for information retrieval. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 2(1):1–89, 2010.

[50] Ben Carterette, Evangelos Kanoulas, Mark Hall, and Paul Clough. Overview of the trec 2014 session track. Technical report, DTIC Document, 2014.

[51] Angel X Chang, Valentin I Spitkovsky, Eneko Agirre, and Christopher D Manning. Stanford-ubc entity linking at tac-kbp, again. In *Proceedings of Text Analysis Conference*, TAC'11, 2011.

[52] Tongfei Chen and Benjamin Van Durme. Discriminative information retrieval for knowledge discovery. *arXiv preprint arXiv:1610.01901*, 2016.

[53] Charles LA Clarke, Gordon V Cormack, and Thomas R Lynam. Exploiting redundancy in question answering. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 358–365. ACM, 2001.

[54] Maxime Clément and Matthieu J Guitton. Interacting with bots online: Users reactions to actions of automated programs in wikipedia. *Computers in Human Behavior*, 50:66–75, 2015.

[55] Daniel Cohen and W Bruce Croft. End to end long short term memory networks for non-factoid question answering. In *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval*, pages 143–146. ACM, 2016.

[56] Gao Cong, Long Wang, Chin-Yew Lin, Young-In Song, and Yueheng Sun. Finding question-answer pairs from online forums. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 467–474. ACM, 2008.

[57] Marco Cornolti, Paolo Ferragina, Massimiliano Ciaramita, Hinrich Schütze, and Stefan Rüd. The smaph system for query entity recog-

nition and disambiguation. In *Proceedings of the First International Workshop on Entity Recognition and Disambiguation*, ERD '14, pages 25–30, New York, NY, USA, 2014. ACM.

[58] Hoa Trang Dang, Diane Kelly, and Jimmy J Lin. Overview of the trec 2007 question answering track. In *TREC*, volume 7, page 63. Citeseer, 2007.

[59] Marco De Boni and Suresh Manandhar. Implementing clarification dialogues in open domain question answering. *Natural Language Engineering*, 11(04):343–361, 2005.

[60] Shilin Ding, Gao Cong, Chin-Yew Lin, and Xiaoyan Zhu. Using conditional random fields to extract contexts and answers of questions from online forums. In *ACL*, volume 8, pages 710–718. Citeseer, 2008.

[61] Li Dong, Furu Wei, Ming Zhou, and Ke Xu. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 260–269, 2015.

[62] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD, pages 601–610, New York, NY, USA, 2014.

[63] Huizhong Duan, Yunbo Cao, Chin-Yew Lin, and Yong Yu. Searching questions by identifying question topic and question focus. In *Proceedings of ACL*, pages 156–164, 2008.

[64] Chad Edwards, Autumn Edwards, Patric R Spence, and Ashleigh K Shelton. Is that a bot running the social media feed? testing the differences in perceptions of communication quality for a human agent and a bot agent on twitter. *Computers in Human Behavior*, 33:372–376, 2014.

[65] Shady Elbassuoni, Maya Ramanath, Ralf Schenkel, Marcin Sydow, and Gerhard Weikum. Language-model-based ranking for queries on rdf-graphs. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 977–986. ACM, 2009.

[66] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the web. *Commun. ACM*, 51(12):68–74, December 2008.

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

[68] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1608–1618, 2013.

[69] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1156–1165, New York, NY, USA, 2014. ACM.

[70] Paolo Ferragina and Ugo Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628. ACM, 2010.

[71] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. The rise of social bots. *Communications of the ACM*, 59(7):96–104, 2016.

[72] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

[73] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. This is watson. *IBM Journal of Research and Development*, 56, 2012.

[74] Michael J Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. Crowddb: answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 61–72. ACM, 2011.

[75] Daniel Fried, Peter Jansen, Gustave Hahn-Powell, Mihai Surdeanu, and Peter Clark. Higher-order lexical semantic models for non-factoid answer reranking. *Transactions of the Association for Computational Linguistics*, 3:197–210, 2015.

[76] Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

174

[77] Mahak Gambhir and Vishal Gupta. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66, 2017.

[78] Rashmi Gangadharaiah and Balakrishnan Narayanaswamy. Natural language query refinement for problem resolution from crowdsourced semi-structured data. In *Proceedings of IJCNLP*, pages 243–251, 2013.

[79] Matt Gardner and Tom Mitchell. Efficient and expressive knowledge base completion using subgraph feature extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1488–1498, 2015.

