Question Answering Using User Generated Content

Doctoral thesis proposal

Denis Savenkov
Dept. of Math & Computer Science
Emory University
denis.savenkov@emory.edu

August, 2016

Abstract

Modern search engines have made a dramatic progress in answering many users questions, especially about facts, such as those that might be retrieved or directly inferred from a knowledge base. However, many other questions that real users ask, *e.g.* more complex factual, opinion or advice questions, are still largely beyond the competence of computer systems. For such information needs users still have to dig deeper into the "10 blue links" and extract relevant pieces of information. As conversational agents become more popular, question answering (QA) systems are increasingly expected to handle such complex questions and provide users with helpful and concise information. Unfortunately, a single method doesn't exist for all of the different QA needs.

The goal of my thesis is to improve the performance of question answering systems for solving users' information needs using various types of user generated content, such as text documents, curated knowledge bases, community question answering archives and direct human contributions. To achieve this goal, I propose first to focus on improving performance of automatic question answering by better utilization of available structured and unstructured data sources. Next, I propose to look into possible direct inputs both from the external crowd of workers and the user who asked the question. More specifically, the first part of my future thesis will study how to combine available unstructured text and structured knowledge base (KB) data for factoid question answering. In the second I will build a non-factoid QA system that improves different stages of a pipeline by utilizing available unstructured and semi-structured (*e.g.* question-answer pairs) data. The next part will discuss some methods for crowdsourcing for near real-time question answering and study how to optimize its cost efficiency. Finally, the last part of my thesis will focus on some aspects of user interactions with the question answering system, and how we can utilize this dialog to improve the success of the information seeking process.

Together, these techniques aim to improve performance of question answering over a variety of different questions a user might have, increasing the power and breadth of QA systems.

Contents

1 Introduction and Motivation **1**

2 Related Work **4**
- 2.1 Factoid question answering . . . 4
  - 2.1.1 Text-based question answering . . . 4
  - 2.1.2 Knowledge base question answering . . . 6
  - 2.1.3 Hybrid question answering . . . 8
- 2.2 Non-factoid question answering . . . 10
- 2.3 Crowdsourcing for Question Answering . . . 11
- 2.4 User Interactions with Question Answering Systems . . . 12

3 Combining Data Sources for Factoid Question Answering **14**
- 3.1 Problem . . . 14
- 3.2 Approaches . . . 15
  - 3.2.1 Relation Extraction from Question-Answer Pairs . . . 15
  - 3.2.2 Text2KB: Knowledge Base Question Answering using External Text Data . 21
- 3.3 Proposed Research: Hybrid Question Answering using Text and Knowledge Base Data . . . 34
  - 3.3.1 Method . . . 34
  - 3.3.2 Experimentation . . . 36
- 3.4 Summary . . . 37

4 Improving Non-factoid Question Answering **38**
- 4.1 Problem . . . 38
- 4.2 Approach . . . 39
  - 4.2.1 System architecture . . . 40
  - 4.2.2 Evaluation . . . 43
  - 4.2.3 Analysis . . . 44
  - 4.2.4 Further improvements for TREC LiveQA 2016 . . . 47
  - 4.2.5 Summary . . . 47
- 4.3 Proposed Research . . . 48
  - 4.3.1 Method . . . 48
  - 4.3.2 Experimentation . . . 48
- 4.4 Summary . . . 50

5 Crowdsourcing for Real-time Question Answering 51
5.1 Problem 51
5.2 Approach 51
5.2.1 Crowdsourcing for time-constraint answer generation and validation 52
5.2.2 CRQA: Crowd-powered Real-time Automated Question Answering System 57
5.2.3 Summary 67
5.3 Proposed Directions for Future Research 68
5.3.1 Method 68
5.3.2 Experimentation 69
5.4 Summary 70

6 User Interactions with Question Answering Systems 71
6.1 Problem 71
6.2 Approach 71
6.2.1 Search Hints for Complex Informational Tasks 71
6.3 Proposed Research 76
6.3.1 Method 77
6.3.2 Experimentation 78
6.4 Summary 79

7 Summary & Research Timeline 80
7.1 Research Objectives 80
7.2 Research Plan 80
7.2.1 Combining KB and Text Data for Factoid Question Answering (Chapter 3) 80
7.2.2 Answer Summarization for Non-factoid Question Answering (Chapter 4) . 81
7.2.3 Crowdsourcing for near Real-time Question Answering (Chapter 5) 82
7.2.4 User Interactions with Question Answering Systems (Chapter 6) 82
7.3 Research Timeline 83
7.4 Summary 84

Bibliography 85

1 Introduction and Motivation

It has long been a dream to communicate with a computer as one might with another human being using natural language speech and text. Nowadays, we are coming 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 Now, Microsoft Cortana, Amazon Alexa, etc., help us with everyday tasks and answer some of our questions. Chat bots are arguably considered "the next big thing", and a number of startups developing this kind of technology has 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 more complex questions users still have to dig into the "10 blue links" 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. Therefore, in my thesis I focus on helping users get answers to their questions by improving question answering methods and the ways a system interact with its users.

User questions vary in many different aspects, each of which has its own set of challenges. It's common to divide questions into *factoid* and *non-factoid*. Factoid questions are inquiring about certain facts and can be answered by a short phrase (or list), *i.e.* entity name, date or number. 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 group, *e.g.* how-to and why questions, recommendation and opinion questions, *etc.* The literature usually refers to these questions as "non-factoid questions". Most of the research in automatic question answering focused on factoid questions [180, 117, 16, 36], and recently more and more works started targeting non-factoid questions category [2, 169, 67, 155]. These types of questions provide quite distinct set of challenges and methods applied to them are often quite different, therefore in my thesis I will first study factoid QA and then propose some ideas to improve non-factoid QA.

Automated question answering systems use various data sources to generate answers to user questions. By their nature, data sources can be classified into *unstructured* (*e.g.* raw natural language text), *semi-structured* (*e.g.* tables) and *structured* (*e.g.* knowledge bases). Each of these types of data has certain advantages and limitations (Table 1.1), which often complement each other. There are a number of methods designed for question answering using text collections, knowledge bases 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. Motivated by this fact, in my thesis I propose to study methods of integrating different data sources for joint question answering.

Two major paradigms for factoid question answering are knowledge base question answering (KBQA) and text-based question answer (TextQA). Information contained in a huge volume of text data on the web 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 encode very limited amount of information about mentioned entities and aggregating it over unstructured data

---

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

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

	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 chunk of user needs - hard to extract semantic meaning of a paragraph to match 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

is quite problematic. On the other hand, modern large scale knowledge bases, such as Freebase [27], dbPedia [12], YAGO [123], WikiData [181], aggregate information about millions of entities into a graph of [subject, predicate, object] RDF triples. The problem with KBs is that they are inherently incomplete and miss a lot of entities, facts and predicates. In addition, triple data representation format complicates retrieval of KB concepts relevant to question phrases. The focus of the proposed research in factoid question answering lies on the idea of combining structured KB and unstructured text data, which can help a QA system to overcome these drawbacks.

One of the main challenges in non-factoid question answering 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 [41, 158]. Unfortunately, it's 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. One of the main challenges of this approach is estimating semantic similarity between the question and an answer candidate [163]. Therefore, one would benefit from knowing what kind of questions could a paragraph of text answer. This information can often be inferred from the structure of a web page, e.g. forums, FAQ pages, or estimated using title, subtitle and other page elements. Therefore, in my thesis I'm planning to build a state-of-the-art system for answering complex informational questions using both QnA archives and passages extracted from regular web pages, and combine them in a single model.

While in some cases a single paragraph of text can perfectly answer the question, there are situations when one would need to combine multiple information pieces together in order to cover all aspects of the question, or provide alternative ideas and evidence. These information nuggets are often scattered across different documents, and need to be combined together to form a single answer, that would satisfy the user. In such cases returning a single passage is suboptimal. Therefore, to overcome this challenge in my thesis I propose to apply extractive and abstractive text summarization methods to the problem of answer summarization.

Unfortunately, no matter how good a QA system is, there likely to be cases, when it's unable to return a satisfactory response to the user question, *e.g.* existing data 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 very 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. In the third part of my thesis I propose to explore the effectiveness of crowdsourcing for question answering, especially in the real-time scenario, when the user is unlikely to wait for the response for more than a couple of minutes.

Finally, in the last part of my thesis I'm planning to look into some ways to enrich the interactions between a QA system and its users with the goal of improving the search success rate. Proliferation of mobile personal assistants and conversational agents provide a very natural dialog-based interface between a human and a computer system. Traditionally research in question answering focused on a one-way conversation, where a user only asks questions and the system responds with the answer. However, having a dialog between an asker and a QA system opens up many opportunities to improve the user satisfaction. I propose to focus on two specific kinds of interactions: strategic search hints, that a system might give the user in case she is not satisfied with the answer, and using user feedback to improve the answer candidate ranking.

In summary, this thesis will address 4 complementary aspects of question answering. Chapter 3 presents some results and proposed research to improve factoid question answering using both structured knowledge bases, semi-structured question-answer pairs and unstructured text data sources. Next, in Chapter 4 I will focus on non-factoid question answering, *i.e.* user information needs, that cannot be answered with an entity, number or date. In this chapter I will describe a question answering system I developed to participate in TREC LiveQA shared task, and propose research for improving its performance by answer summarization. Chapter 5 shows that crowdsourcing can be used in near real-time scenario to improve performance of a QA system, and provides some directions for future work. Finally, in Chapter 6 I touch the topic of user interactions with question answering system. More specifically, this chapter describes my prior results on using strategic hints to improve user search success, and proposes some ideas on incorporating user feedback for answer re-ranking in a dialog QA scenario. The overview of the existing research on the above mentioned topics is given in Chapter 2, and Chapter 7 summarizes the proposed research and gives the details on the research timeline and risks.

Modern personal assistants are still far from being intelligent. The research questions I propose to answer in my PhD thesis focus on improving this situation for question answering use cases. Better utilization of structured and unstructured data sources for factoid and non-factoid question answering will improve the core QA capabilities, and in cases when a system is still unable to produce a reasonable answer, we can use crowdsourcing or learn from the user feedback. The results of my thesis will hopefully be useful for future research in developing better intelligent assistants.

2 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 [159, 160]. In 70s and 80s the development of restricted domain knowledge bases and computational linguistics theories facilitated the development of interactive expert and text comprehension systems [11, 157, 197, 195]. The modern era of question answering research was motivated by a series of Text Retrieval Conference (TREC[1]) question answering shared tasks, which was organized annually since 1999 [180]. A comprehensive survey of the approaches from TREC QA 2007 can be found in [51]. An interested reader can refer to a number of surveys to track the progress made in automatic question answering over the years [82, 10, 187, 104, 141, 7, 74].

The main focus of research in automatic question answering was on factoid questions. However, recently we can observe an increased interest in non-factoid question answering, and as an indicator in 2015 TREC started a LiveQA shared task track[2], in which the participant systems had to answer various questions coming from real users of Yahoo! Answers[3] in real time.

In the rest of the chapter I will describe related work in factoid (Section 2.1) and non-factoid (Section 2.2) question answering with the focus on data sources used. In Section 2.3 I will describe existing research in crowdsourcing for question answering and crowdsourcing for real-time systems. Finally, Section 2.4 explains some of the prior works on user interactions with question answering and search systems and some user assistance techniques.

2.1 Factoid question answering

Since the early days of automatic question answering researches explored different sources of data, which lead to the development of two major approaches to factoid question answering: text-based (TextQA) and knowledge base question answering (KBQA) [159]. We will first describe related work in TextQA (Section 2.1.1), then introduce KBQA (Section 2.1.2) and in Section 2.1.3 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 document collections, popularized by TREC QA task, starts by querying a collection with 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 [98], 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.* the keyword who, what, when, *etc.*), and the answer statement might use language that is different from the question (lexical gap problem). On the other side, there is a certain additional information about expected answer statement, that a

---

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

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

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

QA system might infer from the question (*e.g.* we expect to see in 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 [4, 33]. Raw text data might be extended with certain semantic annotations by applying part of speech tagger, semantic role labeling, named entity recognizer, *etc.* By indexing these annotations a question answering system gets an opportunity to query collection with additional attributes, inferred from the question [26, 213].

The next stage in TextQA is to select sentences, that might contain the answer. One of the mostly used benchmark datasets for the task, proposed in [189], is based on TREC QA questions and sentences retrieved by participating systems[4]. The early approaches for the task used simple keyword match strategies [89, 164]. However, in many cases keywords doesn't capture the similarity in meaning of the sentences very well and researches started looking on syntactic information. Syntactic and dependency tree edit distances and kernels allow to measure the similarity between the structures of the sentences [142, 156, 79, 212, 188]. Recent improvements on the answer sentence selection task come are associated with the deep learning techniques, *e.g.* recursive neural networks using sentence dependency tree [90], convolutional neural networks [217, 151], recurrent neural networks [171, 185]. Another dataset, called WikiQA [208], raises a problem of answer triggering, *i.e.* detecting cases when the retrieved set of sentences don't contain the answer.

To provide a user with the concise answer 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 [212], or as a node labeling problem in an answer sentence parse tree [124].

Unfortunately, passages include very limited amount of information about the candidate answer entities, *i.e.* very often it doesn't 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 will host the 2016 summer olympics?*" we need to know that `Rio de Janeiro` is a city and `Brazil` is the country and the correct answer to the question. Therefore, a lot of effort has been put into developing answer type typologies [85, 84] and predicting and matching expected and candidate answer types from the available data [115, 116, 140]. Many approaches exploited external data for this task, I will describe some of these efforts in Section 2.1.3.

Very large text collections, such as the Web, contain many documents expressing the same information, which makes it possible to use a simpler techniques and rely on redundancy of the information. `AskMSR` QA system was one of the first to exploit this idea, and achieved very impressive results on TREC QA 2001 shared task [32]. 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 [176] introduced 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. Another notable systems, that used the web as the source for question answering are `MULDER`[108], `Aranea` [118], and a detailed analysis of what affects the performance of the redundancy-based question answering systems can be found in [117].