[80] Catherine Grady and Matthew Lease. Crowdsourcing document relevance assessment with mechanical turk. In *Proceedings of the NAACL HLT 2010 workshop on creating speech and language data with Amazon's mechanical turk*, pages 172–179. Association for Computational Linguistics, 2010.

[81] Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 219–224. ACM, 1961.

[82] Poonam Gupta and Vishal Gupta. A survey of text question answering techniques. *International Journal of Computer Applications*, 53(4), 2012.

[83] Rahul Gupta, Alon Halevy, Xuezhi Wang, Steven Euijong Whang, and Fei Wu. Biperpedia: An ontology for search applications. *Proc. VLDB Endow.*, 7(7):505–516, March 2014.

[84] F Maxwell Harper, Joseph Weinberg, John Logie, and Joseph A Konstan. Question types in social q&a sites. *First Monday*, 15(7), 2010.

[85] Christopher G Harris and Padmini Srinivasan. Comparing crowd-based, game-based, and machine-based approaches in initial query and query refinement tasks. In *Advances in Information Retrieval*, pages 495–506. Springer, 2013.

[86] Morgan Harvey, Claudia Hauff, and David Elsweiler. Learning by example: Training users with high-quality query suggestions. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 133–142, New York, NY, USA, 2015.

[87] Ben He and Iadh Ounis. Query performance prediction. *Information Systems*, 31(7):585–594, 2006.

[88] Hua He and Jimmy Lin. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of NAACL-HLT*, 2016.

[89] Marti A Hearst. 'natural'search user interfaces. *CACM*, 54(11):60–67, 2011.

[90] Michael Heilman and Noah A Smith. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1011–1019. Association for Computational Linguistics, 2010.

[91] Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*, 2016.

[92] Wesley Hildebrandt, Boris Katz, and Jimmy J Lin. Answering definition questions using multiple knowledge sources. In *Proceedings of HLT-NAACL*, pages 49–56, 2004.

[93] Lynette Hirschman and Robert Gaizauskas. Natural language question answering: the view from here. *natural language engineering*, 7(4):275–300, 2001.

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

[95] Eduard Hovy, Ulf Hermjakob, and Deepak Ravichandran. A question/answer typology with surface text patterns. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 247–251, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[96] Eduard H Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. Question answering in webclopedia. In *TREC*, volume 52, pages 53–56, 2000.

[97] Eduard H Hovy, Ulf Hermjakob, Chin-Yew Lin, et al. The use of external knowledge of factoid qa. In *TREC*, volume 2001, pages 644–52, 2001.

[98] Ting-Hao Kenneth Huang, Walter S Lasecki, Amos Azaria, and Jeffrey P Bigham. is there anything else i can help you with?: Challenges in deploying an on-demand crowd-powered conversational agent. In *Proceedings of HCOMP*, 2016.

[99] Ting-Hao Kenneth Huang, Walter S Lasecki, and Jeffrey P Bigham. Guardian: A crowd-powered spoken dialog system for web apis. In *Third AAAI Conference on Human Computation and Crowdsourcing*, 2015.

[100] Kateryna Ignatova, Cigdem Toprak, Delphine Bernhard, and Iryna Gurevych. Annotating question types in social q&a sites. In *Tagungsband des GSCL Symposiums Sprachtechnologie und eHumanities*, pages 44–49. Citeseer, 2009.

[101] Abraham Ittycheriah, Martin Franz, and Salim Roukos. Ibm's statistical question answering system-trec-10. In *Proceedings of TREC*, 2001.

[102] Mohit Iyyer, Jordan L Boyd-Graber, Leonardo Max Batista Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *EMNLP*, pages 633–644, 2014.

[103] Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. Answering complicated question intents expressed in decomposed question sequences. *arXiv preprint arXiv:1611.01242*, 2016.

[104] Sarthak Jain. Question answering over knowledge base using factual memory networks. In *Proceedings of NAACL-HLT*, pages 109–115, 2016.

[105] Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM, pages 84–90, New York, NY, USA, 2005.

[106] Valentin Jijkoun and Maarten de Rijke. Retrieving answers from frequently asked questions pages on the web. In *Proceedings of the 14th*

ACM International Conference on Information and Knowledge Management, CIKM '05, pages 76–83, New York, NY, USA, 2005. ACM.