---

[4] 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)

2.1.2 Knowledge base question answering

Earlier in the days knowledge bases were relatively small and contained information specific to a particular domain, *e.g.* baseball [73], lunar geology [197], geography [218]. However, one of the problems of techniques developed in this period is domain adaptation, as it's quite challenging to map from natural language phrases to database concepts in open domain when the search space is quite large. Recent development of large scale knowledge bases (*e.g.* dbPedia [12], Freebase [27], YAGO [167], WikiData[5] shifted the attention towards open domain question answering. Knowledge base question answering approaches can be evaluated on an annual Question Answering over Linked Data (QALD[6]) shared task, and some popular benchmark dataset, such as Free917 [37] and WebQuestions [16]. A survey of some of the proposed approaches can be found in [177].

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.

The problem of lexical gap and lexicon construction for mapping natural language phrases to knowledge base concepts is one of the major difficulties in KBQA. The earlier systems were mainly trained from question annotated with the 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 additional lexicon, built from the Web and other resources, has been proposed by Q. Cai and A. Yates [38]. However, most of the parses of a question produce different results, which means that it is possible to use question-answer pairs directly [16]. PARALEX system of A.Fader et al. [63] constructs a lexicon from a collection of question paraphrases from WikiAnswers[7]. A somewhat backward approach was proposed in ParaSempre model of J.Berant et al. [17], which ranks candidate structured queries by first constructing a canonical utterance for each query and then using a paraphrasing model to score it against the original question. Another approach to learn term-predicate mapping is to use patterns obtained using distant supervision [129] labeling of a large text corpus, such as ClueWeb [210]. Such labeled collections can also be used to train a KBQA system, as demonstrated by S.Reddy et al. [143]. This approach is very attractive as it doesn't require any manual labeling and can be easily transfered 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. Modern knowledge bases also contain certain name or surface forms for their predicates and entities, which makes it possible to convert KB RDF triples into questions and use them for training [28]. 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 use an embedding of text and knowledge base concepts into the same 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 [215]. 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 [28]. Memory Networks, developed by the Facebook AI Lab, can also be used to return triples stored in network memory in a response to the user question [29]. 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 [127], 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. To extend deep learning framework to more complex questions, Li Dong et al. [55] used 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 allows a model to focus on different parts of the question and extend the current set of candidate facts, as shown by S.Jain [91].

---

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

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

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

As for the architecture of KBQA systems, two major approaches have been identified: semantic parsing and information extraction. Semantic parsing starts from question utterances and work to produce the corresponding semantic representation, *e.g.* logical form. The model of J.Berant et al. [16] 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 [18], which maintains current best parses across all levels. An alternative information extraction strategy was proposed by [211], which 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 [210].

Question entity identification and disambiguation is the key component in such systems, they cannot answer the question correctly if the question entity isn't identified. Different systems used NER to tag question entities, which are then linked to a knowledge base using a lexicon of entity names [16, 17, 202]. However, NER can easily miss the right span, which wouldn't allow this question to be answered correctly. Most of the recently developed KBQA systems used a strategy to 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 [209, 14, 173]. Ranking of candidates can be done using a simple linear classification model [209] or a more complex gradient boosted trees ranking model [14, 173].

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 [173, 201]. 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 [216]. Z.Wang at al. [192] 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.

2.1.3 Hybrid question answering

A natural idea of combining available information sources to improve question answering has been explored for a long time. Researchers have used various additional resources, such as Wordnet [128], Wikipedia[8] 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 [86, 137]. Next, Wikipedia, which can be characterized as an unstructured and semi-structured (infoboxes) knowledge base, quickly became a valuable resource for answer extraction and verification [6, 35]. Developers of the Aranea QA [118] 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.

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. One approach to increase the coverage of knowledge bases is to extract information from other resources, such as raw text [129, 94, 75], web tables [36], or infer from existing knowledge [109, 71, 30]. 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 [161], semi-supervised [3] and unsupervised [61] methods have been proposed. These techniques analyze individual sentences and can extract facts stated in them using syntactic patterns, sentence similarity, *etc.* In my thesis I extend existing techniques by adapting them to work on one additional type of text data, *i.e.* QnA pairs.

Another relevant angle of relation extraction research is a joint representation of text and knowledge base data. Introduction of text-based edges, extracted from sentences mentioning a pair of entities, to the Path Ranking Algorithm was demonstrated to be superior to KB data alone for knowledge base construction [109]. Such a graph, consisting of KB entities, predicates and textual data can be viewed as heterogeneous information network, and such representation was effectively used to represent text documents for clustering and classification [183, 182]. The idea of universal schemas for relation extraction is represent KB and natural language predicates with embeddings in low dimensional space. The original work of S.Riedel et al. [145] by 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. This techniques were further improved by learning embeddings of individual entities [179], which allows the model to generalize to unseen entity pairs, and compositionality-aware embeddings of natural language [175] to better capture variability of the language. Wang et al [193] shows how to embed entities and words into the same space by preserving entity relations and word coocurrences in text. These approaches aims at computing a similarity between KB predicates and the ways they are expressed in sentences, and they don't 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 [29, 127]. The work I propose to do in my thesis for factoid question has similar objective and setup as key-value memory networks. However, I'm going to focus on open domain question answering and consider both embedding and IR-based methods to address facts in extended knowledge source.

---

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

However, the larger the knowledge base gets, the more difficult it's to find a mapping from natural language phrases to KB concepts. Alternatively, open information extraction techniques [60] can be used to extract a schema-less knowledge base, which can be very effective for question answering. Open question answering approach of A.Fader et al. [62, 216] combines multiple structured (Freebase) and unstructured (OpenIE) knowledge bases together by converting them to string-based triples. User question can be first paraphrased using paraphrasing model learned from WikiAnswers data, then converted to a KB query and certain query rewrite rules can be applied, and all queries are ranked by a machine learning model.

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 [58, 205] is to extend the RDF triples with keywords, which could be extracted from the context of the triple in text, *e.g.* from relation extraction model. These keywords encode the context of the triple and can be used to match against keywords in the question. To query such knowledge graphs authors proposed an extension of SPARQL language, which allows to specify keywords for some triple patterns. However, such queries now require special answer ranking mechanism, *e.g.* based on a language model idea [58]. When answering natural language questions, its often hard to decide whether to map a phrases to some KB concepts, and which one to use. 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. M.Yahya et al. [205, 204] 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 certain type and them ranks them using all keywords from the question.

However, by applying information extraction to raw text we inevitably lose certain portion of the information due to recall errors, and extracted data is also sometimes erroneous due to precision errors. K. Xu et al. [201] proposed to use textual evidence to do answer filtering in a knowledge base question answering system. On the first stage with produce 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 types hierarchies, which were utilized in QuASE system of [168] 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 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 [136, 178]. In my thesis I'm focusing on more open settings, where the text doesn't have to come from inside the knowledge base. In addition, real user questions tends to be more different than hand-crafted ones, which along with larger datasets allows to use machine learning-based modules for answer ranking and selection.

Another great example of a hybrid question answering system is IBM Watson, which is arguably the most important and well-known QA systems ever developed so far. It was designed to play the Jeopardy TV show[9]. The system combined multiple different approaches, including text-based, relation extraction and knowledge base modules, each of which generated candidate answers, which are then pooled together for ranking and answer selection. The full architecture of the system is well described in [64] or in the full special issue of the IBM Journal of Research and Development [65]. YodaQA [15] is an open source implementation of the ideas behind the IBM Watson system.

2.2 Non-factoid question answering

During earlier days of research non-factoid questions received relatively little attention. 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 system was described by R. Soricut, Radu and E. Brill [163] 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 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 of the community to non-factoid questions. Some questions on CQA websites are repeated very often and answers can easily be reused to answer new questions, Y.Liu et al. [120] 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 [19, 158, 57, 92].

Candidate answer passages ranking problem becomes even more difficult in non-factoid questions answering as systems have to deal with 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 [207]. Experiments conducted in this work demonstrated, that classical retrieval methods doesn't 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 M.Surdeanu et al. [169], who explored information retrieval scores, translation models, tree kernel and other features using tokens and semantic annotations (dependency tree, semantic role labelling, *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 [191]. Alignment and translation models are usually based on term-term similarities, which are often computed from a monolingual alignment corpus. This data can be very sparse, and to overcome this issue [67] proposed higher-order lexical semantic models, which estimates similarity between terms by considering paths of length more than 1 on term-term similarity graph. An alternative strategy to overcome the sparseness of monolingual alignment corpora is to use the discourse relations of sentences in a text to learn term association models [155].

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 [220]. 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 [162]. 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 [47, 93, 206, 54, 114].

---

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

TREC LiveQA shared task organized by Yahoo 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 TREC LiveQA 2015 combined similar question retrieval and web search techniques [198, 152, 184]. Answers to similar questions are very effective for answering new questions [152]. However, we a CQA archive doesn't have any similar questions, we have to fall back to regular web search. The idea behind the winning system of CMU [184] is to represent each answer with a pair of phrases - clue and answer text. Clue is a phrase that should be similar to the given question, and the passage that follows should be the answer to this question.

Typically QA system simply rank passages and return the top scoring one as the answer. However, in many cases such passages might either contain redundant information or no individual passage covers all the aspects of the question. In such cases we can apply answer summarization techniques to build the final response. Previous research focused on summarization of answers provided by the community [121, 174, 135, 43, 144]. Y.Liu et al [121] investigated the idea that different types of questions might require different summarization strategies. Some posts on CQA websites are quite long and actually contain multiple subquestions, by identifying those it's possible to group answers according to which particular subquestion do they answer and use this information for summarization [43, 135]. Additionally, answers in CQA websites have some metadata, including the author of the answer, and this information can be effectively used to improve summarization as shown in [174]. An alternative to summarizing answers is to rank them by acknowledging diversity and novelty of aspects, covered by different answers [133]. The key difference between the existing approaches work I propose to do in my thesis is the source of information to summarize. Since I'm planning to build an answer summarization module for a real QA system, it will have to deal with more diverse set of candidates, many of which will be totally irrelevant to the question, which adds additional challenges. The work I'm proposing to do is in sync with the answer distillation idea, described in the research proposal of [130].

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. [21] explored the use of crowdsourcing for 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 thesis 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 [66] 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 [13] 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 isn't effective, and certain answerer confidence weighting schemas can improve the results. CrowdSearcher platform of [31] proposes to use crowds as a data source in the search process, which connects a searcher with the information available through 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 [194] 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., [9, 8, 72] to name a few. The focus in these works is on document relevance, and the quality of crowdsourced judgments. Whereas in my thesis I'm investigating 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 isn't limited to relevance judgements. The work of [77] explores crowdsourcing for query formulation task, which could also be used inside an IR-based question answering system. [113] provides a good overview of different applications of crowdsourcing in information retrieval.

Crowdsourcing is usually associated with offline data collection, which requires significant amount of time. Its application to (near) real-time scenarios poses certain additional challenges. [20] 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 [24] uses a similar strategy to quickly answer visual questions. This work builds on these ideas and uses the proposed retainer model to integrate a crowd into a real-time question answering system. The work of [110, 87] showed how multiple workers can sit behind a conversational agent named Chorus, where human input is used to propose and vote on responses. The work I propose to do in my thesis uses similar ideas in application to non-factoid question answering, which requires more comprehensive responses from the workers. Another use of a crowd for maintaining a dialog is presented in [22], 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. This idea is similar to the proposed research on selective crowdsourcing for QA, however, the main challenge in my work is to estimate the quality of the automatically generated answer.

2.4 User Interactions with Question Answering Systems

There has been 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 [199] demonstrate that in 59.5% of the cases users need help to refine their searches or to construct search statements. Individual term [148] or query suggestion [23, 39, 95] are among the most popular techniques for helping users to augment their queries. The study in Diane Kelly et al [99] 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 [23].

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 [106] and users reported overall satisfaction with its automatic non-intrusive advices. 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 [131]. The study demonstrated the potential effectiveness and teaching effect of hints. Different from [131] in this thesis work I focus on a different type of hints. 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.

Early QA studies considered users the sole proactive part asking refining questions and clarifying on system's response [53]. QA with a more active system's role was investigated within 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 [52]. 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 [105] introduced a concept of *question-guided search*, which can be seen as a variant of query suggestion scenario: in response to initial query the user is presented with a list of natural language questions that reflect possible aspects of the information need behind the query. Tang et al. [172] proposed a method for refinement questions generation consisting 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. [149] 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 [70] 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 answer candidates' set.

Existing information retrieval tools aren't perfect and in many cases fail to return useful information. User interactions data and implicit feedback can be very effective source of information, and allow a system to refine and come up with a better answer. Relevance feedback for document retrieval has been on a research radar for a long time, since Rocchio [146] developed a method for adjusting the query based on available positive and negative feedback documents. Since then a number of extensions for different retrieval models have been proposed, *e.g.* [150, 111, 122, 80] to name a few. However, relevance feedback for question answering is quite different from ad-hoc retrieval, where instead of a single response the goal is to rank documents according to their relevance. In addition, negative feedback is more common, because if a user is satisfied with an answer, a system doesn't get a chance to use this information, in contrast to document retrieval, where quite often the goal is to find as many relevant documents as possible. Negative relevance feedback has some key differences from the positive feedback, which tells us exactly what kind of information is relevant [190].

Overall, the goal of the proposed thesis directions is to improve user satisfaction using question answering systems and personal assistance. Estimating user satisfaction and studying the corresponding factors is another topic on its own, which I'm leaving aside in my thesis. An interested reader might refer to some of the existing research on the topic, *e.g.* [134, 119, 102].

3 Combining Data Sources for Factoid Question Answering

The majority of user web searches are entity-centric, *i.e.* looking for some information or wanting to transact (*e.g.* buy, download) on entities [139]. 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. This section describes the problem I'm going to focus on in my thesis, some existing results and proposals for future research.