[107] Valentin Jijkoun, Maarten De Rijke, and Jori Mur. Information extraction for question answering: Improving recall through syntactic patterns. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1284. Association for Computational Linguistics, 2004.

[108] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating query substitutions. In *Proceedings of the 15th International Conference on World Wide Web*, pages 387–396, New York, NY, USA, 2006.

[109] Makoto P Kato, Ryen W White, Jaime Teevan, and Susan T Dumais. Clarifications and question specificity in synchronous social q&a. In *CHI'2013 Extended Abstracts on Human Factors in Computing Systems*, pages 913–918, 2013.

[110] Mostafa Keikha, Jae Hyun Park, W Bruce Croft, and Mark Sanderson. Retrieving passages and finding answers. In *Proceedings of the 2014 Australasian Document Computing Symposium*, page 81. ACM, 2014.

[111] Diane Kelly, Karl Gyllstrom, and Earl W. Bailey. A comparison of query and term suggestion features for interactive searching. In *Proceedings of the 32Nd International ACM SIGIR Conference*, pages 371–378, New York, NY, USA, 2009.

[112] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[113] Julia Kiseleva, Kyle Williams, Jiepu Jiang, Ahmed Hassan Awadallah, Aidan C Crook, Imed Zitouni, and Tasos Anastasakos. Understanding

user satisfaction with intelligent assistants. In *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*, pages 121–130. ACM, 2016.

[114] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. Boilerplate detection using shallow text features. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM, pages 441–450, New York, NY, USA, 2010.

[115] Oleksandr Kolomiyets and Marie-Francine Moens. A survey on question answering technology from an information retrieval perspective. *Information Sciences*, 181(24):5412–5434, December 2011.

[116] Alexander Kotov and ChengXiang Zhai. Towards natural question guided search. In *WWW'2010*, pages 541–550, 2010.

[117] Sascha Kriewel and Norbert Fuhr. Evaluation of an adaptive search suggestion system. In *Advances in Information Retrieval*, volume 5993 of *Lecture Notes in Computer Science*, pages 544–555. Springer Berlin Heidelberg, 2010.

[118] Cody Kwok, Oren Etzioni, and Daniel S Weld. Scaling question answering to the web. *ACM Transactions on Information Systems (TOIS)*, 19(3):242–262, 2001.

[119] Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W Cohen. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1017–1026. Association for Computational Linguistics, 2012.

[120] Walter S. Lasecki, Rachel Wesley, Jeffrey Nichols, Anand Kulkarni, James F. Allen, and Jeffrey P. Bigham. Chorus: A crowd-powered conversational assistant. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST, pages 151–162, New York, NY, USA, 2013.

[121] Matthew Lease and Emine Yilmaz. Crowdsourcing for information retrieval: introduction to the special issue. *Information retrieval*, 16(2):91–100, 2013.

[122] Baichuan Li, Xiance Si, Michael R Lyu, Irwin King, and Edward Y Chang. Question identification on twitter. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2477–2480. ACM, 2011.

[123] Peng Li, Wei Li, Zhengyan He, Xuguang Wang, Ying Cao, Jie Zhou, and Wei Xu. Dataset and neural recurrent sequence labeling model for open-domain factoid question answering. *arXiv preprint arXiv:1607.06275*, 2016.

[124] Xin Li and Dan Roth. Learning question classifiers. In *19th International Conference on Computational Linguistics, COLING 2002*, 2002.

[125] Xin Li and Dan Roth. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3):229–249, 2006.

[126] Jimmy Lin. An exploration of the principles underlying redundancy-based factoid question answering. *ACM Transactions on Information Systems (TOIS)*, 25(2):6, 2007.

[127] Jimmy Lin and Boris Katz. Question answering from the web using knowledge annotation and knowledge mining techniques. In *Proceed-

ings of the twelfth international conference on Information and knowledge management, pages 116–123. ACM, 2003.

[128] Qiaoling Liu, Eugene Agichtein, Gideon Dror, Yoelle Maarek, and Idan Szpektor. When web search fails, searchers become askers: understanding the transition. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 801–810. ACM, 2012.

[129] Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

[130] Yandong Liu, Jiang Bian, and Eugene Agichtein. Predicting information seeker satisfaction in community question answering. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 483–490, New York, NY, USA, 2008. ACM.