3.1 Problem

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, *etc.* Information stored in these data sources can be very useful to solve diverse user information tasks. Moreover, due to different nature of the data sources, their pros and cons are often complimentary, and therefore their combination would have a synergistic effect. In particular, text document collections [104] and structured knowledge bases [177] are very useful for automatic factoid question answering. For example, 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 [117]. On the contrary, for knowledge bases, translating a question into one of the structured query languages can be very challenging [16]. 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 often far from being complete [56]. According to D.Wimalasuriya and D.Dou [196], about 80% of information contained in business documents is stated in natural language, *i.e.* unstructured format. However, text fragments include very limited amount of information about the mentioned entities, which complicates the reasoning and often require certain prediction models to be built [115]. 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 [58, 63, 64, 168, 15]. Prior approaches to the problem of combining textual and structured knowledge base data either process data sources using separate pipelines and merge the results [64, 15], extract structured knowledge from text [3, 129, 56], convert both data sources into a semi-structured format [62], extend knowledge bases with information extracted from text [58, 203] or enrich text with some knowledge about the mentioned entities [168]. Unfortunately, such approaches usually sacrifices some potentially useful information available in the data sources. For example, relation extraction methods extract only a small fraction of information encoded in text documents [56], and existing models of semantic enrichment of text data only uses a subset of KB [168].

In summary, in my thesis I'm focusing on the problem of combining information available in textual data sources and structured knowledge bases to improve factoid question answering. The focus of the proposed research is on utilizing all the available information of different data sources for joint reasoning.

3.2 Approaches

This section describes my previous research on the problem, *i.e.* relation extraction from question-answer pairs (Section 3.2.1) and using external textual data to improve knowledge base question answering (Section 3.2.2).

3.2.1 Relation Extraction from Question-Answer Pairs

Knowledge Bases were found to be quite effective in answering some of the users factual information needs [177]. 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 [56], 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 [3, 75, 94, 129], tables [36], *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.*, has 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 answers new questions [158]. 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 [153]). 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's the capital city of Bolivia? Sucre is the legal capital, though the government sits in La Paz*").

In my thesis I propose a novel model for relation extraction from CQA data, that uses discourse of a QnA pair to extract facts between entities mentioned in 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.

---

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

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

[3] http://quora.com

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

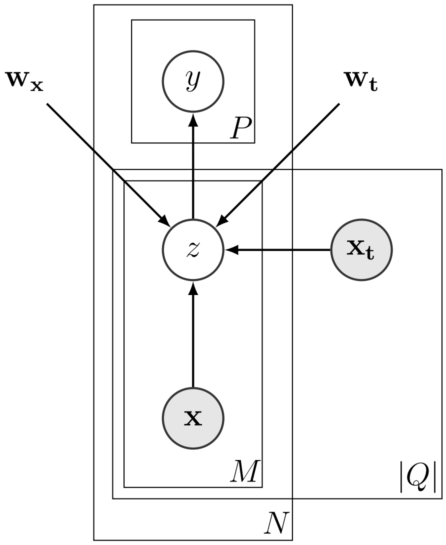


Figure 3.1: QnA-based relation extraction model plate diagram. N - number of different entity pairs, M - number of mentions of an entity pair, $|Q|$ - number of questions where an entity pair is mentioned, $\mathbf{x}$ and $\mathbf{x_t}$ - mention-based and question-based features, $\mathbf{w}$ and $\mathbf{w_t}$ - corresponding feature weights, latent variables z - relation expressed in an entity pair mention, latent variables y - relations between entity pair

The proposed 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 with a set of mention-based features x and question-based features x_t. A multinomial latent variable z represents a relation (or none) expressed in the mention and depends on the features and a set of weights w_x for mention-based and w_t for question-based features:

$$\hat{z} = \underset{z \in P \cup \emptyset}{arg\,max}\ p(z|x, x_t, w_x, w_t)$$

. To estimate this variable we use L2-regularized multinomial logistic regression model, trained using the distant supervision approach for relation extraction [129], 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 $\mathbf{z}$, *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 [170], which was shown to be superior to the original model of [129], is easier to train than some other state of the art distant supervision models and produces comparable results.

Sentence-based model with question features

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 and answer entity to the answer tree root	`Q:(when)→advmod(born)nsubj←[PERSON]` `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*"

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 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 question and answer sentences, designed to capture their topics and relation.

Experiments

For experiments we used 2 publicly available CQA datasets: Yahoo! Answers WebScope dataset[5] and a crawl of WikiAnswers[6] collected by [62]. 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 them have answers. From these clusters we used all possible pairs of questions and corresponding answers (19,629,443 pairs in total).

	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: Yahoo! Answers and WikiAnswers datasets statistics

For each QnA pair we applied tokenization, sentence detection, named entity tagger, parsing and coreference resolution from Stanford CoreNLP [125]. Our cascade entity linking approach is similar to [44] 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 next two stages attempt to match mention text with dictionary of English Wikipedia concepts [165] and its normalized version. Finally for named entity mentions we try spelling correction using Freebase entity names dictionary. We didn't 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 show 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.

---

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

[6] http://wiki.answers.com/

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

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 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 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.

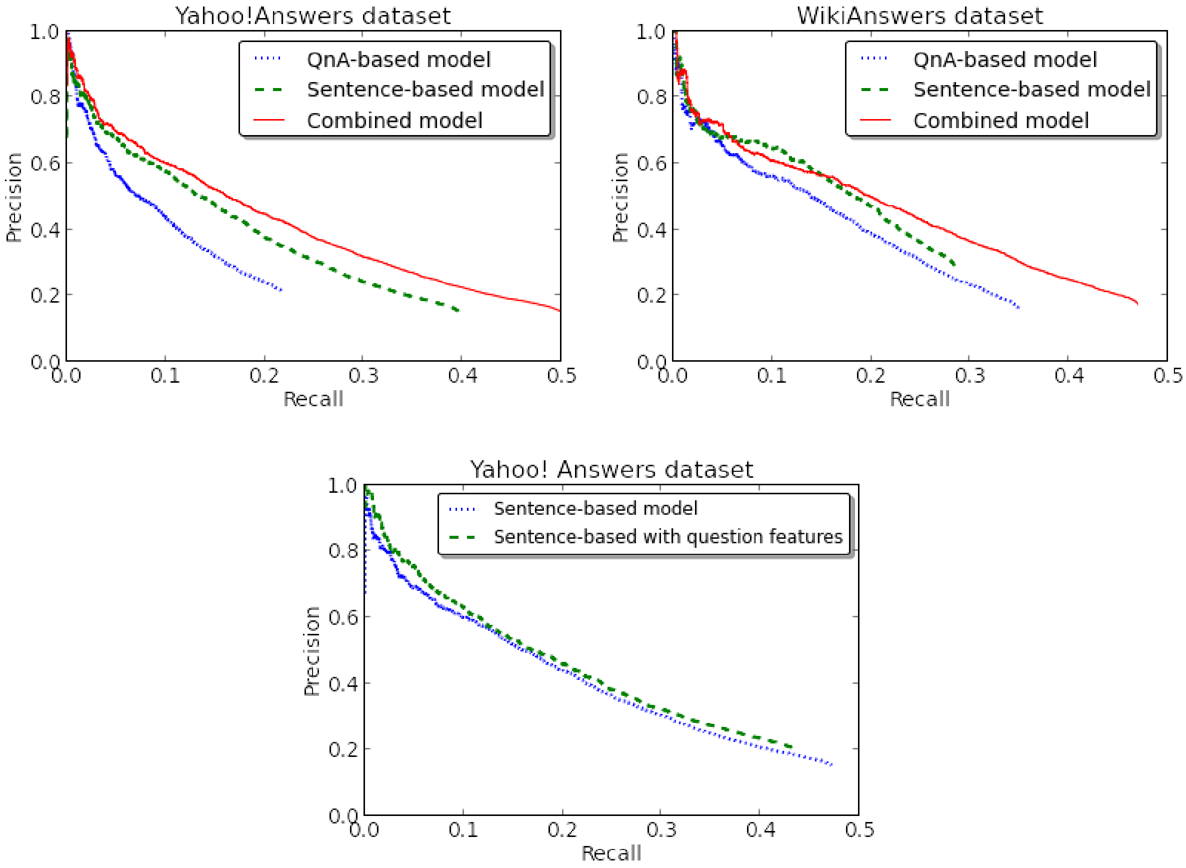


Figure 3.2: Precision-Recall curves for QnA-based vs sentence-based models and sentence-based model with and without question features

	Yahoo! Answers			WikiAnswers		
	QnA	Sentence	Combined	QnA	Sentence	Combined
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: Extraction results for QnA- and sentence-based models on both datasets

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 sentence-based model with and without question-based features (Figure 3.2) didn't show a significant difference.

Analysis

To get an idea of typical problems of 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 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.* [154], 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 advices on different health related topics. How much we can 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 were annotated with existing Freebase relations, which shows a possible limitation of Freebase schema. A promising direction for future work is automatic extraction of new predicates, which users are interested in and which can be useful to answer more future questions.

Summary

In this section we described a model for relation extraction from QnA data, which is capable of predicting relations between entities mentioned in question and answer sentences. We conducted experiments on 2 publicly available CQA datasets and showed that our model can extract triples not available to existing sentence-based techniques and can be effectively combined with them for better coverage of a knowledge base population system.

3.2.2 Text2KB: Knowledge Base Question Answering using External Text Data

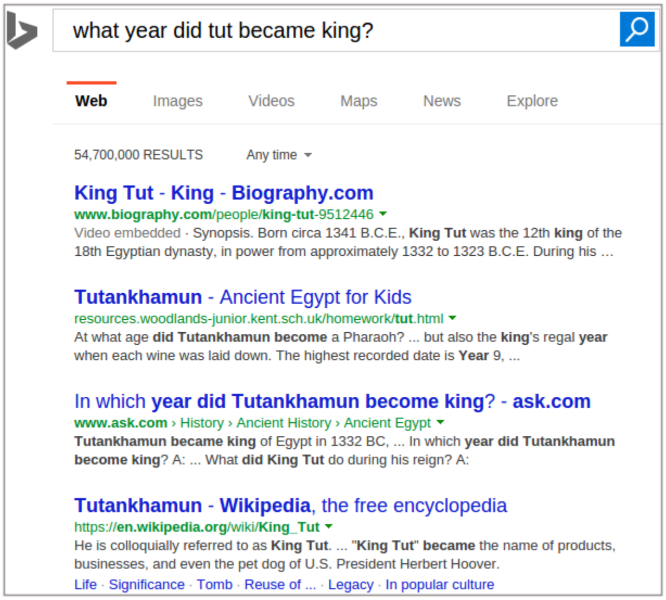
Converting unstructured information into structured form by extracting knowledge from text suffers from certain quality losses. Existing relation extraction tools aren't perfect, in particular due to recall losses a lot of information is left behind. Moreover, extractions contain certain level of incorrect information due to precision losses. These errors cap the upper bound on the question answering system performance. 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).

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 [209]. 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 [166]. 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 doesn't contain a mapping *"tut"* → `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] [48].

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 [14, 16, 17, 18, 28, 210] rely on a lexicon, learned from manually labeled training data, and supported by additional resources, such as question paraphrases [17] and weakly labeled sentences from a large text collection [211]. Such training data tends to be small compared

---

[8] http://web-ngram.research.microsoft.com/ERD2014/



King Tut - King - Biography.com
www.biography.com/people/**king-tut**-9512446
Video embedded · Synopsis. Born circa 1341 B.C.E., **King Tut** was the 12th **king** of the 18th Egyptian dynasty, in power from approximately 1332 to 1323 B.C.E. During his …

Tutankhamun - Ancient Egypt for Kids
resources.woodlands-junior.kent.sch.uk/homework/**tut**.html
At what age **did Tutankhamun become** a Pharaoh? ... but also the **king**'s regal **year** when each wine was laid down. The highest recorded date is **Year** 9, ...

In which **year did Tutankhamun become king**? - **ask.com**
www.ask.com › History › Ancient History › Ancient Egypt
Tutankhamun became king of Egypt in 1332 BC, ... In which **year did Tutankhamun become king**? A: ... What **did King Tut** do during his reign? A:

Tutankhamun - **Wikipedia**, the free encyclopedia
https://**en.wikipedia.org**/wiki/**King_Tut**
He is colloquially referred to as **King Tut**. ... "**King Tut**" **became** the name of products, businesses, and even the pet dog of U.S. President Herbert Hoover.
Life · Significance · Tomb · Reuse of ... · Legacy · In popular culture

Figure 3.3: 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

to the number of different predicates in a KB, and therefore the coverage of these lexicons is limited. By our estimate, in a popular WebQuestions KBQA dataset [16], 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, an 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, 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 [117]. 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 [32]. We propose to adapt 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 [211], in particular it extends the Aqqu system of H.Bast et al. [14], 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, we describe the main stages of the information extraction approach to knowledge base question answering using Aqqu, our baseline system, as an example.