[131] Yuanjie Liu, Shasha Li, Yunbo Cao, Chin-Yew Lin, Dingyi Han, and Yong Yu. Understanding and summarizing answers in community-based question answering services. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING, pages 497–504, Stroudsburg, PA, USA, 2008.

[132] Ewa Luger and Abigail Sellen. Like having a really bad pa: The gulf between user expectation and experience of conversational agents. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 5286–5297. ACM, 2016.

[133] Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *7th Biennial Conference on Innovative Data Systems Research*. CIDR Conference, 2014.

[134] Christopher Malon and Bing Bai. Answer extraction by recursive parse tree descent. *ACL 2013*, page 110, 2013.

[135] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[136] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*, 2016.

[137] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[138] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, 2009.

[139] Bhaskar Mitra, Grady Simon, Jianfeng Gao, Nick Craswell, and Li Deng. A proposal for evaluating answer distillation from web data. In *Proceedings of the WebQA workshop at SIGIR*, 2016.

[140] Neema Moraveji, Daniel Russell, Jacob Bien, and David Mease. Measuring improvement in user search performance resulting from optimal search tips. In *Proceedings of the 34th International ACM SIGIR Conference*, pages 355–364, New York, NY, USA, 2011.

[141] Alessandro Murgia, Daan Janssens, Serge Demeyer, and Bogdan Vasilescu. Among the machines: Human-bot interaction on social q&a

websites. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1272–1279. ACM, 2016.

[142] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

[143] Chorng-Shyong Ong, Min-Yuh Day, and Wen-Lian Hsu. The measurement of user satisfaction with question answering systems. *Information & Management*, 46(7):397–403, 2009.

[144] Bo Pang and Ravi Kumar. Search in the lost sense of query: Question formulation in web search queries and its temporal changes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 135–140. Association for Computational Linguistics, 2011.

[145] Seonyeong Park, Soonchoul Kwon, Byungsoo Kim, and Gary Geunbae Lee. Isoft at qald-5: Hybrid question answering system over linked data and text data. In *Proceedings of CLEF*, 2015.

[146] Marius Pasca and Sanda Harabagiu. The informative role of wordnet in open-domain question answering. In *Proceedings of NAACL-01 Workshop on WordNet and Other Lexical Resources*, pages 138–143, 2001.

[147] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43, 2014.

[148] Jeffrey Pound, Peter Mika, and Hugo Zaragoza. Ad-hoc object retrieval in the web of data. In *Proceedings of the 19th international conference on World wide web*, pages 771–780. ACM, 2010.

[149] John Prager, Jennifer Chu-Carroll, Eric W Brown, and Krzysztof Czuba. Question answering by predictive annotation. In *Advances in Open Domain Question Answering*, pages 307–347. Springer, 2006.

[150] John M Prager. Open-domain question-answering. *Foundations and trends in information retrieval*, 1(2):91–231, 2006.

[151] Vasin Punyakanok, Dan Roth, and Wen-tau Yih. Mapping dependencies trees: An application to question answering. In *Proceedings of AI&Math 2004*, pages 1–10, 2004.

[152] Silvia Quarteroni and Suresh Manandhar. Designing an interactive open-domain question answering system. *Natural Language Engineering*, 15(01):73–95, 2009.

[153] Filip Radlinski and Nick Craswell. A theoretical framework for conversational search. In *Proceedings CHIIR*, 2017.

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

[155] Jinfeng Rao, Hua He, and Jimmy Lin. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016.

[156] Siva Reddy, Mirella Lapata, and Mark Steedman. Large-scale semantic parsing without question-answer pairs. *TACL*, 2:377–392, 2014.

[157] Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140, 2016.

[158] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. *NAACL HLT 2013*, pages 74–84, 2013.

[159] Ian Ruthven and Mounia Lalmas. A survey on the use of relevance feedback for information access systems. *The Knowledge Engineering Review*, 18(02):95–145, 2003.

[160] Hassan Sajjad, Patrick Pantel, and Michael Gamon. Underspecified query refinement via natural language question generation. In *COLING'2012*, pages 2341–2356, 2012.

[161] Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*, 2016.

[162] Denis Savenkov. Ranking answers and web passages for non-factoid question answering: Emory university at trec liveqa. In *Proceedings of TREC*, 2015.

[163] Denis Savenkov and Eugene Agichtein. To hint or not: exploring the effectiveness of search hints for complex informational tasks. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 1115–1118. ACM, 2014.