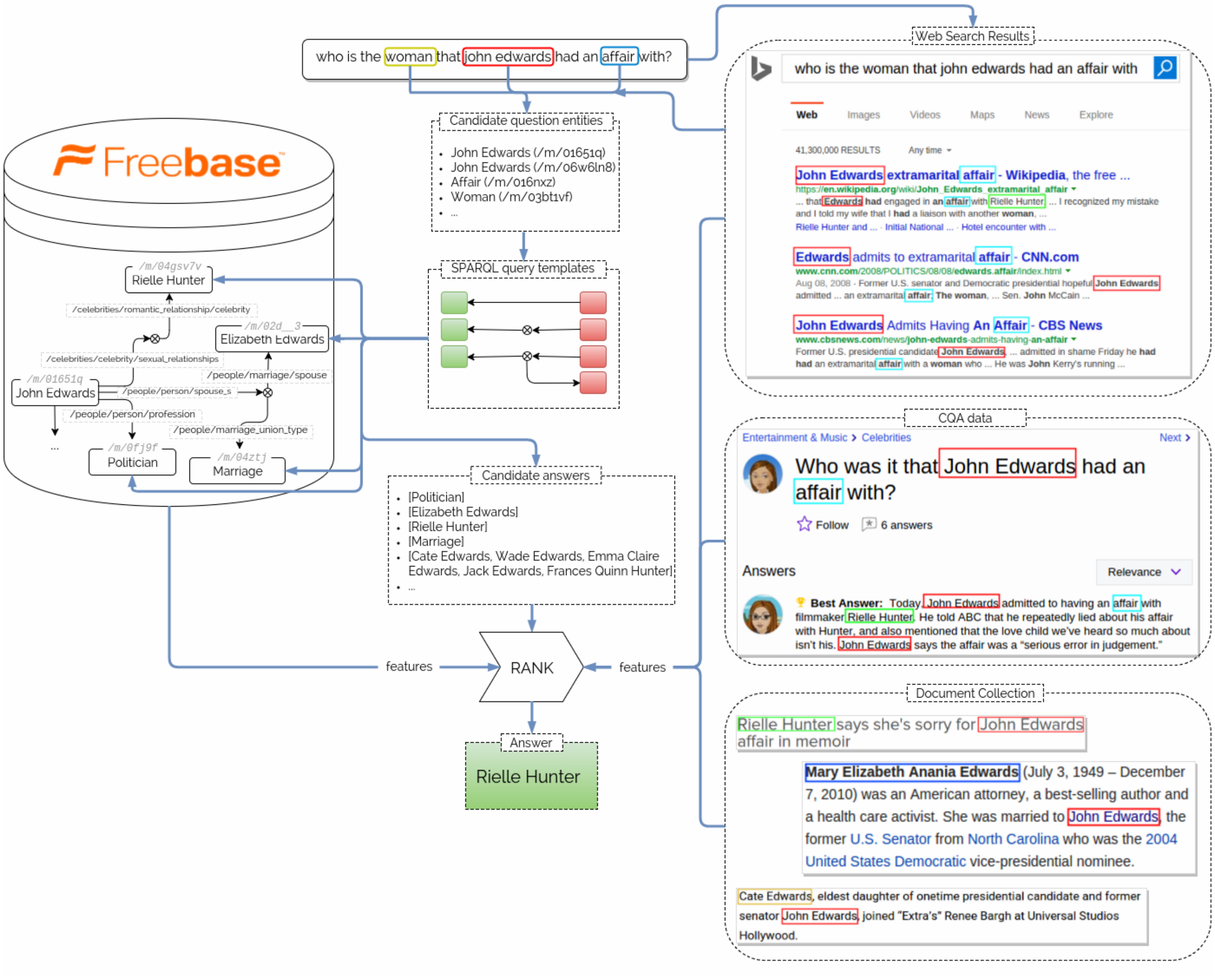

who is the woman that john edwards had an affair with?
Freebase
/m/04gsv7v
Rielle Hunter
/celebrities/romantic_relationship/celebrity
/m/02d__3
Elizabeth Edwards
/celebrities/celebrity/sexual_relationships
/people/marriage/spouse
/m/01651q
John Edwards
/people/person/spouse_s
/people/person/profession
/people/marriage_union_type
/m/0fj9f
Politician
/m/04ztj
Marriage
...
Candidate question entities
John Edwards (/m/01651q)
John Edwards (/m/06w6ln8)
Affair (/m/016nxz)
Woman (/m/03bt1vf)
...
SPARQL query templates
Candidate answers
[Politician]
[Elizabeth Edwards]
[Rielle Hunter]
[Marriage]
[Cate Edwards, Wade Edwards, Emma Claire Edwards, Jack Edwards, Frances Quinn Hunter]
...
features
RANK
features
Answer
Rielle Hunter
Web Search Results
who is the woman that john edwards had an affair with
Web
Images
Videos
Maps
News
Explore
41,300,000 RESULTS
Any time
John Edwards extramarital affair - Wikipedia, the free ...
Edwards admits to extramarital affair - CNN.com
John Edwards Admits Having An Affair - CBS News
CQA data
Entertainment & Music › Celebrities
Next ›
Who was it that John Edwards had an affair with?
Follow
6 answers
Answers
Relevance
Best Answer: Today John Edwards admitted to having an affair with filmmaker Rielle Hunter. He told ABC that he repeatedly lied about his affair with Hunter, and also mentioned that the love child we've heard so much about isn't his. John Edwards says the affair was a "serious error in judgement."
Document Collection
Rielle Hunter says she's sorry for John Edwards affair in memoir
Mary Elizabeth Anania Edwards (July 3, 1949 – December 7, 2010) was an American attorney, a best-selling author and a health care activist. She was married to John Edwards, the former U.S. Senator from North Carolina who was the 2004 United States Democratic vice-presidential nominee.
Cate Edwards, eldest daughter of onetime presidential candidate and former senator John Edwards, joined "Extra's" Renee Bargh at Universal Studios Hollywood.

Figure 3.4: The architecture of our Text2KB Question Answering system

Information Extraction Approach to KBQA

The first stage of the knowledge base question answering process is 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 (`/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 [166] 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 represent 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 doesn't 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 )
}
```

Text2KB model

Now, let's look into the improvements introduced in Text2KB by employing various textual resources.

Web search results for KBQA

Traditional Text-QA 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 [48]. Furthermore, retrieved document set often contains multiple statements of the answer, which can be a strong signal for candidate ranking [117].

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.

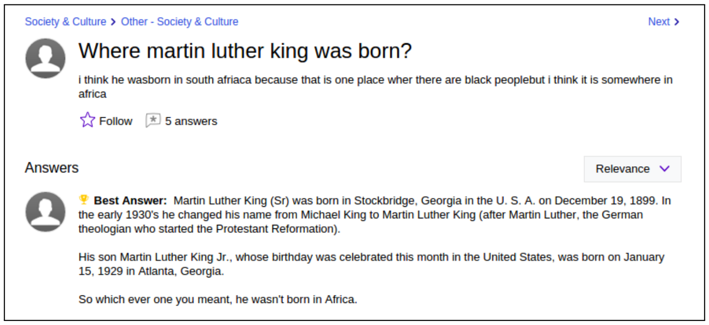
Question text provides only a limited context for entity disambiguation and linking; additionally, the entity name can be misspelled or an uncommon variation used. This complicates the task of entity identification, which is the foundation of KB question answering process. Fortunately, web search results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

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

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. Text-QA 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 [215]. 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:

---

[9] In our experiments we use the Bing Web Search API https://datamarket.azure.com/dataset/bing/search and local Wikipedia search using Lucene



Where martin luther king was born?

i think he wasborn in south afriaca because that is one place wher there are black peoplebut i think it is somewhere in africa

Best Answer: Martin Luther King (Sr) was born in Stockbridge, Georgia in the U. S. A. on December 19, 1899. In the early 1930's he changed his name from Michael King to Martin Luther King (after Martin Luther, the German theologian who started the Protestant Reformation).

His son Martin Luther King Jr., whose birthday was celebrated this month in the United States, was born on January 15, 1929 in Atlanta, Georgia.

So which ever one you meant, he wasn't born in Africa.

Figure 3.5: Example of a question and answer pair from Yahoo! Answers CQA website

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 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 [211]. However, the language used in questions to query about a certain predicate may differ from the language used in statements. In Section 3.2.1 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 learned from single sentence statements, which are already used by our baseline system, but rather introduces the new ones learned from the CQA data.

---

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

Term	Predicate	PMI score
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

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. This 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 (terms with 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.

Estimating Entity Associations

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 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

---

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

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

Entity 1	Entity 2	Term counts
John Edwards	Rielle Hunter	campaign, affair, mistress, child, former ...
John Edwards	Cate Edwards	daughter, former, senator, courthouse, greensboro, eldest ...
John Edwards	Elizabeth Edwards	wife, hunter, campaign, affair, cancer, rielle, husband ...
John Edwards	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

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 text, therefore extending 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.

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 [168]. 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.

Evaluation

This section reports the experimental setup, including the dataset and metrics, as well as the main methods compared for evaluating the performance of our Text2KB system. Additionally, we describe a series of ablation studies to analyze contribution of different system components.

textbfMethods Compared. We compare our system, Text2KB, to state-of-the-art approaches,

System	avg Recall	avg Precision	F1 of avg P and R	avg F1
OpenQA [62]	-	-	-	0.35
YodaQA [15]	-	-	-	0.343
Jacana [211]	0.458	0.517	0.486	0.330
SemPre [16]	0.413	0.480	0.444	0.357
Subgraph Embeddings [28]	-	-	0.432	0.392
ParaSemPre [17]	0.466	0.405	0.433	0.399
Kitt AI [209]	0.545	0.526	0.535	0.443
AgendaIL [18]	0.557	0.505	0.530	0.497
STAGG [215]	0.607	0.528	0.565	0.525
FMN[13] [91]	**0.649**	**0.552**	**0.597**	**0.557**
Aqqu (baseline) [14]	0.604	0.498	0.546	0.494
Text2KB (Wikipedia search)	**0.632*** (+4.6%)	0.498	0.557* (+2.0%)	0.514* (+4.0%)
Text2KB (Web search)	**0.635*** (+5.1%)	0.506* (+1.6%)	**0.563*** (+3.1%)	**0.522*** (+5.7%)

Table 3.6: Performance 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

notably:

- **Aqqu**: a state-of-the-art baseline KBQA system [14], described in Section 3.2.2.
- **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 current KBQA systems [173] 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. We 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, we report average precision and recall, to gain better understanding of the tradeoffs achieved by different methods.

Main Results. The results of existing approaches and our Text2KB system are presented in Table 3.6. 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 F1 score at position 1.

As we can see, Text2KB significantly improves over the baseline system and reaches the current best published result - STAGG [173]. We believe that this system will also benefit from the ideas of our work.

Datasource and Features Contribution. To analyze the contribution of the features and data sources we introduced, we report results from a series of ablation studies. For convenience, we 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 our results table we 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, we analyze the improvements introduced by different components of our system (Table 3.7). 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 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 didn’t 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.

We now anylize the contribution of the different data sources. We will remove a group of web search, CQA or Clueweb-based features and see how the performance of the whole system changes

(Table 3.8). 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	R	P	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 3.7: Average Recall (R), Precision (P), 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.

System	R	P	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 3.8: Average Recall (R), Precision (P), and F1 of Text2KB with and without features based on web search results, CQA data and ClueWeb collection.

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, we analyze the system performance in more detail, and investigate factors for future extension.

Analysis

We now investigate how our system would compare to other systems on the same benchmark; then, we investigate in depth the different error modes, which helps identify the areas of most substantial future improvements.

We took an existing KBQA systems and demonstrated that by combining evidence from knowledge base and external text resources we can boost the performance. A reasonable question is whether the same approach will be helpful to other systems, *e.g.* the currently best system – STAGG [173]. STAGG differs from our baseline system Aqqu in the components: entity linking

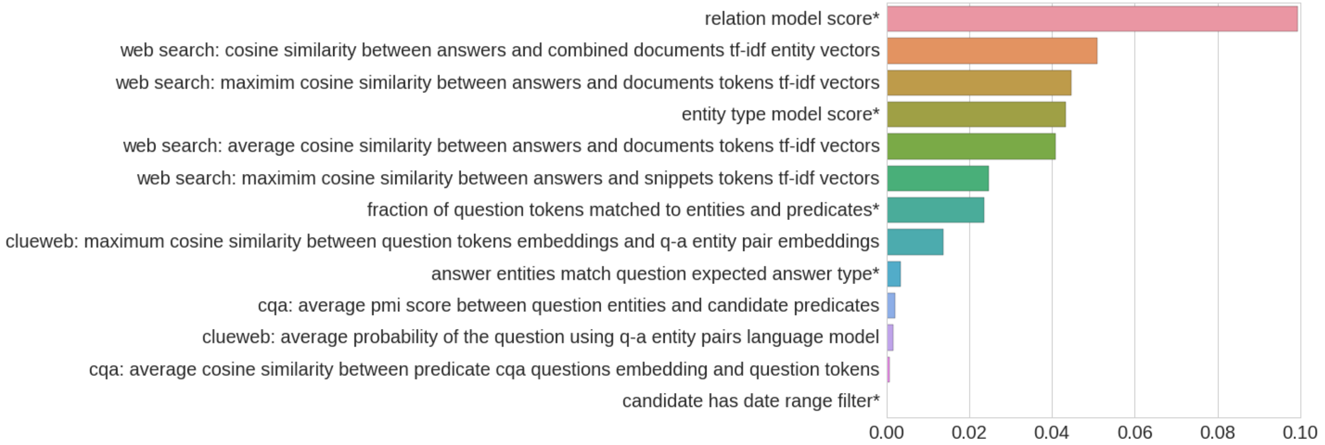


Figure 3.6: A plot of Gini importances of different features of our answer ranking random forest model (features marked * are not text-based and are provided for comparison)

System	avg F1
Text2KB	0.522
STAGG [173]	0.525
Text2KB + STAGG	0.532 (+1.3 %)
Text2KB + STAGG (Oracle)	0.606 (+15.4 %)

Table 3.9: Average F1 for combinations of Text2KB and STAGG using a simple heuristic based on the length of the answer list and Oracle upper bound

algorithm, a set of query templates and ranking methods. Therefore, our approach is complementary and should be helpful for STAGG as well. To support this claim, we 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, we combined answers of STAGG and Text2KB using a simple heuristic: we chose to use the answer returned by STAGG if the number of answer entities is less than in the Text2KB answer, otherwise we use the answer of our approach. Table 3.9 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 we mentioned earlier, answers to 112 of the test questions in the WebQuestions dataset involve predicates that weren't observed in the training set, which may be a problem for approaches that rely on a trained lexicon. We 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 a better insights into the problems that remain, we collected 1219 questions for which Text2KB didn't return completely correct answer, *i.e.* F1 score < 1. We manually looked through a couple of hundreds of these examples 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

---

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

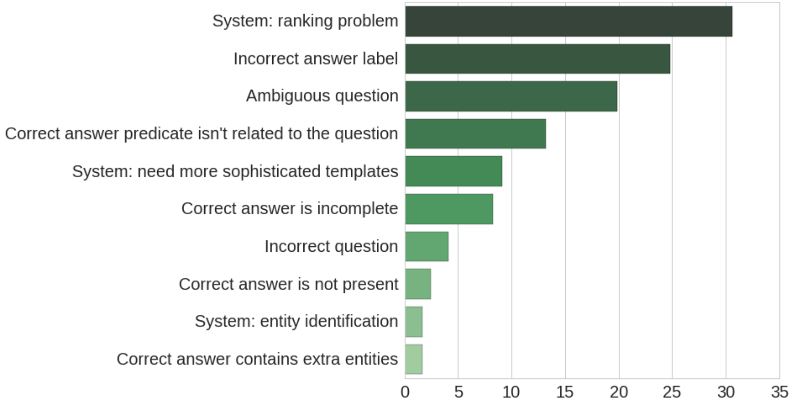


Figure 3.7: Distribution of problems with questions, where Text2KB returns an answer with F1<1

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?"* doesn't 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 are 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 better entity linking strategy, *e.g.* [48]. 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 isn't 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 cooccurrence 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's 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 large number of mentions in the search results.

---

[15] `olympics.olympic_participating_country.athletes`

Summary

In summary, in this section we 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 doesn't help with the problem of knowledge base incompleteness, *i.e.* our system won't 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 research I propose to overcome this problem.

3.3 Proposed Research: Hybrid Question Answering using Text and Knowledge Base Data

Experiments from Section 3.2.2 demonstrated the effectiveness of unstructured data, such as natural language text, to bridge the lexical chasm between KB concepts and natural language questions. However, such an approach only works if concepts and facts referred in the question actually exists in KB. In reality, a big fraction of real user questions cannot be answered using KB data, which is supported by relatively low performance of KBQA systems on general set of factoid questions, such as TREC QA [168].

3.3.1 Method

In this section I propose a novel hybrid QA architecture, that combines unstructured text and structured knowledge base data for joint inference. Figure 3.8 gives a general overview of the proposed system. The idea is to extend the documents representation with annotations of KB entity mentions, which essentially creates additional edges in the knowledge graph. These edges connecting KB entities with text fragments can be traversed by a QA system in both directions in order to get more syntactic (from entity to text) or semantic (from text to entity) information. In addition, text fragments, that mention 2 different entities close to each other can serve as an addition knowledge triple. Unlike information extraction approaches, however, we don't try to extract the predicate and get rid of all the other information stated in a sentence. In the proposed approach we are going to use text similarity metrics to retrieve such triples.

The main stages of the proposed QA pipeline are the following (Figure 3.8):

1. **Question analysis**

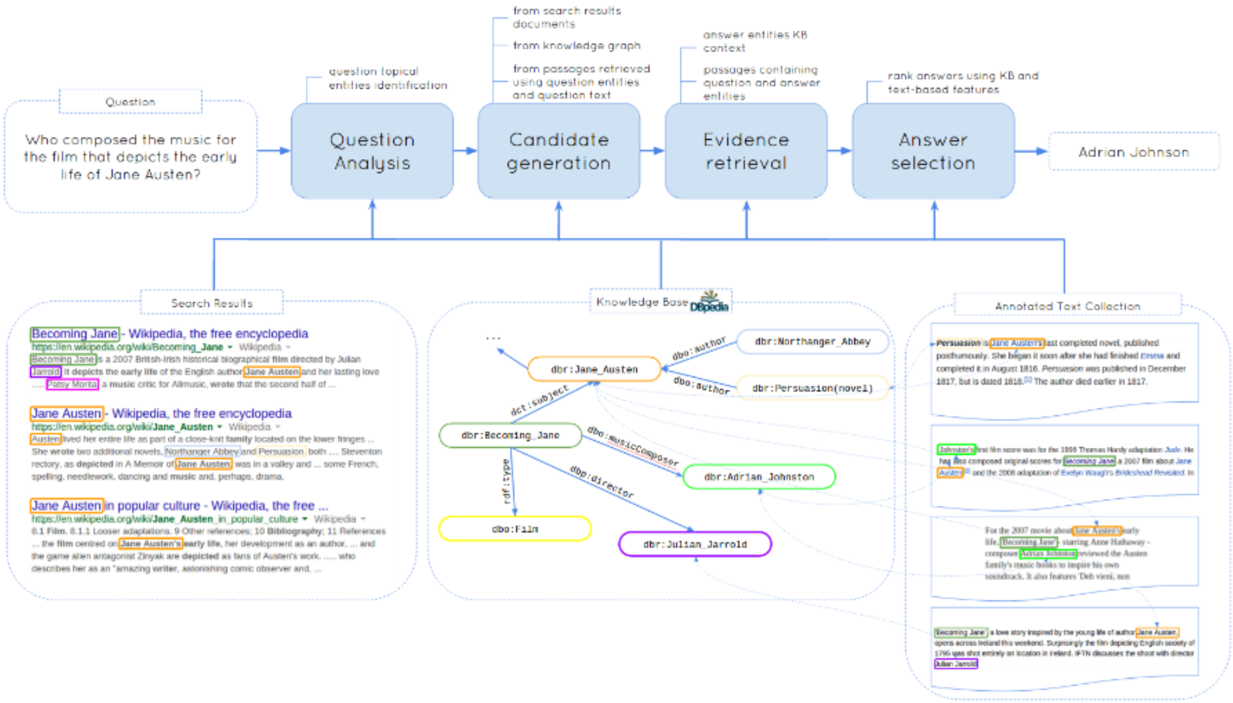

Question
Who composed the music for the film that depicts the early life of Jane Austen?
question topical entities identification
Question Analysis
from search results documents
from knowledge graph
from passages retrieved using question entities and question text
Candidate generation
answer entities KB context
passages containing question and answer entities
Evidence retrieval
rank answers using KB and text-based features
Answer selection
Adrian Johnson
Search Results
Knowledge Base
DBpedia
Annotated Text Collection
dbr:Jane_Austen
dbo:author
dbr:Northanger_Abbey
dbr:Persuasion(novel)
dct:subject
dbr:Becoming_Jane
dbo:musicComposer
dbr:Adrian_Johnston
dbo:director
dbr:Julian_Jarrold
rdf:type
dbo:Film

Figure 3.8: Architecture of a hybrid factoid question answering system, that uses a combination of structured knowledge base and unstructured text data

- **Pre-processing**: identify mentions of KB entities in text document collection and index the documents text and mentions in separate fields
- **Topical entity identification**: search the text collection using question (or reformulated question [4]) as a query and use an approach similar to [49] to detect question topical entities

2. **Candidate generation**
    - **Candidate generation from text**: extract candidate answer (or intermediate answer) entities with evidence from the retrieved text documents using existing techniques, e.g. [176].
    - **Candidate generation from KB**: explore the KB neighborhood of question topical entities and entities extracted from text documents on the previous step
    - **Candidate generation from KB & Text**: use entity and text index to find entities mentioned near question topical entity and question terms in the document collection
3. **Evidence retrieval**
    - **KB evidence extraction**: match neighborhood of answer entities (entity type and other entities) against the question to get additional evidence
    - **Text evidence extraction**: estimate the similarity between the collection text fragments mentioning question and answer entities and the question text
4. **Answer selection**