[164] Denis Savenkov and Eugene Agichtein. Crqa: Crowd-powered real-time automated question answering system. In *will appear in the proceedings of HCOMP*, 2016.

[165] Denis Savenkov and Eugene Agichtein. Emory university at trec liveqa 2016: Combining crowdsourcing and learning-to-rank approaches for real-time complex question answering. In *TREC*, 2016.

[166] Denis Savenkov and Eugene Agichtein. When a knowledge base is not enough: Question answering over knowledge bases with external text data. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 235–244, New York, NY, USA, 2016. ACM.

[167] Denis Savenkov and Eugene Agichtein. Evinets: Neural networks for combining evidence signals for factoid question answering. In *Proceedings of ACL*, 2017.

[168] Denis Savenkov, Wei-Lwun Lu, Jeff Dalton, and Eugene Agichtein. Relation extraction from community generated question-answer pairs. In *Proceedings of the Student Research Workshop at the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 96–102, 2015.

[169] Denis Savenkov, Scott Weitzner, and Eugene Agichtein. Crowdsourcing for (almost) real-time question answering. In *NAACL Workshop on Human-Computer Question Answering*, 2016.

[170] Chirag Shah and Jefferey Pomerantz. Evaluating and predicting answer quality in community qa. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 411–418. ACM, 2010.

[171] Rebecca Sharp, Peter Jansen, Mihai Surdeanu, and Peter Clark. Spinning straw into gold: Using free text to train monolingual alignment

models for non-factoid question answering. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics-Human Language Technologies (NAACL HLT)*, 2015.

[172] Dan Shen, Geert-Jan M Kruijff, and Dietrich Klakow. Exploring syntactic relation patterns for question answering. In *Natural Language Processing–IJCNLP 2005*, pages 507–518. Springer, 2005.

[173] Edward H Shortliffe and Bruce G Buchanan. A model of inexact reasoning in medicine. *Mathematical biosciences*, 23(3):351–379, 1975.

[174] Anna Shtok, Gideon Dror, Yoelle Maarek, and Idan Szpektor. Learning from the past: Answering new questions with past answers. In *Proceedings of the 21st International Conference on World Wide Web*, WWW, pages 759–768, New York, NY, USA, 2012.

[175] R. F. Simmons. Answering english questions by computer: A survey. *Communications of ACM*, 8(1):53–70, January 1965.

[176] Robert F. Simmons. Natural language question-answering systems: 1969. *Commun. ACM*, 13(1):15–30, January 1970.

[177] Rion Snow, Daniel Jurafsky, and Andrew Y Ng. Learning syntactic patterns for automatic hypernym discovery. *Advances in Neural Information Processing Systems 17*, 2004.

[178] Parikshit Sondhi and ChengXiang Zhai. Mining semi-structured online knowledge bases to answer natural language questions on community qa websites. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 341–350. ACM, 2014.

[179] Hongya Song, Zhaochun Ren, Shangsong Liang, Piji Li, Jun Ma, and Maarten de Rijke. Summarizing answers in non-factoid community question-answering. In *Proceedings of WSDM conference*, 2017.

[180] Radu Soricut and Eric Brill. Automatic question answering using the web: Beyond the factoid. *Information Retrieval*, 9(2):191–206, 2006.

[181] Martin M Soubbotin and Sergei M Soubbotin. Patterns of potential answer expressions as clues to the right answers. In *TREC*, 2001.

[182] Valentin I. Spitkovsky and Angel X. Chang. A cross-lingual dictionary for english wikipedia concepts. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012.

[183] Svetlana Stoyanchev, Alex Liu, and Julia Hirschberg. Modelling human clarification strategies. In *Proceedings of SIGDIAL*, pages 137–141, 2013.

[184] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[185] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[186] Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. Open domain question answering via semantic enrichment. In *Proceedings of the 24th International Conference on World Wide Web*, WWW, pages 1045–1055, 2015.

[187] Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics*, 37(2):351–383, 2011.

[188] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL, pages 455–465, Stroudsburg, PA, USA, 2012.

[189] Ming Tan, Bing Xiang, and Bowen Zhou. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*, 2015.

[190] Yang Tang, Fan Bu, Zhicheng Zheng, and Xiaoyan Zhu. Towards interactive qa: suggesting refinement for questions. In *SIGIR'2011 Workshop on "entertain me": Supporting Complex Search Tasks*, pages 13–14, 2011.

[191] 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 Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*, 2015.

[192] Mattia Tomasoni and Minlie Huang. Metadata-aware measures for answer summarization in community question answering. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 760–769, 2010.

[193] Kristina Toutanova, Danqi Chen, Patrick Pantel, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. *ACL Association for Computational Linguistics*, 2015.

[194] Johanne R Trippas, Damiano Spina, Mark Sanderson, and Lawrence Cavedon. Results presentation methods for a spoken conversational search system. In *Proceedings of the First International Workshop on Novel Web Search Interfaces and Systems*, pages 13–15. ACM, 2015.

[195] C Tsai, Wen-tau Yih, and C Burges. Web-based question answering: Revisiting askmsr. Technical report, Technical Report MSR-TR-2015-20, Microsoft Research, 2015.

[196] Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. Learning to rank non-factoid answers: Comment selection in web forums. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2049–2052. ACM, 2016.

[197] Christina Unger, André Freitas, and Philipp Cimiano. An introduction to question answering over linked data. In *Reasoning Web. Reasoning on the Web in the Big Data Era*, pages 100–140. Springer, 2014.

[198] Ricardo Usbeck and Axel-Cyrille Ngonga Ngomo. Hawk@ qald5–trying to answer hybrid questions with various simple ranking techniques. In *Proceedings of CLEF*, 2015.

[199] Pertti Vakkari. Searching as learning: A systematization based on literature. *Journal of Information Science*, 42(1):7–18, 2016.

[200] Patrick Verga and Andrew McCallum. Row-less universal schema. *arXiv preprint arXiv:1604.06361*, 2016.

[201] Ellen M Voorhees. The trec question answering track. *Natural Language Engineering*, 7(04):361–378, 2001.

[202] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[203] Alexandra Vtyurina, Denis Savenkov, Eugene Agichtein, and Charles L.A. Clarke. Exploring conversational search with humans, search engines, and wizards. In *In Proceedings of CHI*, 2017.

[204] Chenguang Wang, Yangqiu Song, Ahmed El-Kishky, Dan Roth, Ming Zhang, and Jiawei Han. Incorporating world knowledge to document clustering via heterogeneous information networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1215–1224. ACM, 2015.

[205] Chenguang Wang, Yangqiu Song, Haoran Li, Ming Zhang, and Jiawei Han. Text classification with heterogeneous information network kernels. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[206] Di Wang and Eric Nyberg. Cmu oaqa at trec 2015 liveqa: Discovering the right answer with clues. Technical report, Carnegie Mellon University Pittsburgh United States, 2015.

[207] Di Wang and Eric Nyberg. A long short-term memory model for answer sentence selection in question answering. 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*, pages 707–712, 2015.

[208] Lu Wang, Hema Raghavan, Claire Cardie, and Vittorio Castelli. Query-focused opinion summarization for user-generated content. *arXiv preprint arXiv:1606.05702*, 2016.

[209] Mengqiu Wang. A survey of answer extraction techniques in factoid question answering. *Computational Linguistics*, 1(1), 2006.

[210] Mengqiu Wang and Christopher D Manning. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1164–1172, 2010.

[211] Mengqiu Wang, Noah A Smith, and Teruko Mitamura. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*, volume 7, pages 22–32, 2007.

[212] Pengwei Wang, Lei Ji, Jun Yan, Lianwen Jin, and Wei-Ying Ma. Learning to extract conditional knowledge for question answering using dialogue. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 277–286. ACM, 2016.

[213] Xun Wang, Katsuhito Sudoh, Masaaki Nagata, Tomohide Shibata, Kawahara Daisuke, and Kurohashi Sadao. Reading comprehension using entity-based memory network. *arXiv preprint arXiv:1612.03551*, 2016.

[214] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *EMNLP*, pages 1591–1601. Citeseer, 2014.

[215] Zhenghao Wang, Shengquan Yan, Huaming Wang, and Xuedong Huang. Large-scale question answering with joint embedding and proof

tree decoding. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1783–1786. ACM, 2015.

[216] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*, 2017.

[217] Zhiguo Wang and Abraham Ittycheriah. Faq-based question answering via word alignment. *arXiv preprint arXiv:1507.02628*, 2015.

[218] Steven Euijong Whang, Peter Lofgren, and Hector Garcia-Molina. Question selection for crowd entity resolution. *Proc. VLDB Endow.*, 6(6):349–360, April 2013.