- **Rank candidate**: rank candidate answers using evidence extracted from the KB as well as from text. One particular appealing idea for a ranking model is to estimate $p(q|G_e)$, where G_e is a knowledge subgraph, associated with the current answer candidate. This probability estimates how likely the question could be generated from the knowledge subgraph language model.

As a motivating example, let's consider the following question from the QALD dataset: "*Who composed the music for the film that depicted the early life of Jane Austen?*". Even though it's quite easy to identify the "`Jane Austen`" entity in the question, the knowledge base (dbPedia in this example) cannot help us to determine which movie is being referred to. However, there are plentiful of documents on the web, that describe the plot of the `Becoming Jane` movie, and a system can use the proximity of terms from the question *"... depicted early life of Jane Austen"* to the movie entity. Unfortunately, extracting the name of the composer from these documents is quite challenging, but this task can be easily accomplished by checking the value of the `musicComposer` property in the knowledge base. At the end, for each candidate answer entity, we have all the KB information and passages that mention this entity as evidence to help with the correct answer selection.

Essentially, the proposed model is a search based approach, similar to existing information extraction methods for knowledge base question answering [211, 14], but over the knowledge graph extended with the links to entity mentions in text collections. In traditional KBQA a special lexicon is used to match terms and phrases in the question to predicates, which essentially represent KB edges. For text edges I'm planning to use the following two approaches: IR-based methods, in particular recent advances in entity search [219, 132], and neural network joint embeddings of predicates and text fragments [28, 127].

3.3.2 Experimentation

To evaluate the performance of the proposed approach I will compare it against several alternatives on multiple datasets. The baselines I will compare against include knowledge base question answering system Aqqu [14], a semantically enriched text-based system QuASE [168], a hybrid YodaQA system, designed to be an open source analogue to IBM Watson [15] and open question answering approach of A.Fader et al [62].

Most of the recent works in knowledge base question answering were evaluated on the WebQuestions dataset [16]. This dataset was recently updated to include ground truth semantic parses and many wrong labels were fixed [214]. I'm planning to evaluate the system I'm going to built on these datasets. Since the answers to these questions can be lists, the official evaluation metric is average F1 score over all questions. However, WebQuestions dataset has its limitations, *e.g.* all the questions are selected so that they could be answered from Freebase. Furthermore, answers to the question were labeled using entities' Freebase profile pages, which only displays relations in the close proximity to the target entity. This makes it possible to exploit a small set of templates to generate candidate answer queries.

Traditionally, most of the works in question answering have been evaluated on the TREC QA datasets. These datasets come with a set of regular expression patterns that can judge an answer as correct or incorrect. Unfortunately, these patterns aren't complete and researchers often end up re-validating the answers of their systems manually [168, 176]. Unlike most of the previous approaches, however, the proposed system aims at retrieving KB entities as answers of the questions. Therefore, I will annotate TREC QA datasets with KB entity identifiers of the correct answers. The baseline systems I mentioned were evaluated of some parts of TREC QA dataset, and I'm going to use reported results instead of reimplementing them. For my system, I will use ClueWeb12 collection, which was annotated with Freebase entity mentions [69]. The text around mentions will be indexed with Lucene[16] open source search engine. The metrics used for evaluation typically include accuracy and mean reciprocal rank (MRR).

However, TREC QA dataset contains only a couple of thousands of examples, which is relatively small. As an alternative, I'm designing a new factoid QA dataset, derived from questions posted to Yahoo! Answers CQA website.

New factoid question answering dataset

Community question answering websites contains hundreds of millions of different questions and corresponding answers, posted by real users. A fraction of these questions represent factoid information needs. I'm going to filter a subset of these questions using a set of heuristics or rules. For example, we can select a subset of QnA pairs with at least one entity in the question and answer, without personal pronouns, words like "*recommend*", "*suggest*", superlative adjectives like "*best*", *etc.* Next this QnA pairs will be further labeled by Mechanical Turk users, who will determine if the questions are indeed factoid and non-subjective and select the actual answer entity from the list of entities, mentioned in the answer text. The preliminary analysis showed, that from 3.8M QnA pairs from Yahoo! Answers WebScope collection, 80K passed the heuristics filters described above and about 30% of them are actually good factoid questions. Examples of questions are: "*What was James Bond's wife's name?*", "*What is the name of the second US astronaut to land on the moon?*", "*When was the first "cartoon human" movie?*", "*What movie did Joe Pesci describe kids as "YUTS"?*". I expect this dataset to contain 10K real user questions annotated with answer entities, which can be used for future research in question answering in both knowledge base and general factoid question answering.

In my thesis this dataset will be used to further compare performances of the proposed approach versus existing KBQA, text and hybrid systems with open code.

3.4 Summary

In this section we considered two different ways of combining unstructured and structured data to improve factoid question answering. Relation extraction from question-answer pairs aims at filling some gaps in KB fact coverage, whereas semantic annotations of text documents provides a way to incorporate information available in unstructured text documents for reasoning along with KB data to improve the performance of factoid question answering. The experiments proposed in this Chapter will help to answer the question on whether combining structured and unstructured data into a single entity graph is beneficial for the question answering performance compared to text-based, KBQA-based or alternative hybrid approaches.

Factoid questions represent just a part of user information needs. Many problems require 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 the Chapter 4.

---

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

4 Improving Non-factoid Question Answering

Even though factoid questions represent a big portion of user information needs, there are many other types of questions, that do not fit into this category, *e.g.* why-questions, how-to questions *etc.* For the majority of such questions modern search engines still provide users with "10 blue links" only. Extracting relevant pieces of information from search results is often quite challenging. With the goal of improving the performance of automatic question answering systems for such generic user information needs in 2015 TREC started a series of LiveQA evaluation campaigns[1]. In TREC LiveQA the task 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 my experience participating in this shared task and research I propose to do to improve non-factoid question answering.

4.1 Problem

User information needs are very diverse, which is reflected in variety of different types of questions, that people post to community question answering websites [76, 88, 121]. Different types of questions require different type of response, which further complicates the problem of automatic question answering. For example, procedural how-to questions are usually answered with a list of instructions, causal why-questions require a passage with certain explanations, whereas some recommendation questions could be answered with an option (*e.g.* hotel name) and possibly some supporting statements.

Previous research on non-factoid question answering either focused on a small subset of questions (*e.g.* definition questions [81]), or considered this as a problem of ranking existing answers in CQA archives, which can be reused to answer new questions [41, 158]. The later strategy of retrieving similar previously posted questions turned out to be quite effective, 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, the system has no other option but to fall back to retrieving potentially relevant passages from regular document collections, such as the web. The system I developed to participate in TREC LiveQA shared task combines these data sources in a single framework, which selects the answer to return from a single pool of candidate answers. The problem of selecting best answer sentence received a lot of attention in recent years[3], and some of this results generalize well to selecting whole passages [186]. The candidate ranking module of my system builds on some of these results. However, individual passages, and even answers to similar questions, often provide just a portion of relevant information a user might want to get. Therefore, a problem of answer summarization becomes quite important [121, 135, 174]. In my thesis I'm planning to work on this problem and the proposed research is described later in this chapter.

In section 4.2 I will describe a system I developed to participate in TREC LiveQA shared task, which establishes a baseline for future experiments. Section 4.3 describes the proposed research

---

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

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

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

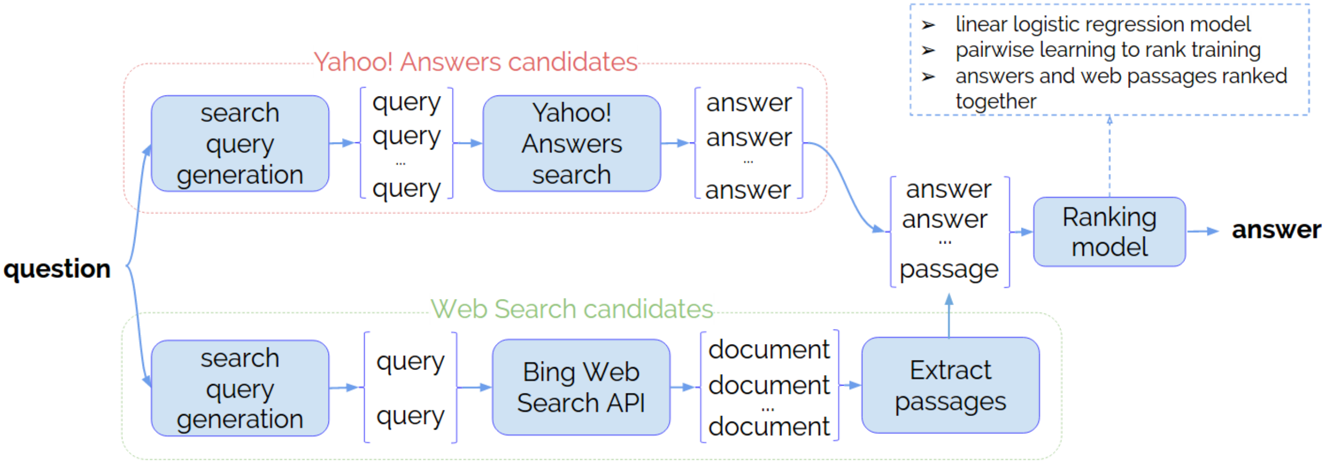

linear logistic regression model
pairwise learning to rank training
answers and web passages ranked together
Yahoo! Answers candidates
search query generation
query
query
...
query
Yahoo! Answers search
answer
answer
...
answer
question
answer
answer
...
passage
Ranking model
answer
Web Search candidates
search query generation
query
query
Bing Web Search API
document
document
...
document
Extract passages

Figure 4.1: Architecture of the question answering system I developed to participate in TREC LiveQA shared task

to improve the performance of non-factoid question answering.

4.2 Approach

In this section, I describe the architecture of the automatic question answering system, which I developed to participate in TREC LiveQA shared task. The system builds on some existing research on question answering and is based on a combination of CQA archive and web search based approaches. There are a lot of different types of questions that users post to CQA websites and it is probably beneficial to study them separately. However, for simplicity the model I built treats all questions in the same way. My system is based on a single trained model, that ranks a set of extracted answer candidates and returns the top one as the response. Preliminary analysis of questions and potential answer sources gave an insight that the best data source is answers to similar questions in case they exist and we can find them. People often have similar tasks and situations which pose same questions. Therefore, it's frequently the case that a similar question was already asked by someone and potentially even received a good reply and can be reused to answer new questions [158]. Of course, many questions or their details are unique, which makes it impossible to find a good match from the existing answers. Therefore I also use web search to generate additional answer candidates. For non-factoid questions it's harder to use the redundancy of the information on the web, which is exploited very effectively in factoid QA [117]. The system I developed extracts passages containing question terms from all retrieved web documents independently. For training I used the publicly available collection of QnA pairs from Yahoo! Answers. The assumption made was that for each question the answer selected as the "best answer" on Yahoo! Answers is indeed the best and should be ranked higher than answers to other questions. However, taking all other answers is intractable and probably detrimental as almost all of them would be totally unrelated to the subject of the given question. Therefore, I used search to retrieve a set of similar questions and took their answers as negative examples. The following chapters describe the QA system in more detail.

4.2.1 System architecture

The general architecture of our question answering system is presented on Figure 4.1. It uses two primary data sources for generating answer candidates: answers to similar questions from Yahoo! Answers website and documents retrieved using web search. All candidates are mixed together, ranked and the top answer is returned.

Candidate generation

Each question issued to a QA system in TREC LiveQA consists of 3 main parts: title, body and category. For example:

> **Question category**: Astronomy & Space
> **Question title**: Why do people claim the Earth isn't the center of the universe?
> **Question body**: Clearly the sun and moon are moving around the Earth otherwise we wouldn't have night and day.

When the QA system receives a question it first generates a set of candidate answers from Yahoo! Answers and regular web search. To generate a set of candidates the system produces several search queries and issues them to both resources.

To find similar questions and extract the corresponding answers from Yahoo! Answers we use the search functionality already available on the website. Some questions are very concise while other provide many useful as well as redundant details. Ideally we want to match as many of them as possible, however, there is a chance that search won't return any results if there are no good matches. Therefore the system generates a set of search queries of different granularity, issues them all to Yahoo! Answers search and collects top 10 responses from all of them. Here is the list of queries that our system generates:

- Concatenation of question title and question body (with and without stopwords)