[219] Ryen W White. *Interactions with search systems*. Cambridge University Press, 2016.

[220] Ryen W White, Matthew Richardson, and Wen-tau Yih. Questions vs. queries in informational search tasks. In *Proceedings of the 24th International Conference on World Wide Web*, pages 135–136. ACM, 2015.

[221] Robert Wilensky, David N Chin, Marc Luria, James Martin, James Mayfield, and Dekai Wu. The berkeley unix consultant project. *Computational Linguistics*, 14(4):35–84, 1988.

[222] Daya C Wimalasuriya and Dejing Dou. Ontology-based information extraction: An introduction and a survey of current approaches. *Journal of Information Science*, 2010.

[223] William A Woods and R Kaplan. Lunar rocks in natural english: Explorations in natural language question answering. *Linguistic structures processing*, 5:521–569, 1977.

[224] GuoShun Wu and Man Lan. Leverage web-based answer retrieval and hierarchical answer selection to improve the performance of live question answering. In *Proceedings of TREC*, 2015.

[225] Iris Xie and Colleen Cool. Understanding help seeking within the context of searching digital libraries. *Journal of the American Society for Information Science and Technology*, 60(3):477–494, 2009.

[226] Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. Question answering on freebase via relation extraction and textual evidence. In *Proceedings of ACL*, 2016.

[227] Kun Xu, Sheng Zhang, Yansong Feng, and Dongyan Zhao. Answering natural language questions via phrasal semantic parsing. In *Natural Language Processing and Chinese Computing*, pages 333–344. Springer, 2014.

[228] Mohamed Yahya, Denilson Barbosa, Klaus Berberich, Qiuyue Wang, and Gerhard Weikum. Relationship queries on extended knowledge graphs. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 605–614. ACM, 2016.

[229] Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. Robust question answering over the web of linked data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1107–1116. ACM, 2013.

[230] Jiang-Ming Yang, Rui Cai, Yida Wang, Jun Zhu, Lei Zhang, and Wei-Ying Ma. Incorporating site-level knowledge to extract structured data from web forums. In *Proceedings of the 18th International Conference*

on World Wide Web, WWW '09, pages 181–190, New York, NY, USA, 2009. ACM.

[231] Liu Yang, Qingyao Ai, Jiafeng Guo, and W Bruce Croft. anmm: Ranking short answer texts with attention-based neural matching model. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016.

[232] Liu Yang Yang, Qingyao Ai, Damiano Spina, Ruey-Cheng Chen, Liang Pang, W. Bruce Croft, Jiafeng Guo, and Falk Scholer. Beyond factoid qa: Effective methods for non-factoid answer sentence retrieval. In *Proceedings of ECIR*, 2016.

[233] Yi Yang, Wen-tau Yih, and Christopher Meek. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018. Citeseer, 2015.

[234] Xuchen Yao. Lean question answering over freebase from scratch. In *Proceedings of NAACL Demo*, 2015.

[235] Xuchen Yao, Jonathan Berant, and Benjamin Van Durme. Freebase qa: Information extraction or semantic parsing? *ACL 2014*, page 82, 2014.

[236] Xuchen Yao and Benjamin Van Durme. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014*, pages 956–966, 2014.

[237] Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. Answer extraction as sequence tagging with tree edit distance. In *HLT-NAACL*, pages 858–867. Citeseer, 2013.

[238] Xuchen Yao, Benjamin Van Durme, and Peter Clark. Automatic coupling of answer extraction and information retrieval. In *ACL (2)*, pages 159–165, 2013.

[239] Scott Wen-tau Yih, Xiaodong He, and Christopher Meek. Semantic parsing for single-relation question answering. In *ACL (2)*, pages 643–648. Citeseer, 2014.

[240] Scott Wen-tau Yih, Matt Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of ACL*, 2016.

[241] Pengcheng Yin, Nan Duan, Ben Kao, Junwei Bao, and Ming Zhou. Answering questions with complex semantic constraints on open knowledge bases. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1301–1310. ACM, 2015.

[242] Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*, 2014.

[243] John M Zelle and Raymond J Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055, 1996.

[244] Guangyou Zhou, Tingting He, Jun Zhao, and Po Hu. Learning continuous word embedding with metadata for question retrieval in community question answering. In *Proceedings of ACL*, pages 250–259, 2015.