- Question title only (with and without stopwords)
- Question title concatenated with question body and question category
- Question title concatenated with the name of the question category
- Top 5 terms from question title scored by tf-idf[4]
- Top 5 terms from question title and body scored by tf-idf

For each query and top-10 retrieved questions the system extracts its top answer if provided and puts it into the candidate pool along with some information about the corresponding question and its category.

To extract candidate passages from relevant web documents previous research in factoid question answering have tried query reformulations [5] to better match the potential answer text. However recently [176] demonstrated that such reformulations are no longer necessary as search engines have improved the query processing techniques. Inspired by this observation and considering that retrieving web documents and extracting passages from them is more time consuming, the system issues only 2 web search queries: question title and title concatenated with body. I used Bing Web Search API[5] and the system downloads top-10 retrieved documents, parses HTML code and extracts the main content text [103]. Document content is further split into sentences [125] and candidates are built by taking contiguous sequences of sentences no longer than the answer character limit[6]. The model only keeps passages that contain at least one non-stopword from the question. Web search snippets are also included as candidates.

---

[4] Document frequency is computed on WebScope collection of QnA pairs from Yahoo! Answers

Candidate ranking

A trained linear logistic regression model is used to rank candidate answers, represented with a set of features:

- answer text statistics: length in character, tokens and sentences, average number of tokens per sentence.
- Okapi BM25 scores, which consider question title and concatenation of title and body as queries. Term statistics were computed on Yahoo! Answers WebScope dataset. The score is calculated as follows:

  $$\text{score}(A, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, A) \cdot (k_1 + 1)}{f(q_i, A) + k_1 \cdot (1 - b + b \cdot \frac{|A|}{\text{avg\_al}})}$$

  where $f(q_i, A)$ is frequency of term q_i in the answer text, $k_1 = 1.2$, $B = 0.75$ and $avg_a l = 50$ (average answer length).
- term matches features: lemmas, part of speech tags of matched terms between the question and answer texts, the fraction of unique question terms matched in the answer, length of the maximum span of matched terms in the answer.
- number of matched terms between the question title, body and the title of the page from which the candidate answer is retrieved. For Yahoo! Answers the text of the retrieved question is used as title.
- category match feature for Yahoo! Answers candidates.
- pairs of lemmas from question and answer texts, that are supposed to bridge the lexical gap between question and answer language.
- average, minimum and maximum normalized pointwise mutual information (NPMI) scores between pairs of terms from the question and answer texts. The scores are estimated from QnA pairs from Yahoo! Answers WebScope dataset using the following formula:

  $$\text{npmi}(q_i; a_j) = \frac{pmi(q_i; a_j)}{-\log p(q_i, a_j)}$$

  $$\text{pmi}(q_i; a_j) = \log \frac{p(q_i, a_j)}{p(q_i)p(a_j)} = \log \frac{p(a_j|q_i)}{p(a_j)}$$
- QnA pair score from a Long Short Term Memory (LSTM) neural network model, described in Section 4.2.1.

---

[5] http://datamarket.azure.com/dataset/bing/searchweb

[6] In the final run the limit was 1000 characters

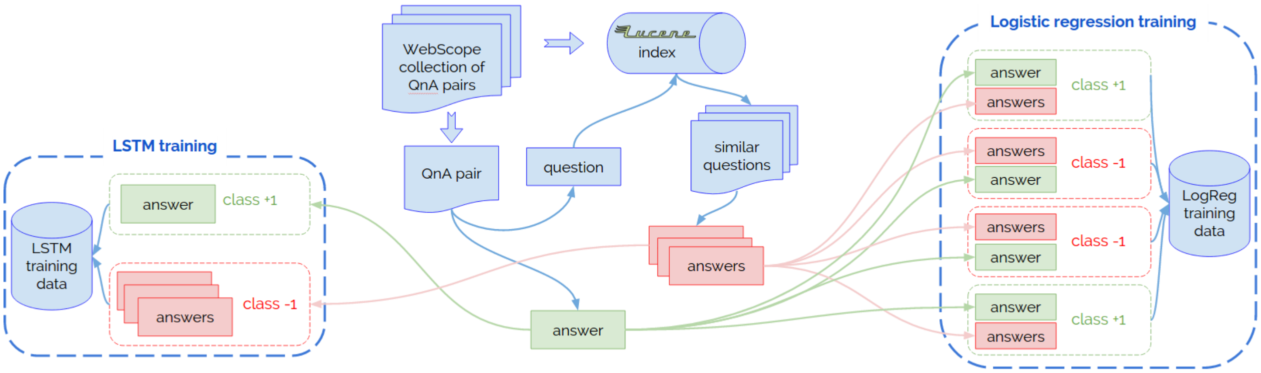

LSTM training
answer
class +1
LSTM training data
answers
class -1
WebScope collection of QnA pairs
Lucene
index
QnA pair
question
similar questions
answers
answer
Logistic regression training
answer
answers
class +1
answers
answer
class -1
answers
answer
class -1
answer
answers
class +1
LogReg training data

Figure 4.2: Workflow for generating training datasets for LSTM and answer ranking logistic regression model from the Yahoo! Answers QnA pairs

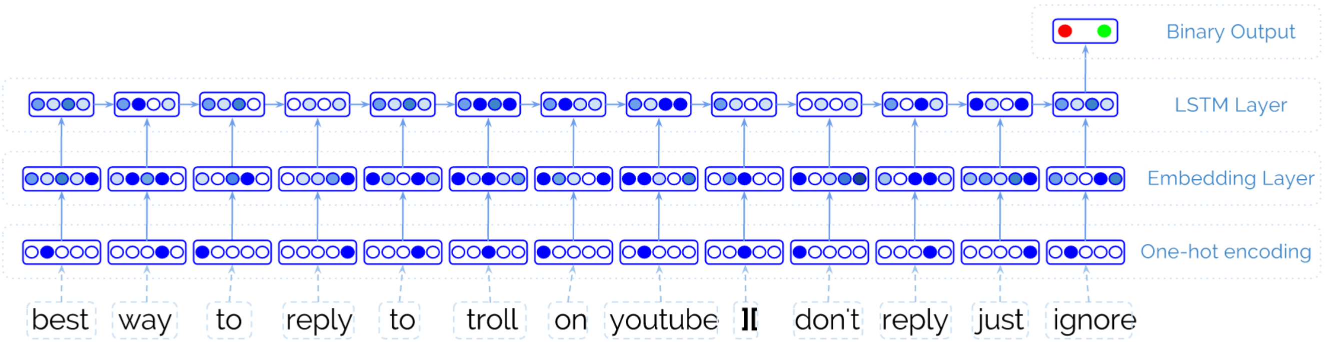

Binary Output
LSTM Layer
Embedding Layer
One-hot encoding
best
way
to
reply
to
troll
on
youtube
][
don't
reply
just
ignore

Figure 4.3: LSTM model for answer scoring. The example shows a QnA pair where the question is “Best way to reply to trolls on youtube?” and the answer is “Don’t reply, just ignore”.

The candidate with the highest score is returned as the answer to the question. If something goes wrong and no candidates were generated or some problem occurred the system returns “I don’t know” as the default answer.

Model Training

There are two trained models used in the system: LSTM recurrent neural network based model, which is used as one of the features for the final logistic regression model that scores all candidates and selects the best one as the answer. I use WebScope Yahoo! Answers dataset[7] (different splits are used) to generate training data for both LSTM and ranking model, Figure 4.2 describes the steps I took to build training datasets.

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. [217, 186] to name a few. I decided to explore this direction and built a recurrent neural network model to score how well a candidate answers a question. Long Short-Term Memory (LSTM) [83] is a particular architecture of recurrent neural networks that helps with the exploding and vanishing gradients problems. The model I developed reads 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.

---

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

Question (title with body) and answer texts are tokenized, punctuation characters are removed and for each token lowercase 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.

To train the model QnA pairs from Yahoo! Answers WebScope dataset were used (we selected a subset of questions from the categories chosen for TREC LiveQA). Each question and the corresponding best answer was used as a positive training example. 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[8] 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[9].

The model was implemented using Keras[10] library. I used an embedding and hidden layers of dimension 128 and the vocabulary size of 1M words. The model was trained using Adam optimization technique [100] 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 we used a different 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. $wf(Q, A_i)$.

4.2.2 Evaluation

From the final run of the system, 1087 questions were judged by the organizers on a scale from 1 to 4:
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 following performance metrics were reported:

- **avg-score(0-3)**: average score over all questions, where scores are translated to 0-3 range. This metric considers "Bad", unreadable answers and unanswered questions, as scored 0.

---

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

[9] It's true, that some of them can indeed be relevant to the original question

[10] http://keras.io

	# answers	avg score (0-3)	s@2+	s@3+	s@4+	p@2+	p@3+	p@4+
1. CMUOAQA	1064	1.081	0.532	0.359	0.190	0.543	0.367	0.179
2. ecnucs	994	0.677	0.367	0.224	0.086	0.401	0.245	0.094
3. NUDTMDP1	1041	0.670	0.353	0.210	0.107	0.369	0.219	0.111
4. RMIT0	1074	0.666	0.364	0.220	0.082	0.369	0.223	0.083
5. Yahoo-Exp1	647	0.626	0.320	0.211	0.095	0.538	0.354	0.159
7. Emory	884↓	0.608↑	0.332↑	0.190↑	0.086↑	0.408↑	0.233↑	0.106↑
Average results	1007	0.467	0.262	0.146	0.060	0.284	0.159	0.065

Table 4.1: Results of the TREC LiveQA evaluation of Emory University QA system and average results of all systems. ↑ means that results of Emory system in this metric are above average, and ↓ means that results are below average

- **s@i+**: the fraction of answers with score i or greater (i=1..4)
- **p@i+**: the number of questions with score i or greater (i=2..4) divided by the number of answered questions

Table 4.1 provides the results of top 5 teams by average answer score, results for our system and average scores. Please refer to the [2] for more details and results of all systems.

The metric succ@1+ shows the fraction of questions for which a readable answer was returned by the system. A lower score in this metic means that my system didn't return an answer in time in almost 20 % of the cases. A part of the problem is caused by several technical issues, that appeared on the day of the evaluation run. Due to one of them LSTM model didn't provide any score for a significant fraction of submitted questions, and the other problem made the whole system unresponsive for a couple of hours. Nevertheless, the overall results look promising and I will make a more thorough analysis of the system performance for the final version of the paper.

The absolute values of the performance metrics demonstrate a great room for improvement as our system was able to return partial or good answer only in 23% of the cases (prec@3+) when the answer was returned. And for 60% of the questions the answer doesn't contain any useful information.

4.2.3 Analysis

In this section we will answer some of the questions about the performance of different system components and their relative importance.

The first question, that we are going to study is the relative effectiveness of web passages and answers to previously posted questions for answering new questions. As Figure 4.4 shows, almost half of all answers returned by my system were generated from Yahoo! Answers. In ∼21% of the cases our system didn't return any results[11], and in the rest ∼31% of the cases a passage from a web page was returned as the answer. I further looked into the domains of the web pages used to generate the answer and noticed, that many more were extracted from other community question answering websites and forums.

---

[11] This happened mainly due to a couple of technical issues that made our system unresponsive for quite some time

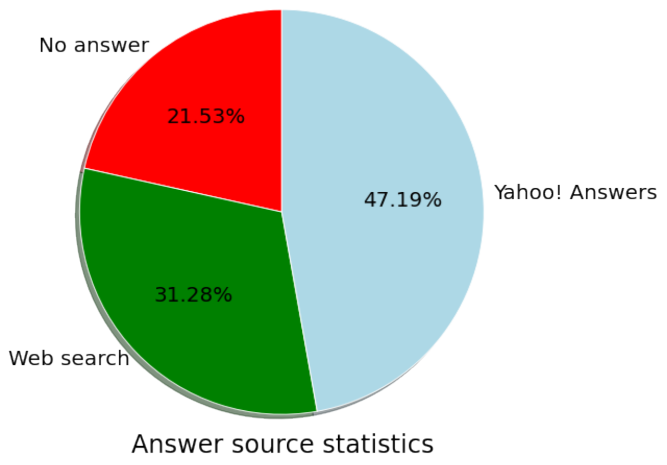
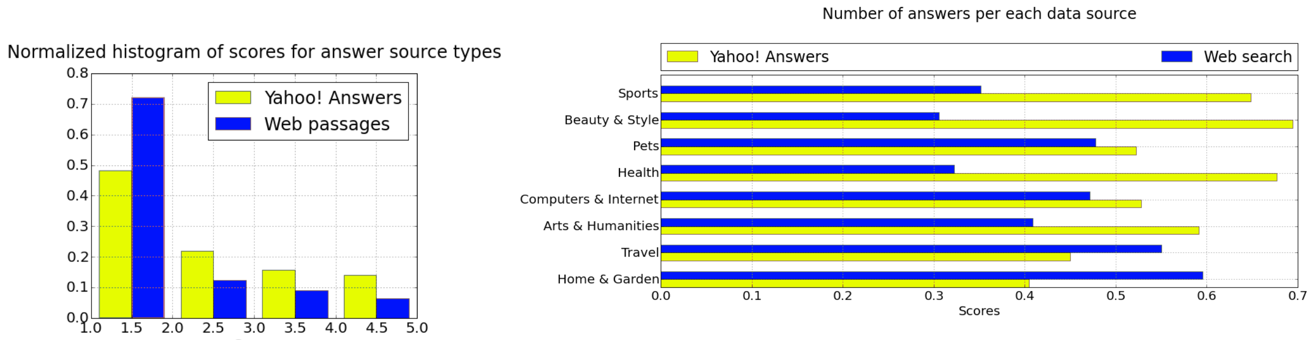


Figure 4.4: Distribution of sources for answers returned by our system



(a) Histogram of qrel scores

(b) Average scores of answers from Yahoo! Answers and web passages for different categories

Figure 4.5: Comparison of web passages and Yahoo! Answers as candidate sources

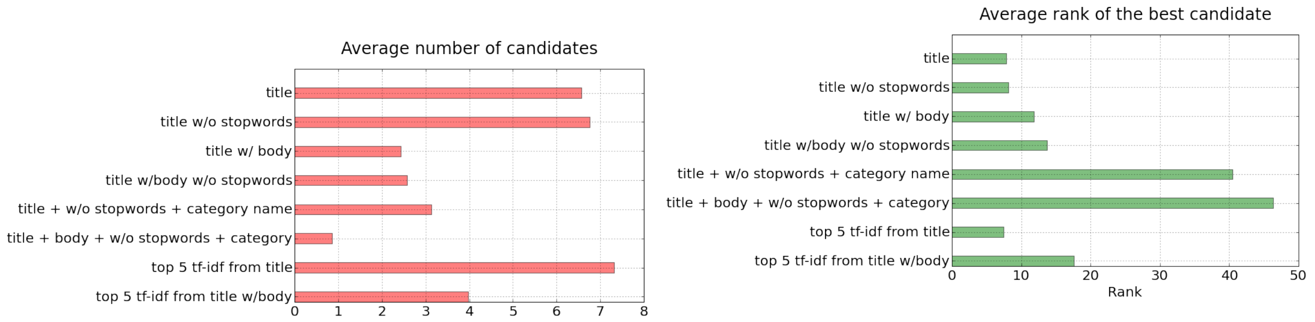
The quality of answers generated from passages built from web search results are lower on average compared to Yahoo! Answers candidates. Figure 4.5 shows the distribution of scores for each of our data sources. Some categories were harder than the other [2] and as we see on Figure 4.5b in some cases web passages were actually more effective than answers to previously posted questions.

The next question, that we analyze is the effectiveness of search query generation strategies. Figure 4.6 plots average number of candidates and the position of the best candidate retrieved by each of the question generation strategies. The longer the search query the less results it retrieved, which is expected, and the lower the quality of the candidates. As a result, in half of the cases the answer returned by our system was retrieved using just top 5 highest IDF terms as the query[12]. For web search we only used 2 query generation strategies, namely question title and concatenation of title with body. Analogously, concatenation of title with body query had lower quality and more often returned few or no results.

Figure 4.7 demonstrates a plot of importances of different features in our answer ranking linear logistic regression model. The feature with the highest weight is category match, but we should note, that this feature is overfitted to the way we build training set (category of the correct answer always matched the category of the question). The next most useful feature is the cosine similarity

---

[12] The same candidate is often also retrieved by other queries



(a) Average number of candidates

(b) Average rank (according to our ranking model) of the best candidate

Figure 4.6: Comparison of different query generation strategies for Yahoo! Answers similar questions search

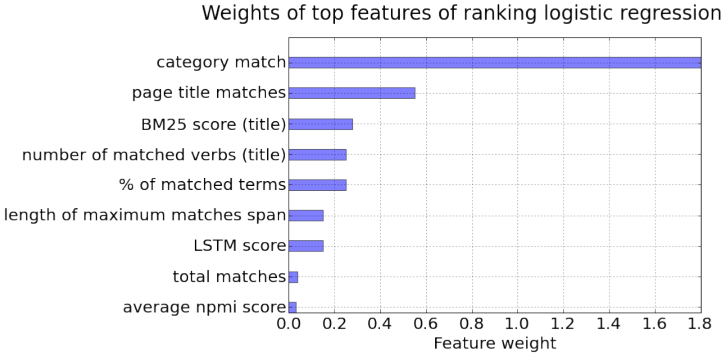


Figure 4.7: Weights of features in answer ranking logistic regression model

between the page title (or question text for Yahoo! Answers) and the current question, followed by BM25 score, number of matched verbs, etc.

I also looked through a small sample of our answers manually. There are a number of typical problems, and one of them is the lack of good question semantic similarity measure. E.g. the question *"Is there a section fro the subject of writing"* in the `Books & Authors` category retrieved a question *"I can't write in the To: Cc: Subject: section"* from the `Yahoo Mail` category. Even though the questions have many terms in common, they are obviously semantically unrelated. Therefore, in future we need to focus more on better question similarity measures.

Answer doesn't have to have many words in common with the question. On the contrary, the maximum possible term overlap will be if a candidate is just a copy of the answer. This was one of the problems for answers, retrieved from the web search results. The way we used to generate the training data didn't include such "artificial" cases, however, they are pretty common in practice. For example, in a number of cases the answer our system chose came from a forum post and instead of selecting the answer posts, the system ranked the question post higher as it had more term matches. The winning CMU team addressed this issue by considering answer-clue pairs, where the clue is supposed to match the question text and the former answers the question. We plan to explore a similar strategy.

4.2.4 Further improvements for TREC LiveQA 2016

TREC LiveQA 2016 task was run on May 31, and I made a few improvements to my system. One of the major quality loss in my previous system was due to random crashes, which resulted in no response for more than 200 questions. This year I changed the architecture of the system to make it more reliable and make sure the answer is returned in time.

Next, as analysis of the previous year track has revealed, many of answers generated using regular web search still came from different community question answering websites. However, the system didn't treat such pages differently from regular documents and didn't extract additional meta-data, such as the question text, *etc.* Therefore, for LiveQA 2016 I decided to extend a set of data sources and in addition to Yahoo! Answers I added separate candidate generation pipelines for Answers.com and WikiHow.com. For these verticals the system uses the search engine built into the platforms, and extracts answers along with the corresponding question meta-data.

The other improvements target the candidate answer ranking module of the system. I've extended a set of features representing each candidate answer to include the following:

- Percentage of the terms in the answer, that are not matched against the question. This features aim at estimating the new information in the answer. The previous system often selected answer that simply repeated the question.
- N-gram matches features: fraction of 1,2-3-grams from the question title, body that overlaps the n-grams from the answer text
- All term match features were computed for related question and body. More specifically, additionally to taking the answer text itself, we added features that compute similarity scores between the current question and retrieved question (title and body). For candidates, generated from regular web documents we tool the title of the page as question title, and the previous paragraph in text as body.

Additionally, I replaced the logistic regression ranking model with LambdaMART [34], as implemented in RankLib library[13]. Unlike the previous year, when training data in a usual sense didn't exist, in 2016 we had relevance judgments from previous campaign. To train the LambdaMART model I took this data and scraped the original web pages to get meta-information and generate all the features for the candidates. Using this data I trained the model to optimize NDCG metric. The results of the evaluation are not available at the moment.

4.2.5 Summary

The pilot year of TREC LiveQA establishes a very good baseline for the future of non-factoid question answering. It confirmed that the task itself is quite challenging as only ∼35% of the questions returned by the winning system had a score of 3 or higher, and there is still a big gap between in the quality of human and machine answers. It will be exciting to see the next version of the task next year, and how the participants will build on this year approaches.

---

[13] https://people.cs.umass.edu/ vdang/ranklib.html

4.3 Proposed Research

The results of TREC LiveQA 2015 established a baseline performance of automatic non-factoid question answering systems, and shed some light on typical problems and future research directions. One particular problem, that I'm proposing to concentrate research in my thesis, is that passages a QA system extract as candidate answers often either contain redundant information, or, on contrary, the answer is split across multiple passages and a single one isn't going to satisfy user information needs. One way to tackle these problems is to introduce an answer summarization module, which instead of returning the top ranked answer, will generate the response by summarizing the relevant pieces of information from ranked candidates. Existing research on answer summarization usually targets community generated answers on CQA websites, which is different from summarizing automatically generated candidates. The developed algorithms typically follow an extractive summarization approach, and explore different answer sentences ranking strategies to either shorten the answer, cover multi-intent questions or to improve diversity [43, 144], and less attention is paid to the overall quality and readability of summarized answer text.

For my thesis I propose to capitalize on the recent successes of deep learning for natural language processing and apply some of the techniques on the new problem of answer summarization. Section 4.3.1 describes the proposed methodology in more details.

4.3.1 Method

Distributed representations for words [126, 138] and longer phrases [112, 101] opened up a new era in natural language processing and information retrieval. Using distributed representations or embeddings one can simply calculate the similarity between words or sentences using various distance metrics in the embeddings space. Such representations made it possible to train end-to-end models for various NLP tasks, which reach or outperform existing state-of-the-art models [46].

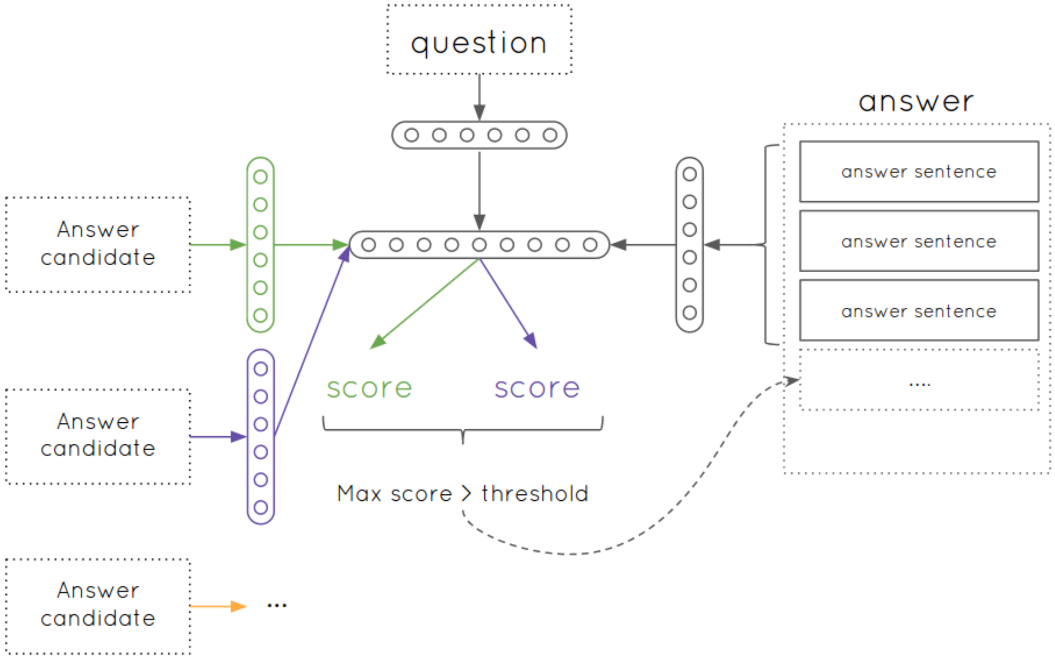
Text summarization isn't an exception, and recently a number of different architectures were proposed for both more traditional extractive [96] as well as abstractive summarization [147, 45]. In answer summarization we are dealing with multiple different piece of text, therefore it's more related to the problem of multi-document summarization [40]. However, unlike document summarization, here we have a notion of the question and some answer candidates (if not the majority) might be totally irrelevant to the question and not worth including overall.

The schematic of the models I propose to implement for answer summarization is depicted on Figure 4.8.

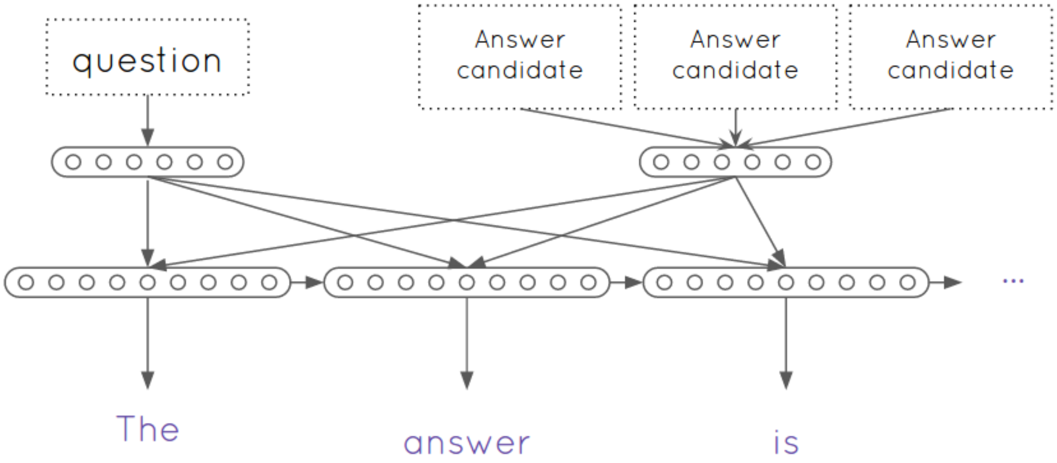
On each step of the extractive summarization the model takes 3 kinds of inputs: embedding of the question, already constructed summary and answer candidate. Embeddings of the question and already constructed summaries can be obtained using the attention mechanism [200] to focus the model on specific parts of the input. The idea of abstractive summarization model is based on the sequence to sequence attention-based models, and similar in architecture to the works of Sumit Chopra and Alexander Rush [45, 147].

4.3.2 Experimentation

Since the problem of answer summarization isn't new, there are some datasets already available and several benchmark results exists. For my experiments I'm planning to use the datasets of M.Tomasoni and M.Huang [174], which was constructed based on Yahoo! Answers archive and manually generated gold summaries, and of Adi Omari et al [133], constructed for the task of


question
answer
answer sentence
answer sentence
answer sentence
....
Answer candidate
Answer candidate
Answer candidate
...
score
score
Max score > threshold

(a) Extractive summarization


question
Answer candidate
Answer candidate
Answer candidate
...
The
answer
is

(b) Abstractive summarization

Figure 4.8: The architecture of the proposed answer summarization models

novelty-based answer reranking. In the later dataset human annotators extracted nuggets of different aspects for the same question, and therefore its possible to judge the relevance of each of the answer with respect to these nuggets. These datasets will be used to test my extractive summarization model. The results on the first dataset could be directly compared to other existing answer summarization techniques, *e.g.* [144, 59, 174]. Thus, this experiment will show if the deep learning based answer summarization model can compete with other approaches as it does for regular document summarization. The later dataset was developed for the task of novelty-based answer ranking, *i.e.* arranging answers in such a way that a user is exposed to different aspects and opinions. On this dataset I will test if sentence-based answer summarization is better (in terms of diversity of covered aspects) than ranking answers under certain maximum character length limit, *i.e.* whether extracting sentences to form an N char length summary gives a better experience than ranking complete answers under the same limit.

However, in these datasets the answer candidates were taken from community generated answers to each question, and therefore on average they are relevant. To test the model on more realistic scenario of automatic question answering, I'm planning to reuse the candidate answers with relevance labels from TREC LiveQA datasets. I will conduct two experiments:

- use the extractive summarization model trained on the previously mentioned datasets to summarize answer candidates
- train extractive and abstractive answer summarization model using community generated answers without human labeling.

The first experiment will help me answer the question if summarizing automatically generated candidates is more difficult than community answers, and whether a model trained for one task can be used for another. The later experiment will test the hypothesis that community generated answer can be used as a ground truth for model training, which would allow one to get millions of examples from CQA archives. The quality of summaries will be judged manually using crowdsourcing for both relevance and coherence/readability.

4.4 Summary

This chapter focus on the problem of improving the performance of non-factoid question answering system. The system I developed participated in both TREC LiveQA 2015 and 2016 shared tasks, which allowed me to test certain ideas and discover directions for future improvements. The thesis will focus on the problem of answer summarization, and proposed extractive and abstractive summarization models, based on recent developments in the field of neural networks.

However, no matter how well an automatic system performs, there will be cases, when it is unable to generate a good response to a user question. In such cases, computers can ask for help, which would lead to a hybrid human-computer question answering systems. Chapter 5 focuses on the idea of crowdsourcing for real-time question answering.

5 Crowdsourcing for Real-time Question Answering

5.1 Problem

Despite the progress in automatic question answering, existing systems are still far from being able to handle every human question. For example, the winning system in TREC LiveQA 2015 shared task were able to return a good or excellent answer to less than 40% of the questions. And overall, according to the TREC LiveQA 2015 assessor scores, for 12.6% of the questions none of the 21 participating systems produced even moderately useful answer, and for 30% a good or better response. This can happens for multiple different reasons, *e.g.* a question is ill formulated, available data sources doesn't contain any relevant information, a system fails to retrieve or rank good answer candidates, *etc.* As conversational agents become more popular, QA systems are increasingly expected to handle such complex questions, and to do so in (nearly) real-time, as the searcher is unlikely to wait longer than a minute or two for an answer.

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 provides some preliminary results and proposed research in crowdsourcing for real-time question answering.

5.2 Approach

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. First, we study if crowd workers can quickly and reliably judge the quality of the proposed answer candidates, and if it is possible to obtain reasonable written answers from the crowd within a limited amount of time. Then, we present CRQA, 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. The crowd input, obtained in real-time, is integrated into CRQA via a learning-to-rank model, to select the final system answer. Our large-scale experiments, performed on a live stream of real users questions, show that even within a one minute time limit, CRQA can produce answers of high quality.

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 tradeoffs in performance, cost, and scalability of using crowdsourcing for real-time question answering?

To answer the first three questions, we conducted a series of crowdsourcing experiments using the Amazon Mechanical Turk platform[1]. We used questions from the TREC LiveQA 2015 shared task, along with the system answers, rated by the NIST assessors[2]. 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[3].

5.2.1 Crowdsourcing for time-constraint answer generation and validation

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

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.

Answer validation experiment

The interface for collecting answer ratings is illustrated in Figure 5.1a[4]. On top of the interface 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

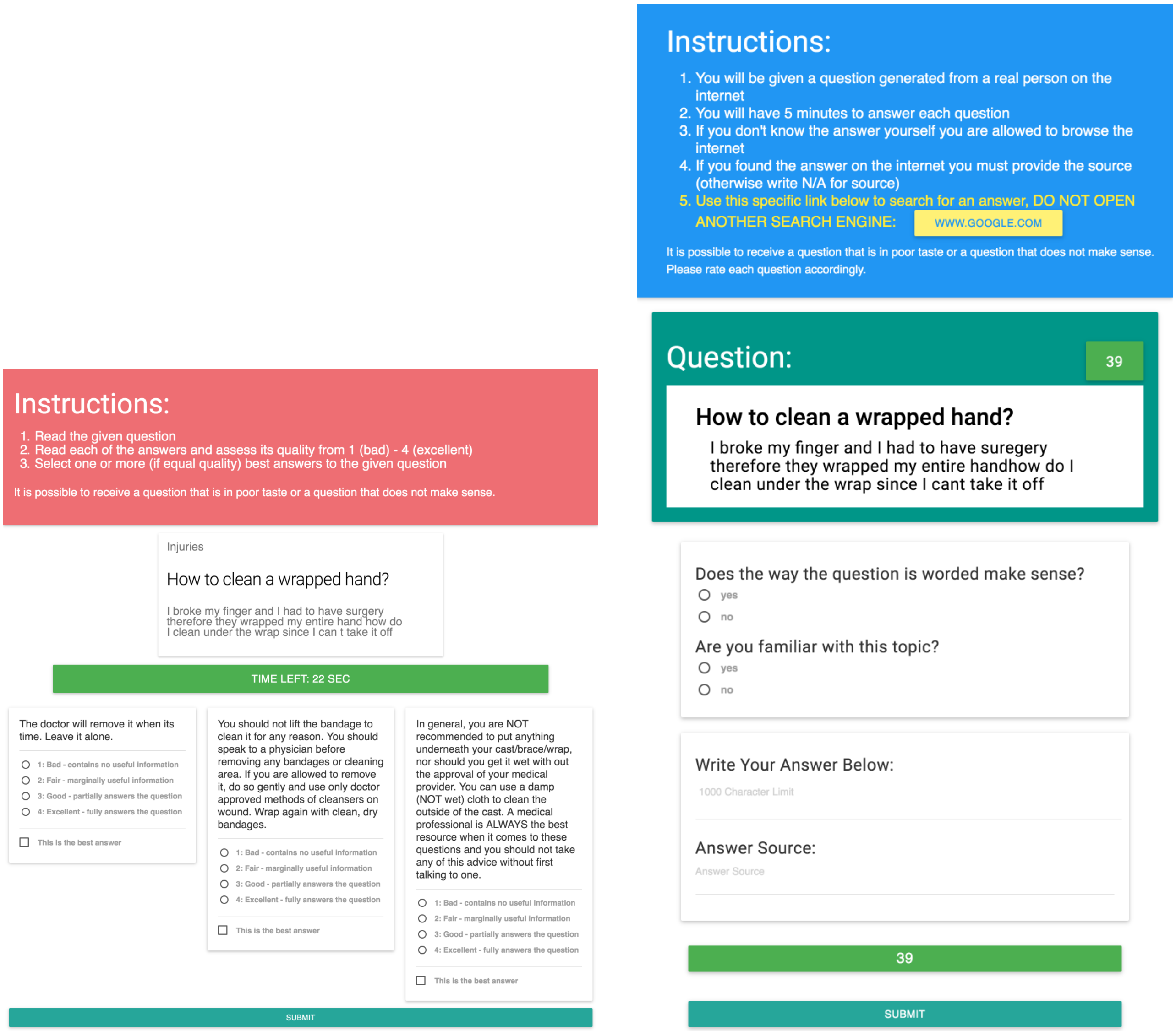
---

[1] http://mturk.com

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

[3] As the answer we took the top question, which was selected as the "Best answer" by the author of the question or by the community.

[4] 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



Instructions:

1. Read the given question
2. Read each of the answers and assess its quality from 1 (bad) - 4 (excellent)
3. Select one or more (if equal quality) best answers to the given question

It is possible to receive a question that is in poor taste or a question that does not make sense.

Injuries

How to clean a wrapped hand?

I broke my finger and I had to have surgery therefore they wrapped my entire hand how do I clean under the wrap since I can t take it off

TIME LEFT: 22 SEC

The doctor will remove it when its time. Leave it alone.

○ 1: Bad - contains no useful information
○ 2: Fair - marginally useful information
○ 3: Good - partially answers the question
○ 4: Excellent - fully answers the question

☐ This is the best answer

You should not lift the bandage to clean it for any reason. You should speak to a physician before removing any bandages or cleaning area. If you are allowed to remove it, do so gently and use only doctor approved methods of cleansers on wound. Wrap again with clean, dry bandages.

○ 1: Bad - contains no useful information
○ 2: Fair - marginally useful information
○ 3: Good - partially answers the question
○ 4: Excellent - fully answers the question

☐ This is the best answer

In general, you are NOT recommended to put anything underneath your cast/brace/wrap, nor should you get it wet with out the approval of your medical provider. You can use a damp (NOT wet) cloth to clean the outside of the cast. A medical professional is ALWAYS the best resource when it comes to these questions and you should not take any of this advice without first talking to one.

○ 1: Bad - contains no useful information
○ 2: Fair - marginally useful information
○ 3: Good - partially answers the question
○ 4: Excellent - fully answers the question

☐ This is the best answer

Instructions:

1. You will be given a question generated from a real person on the internet
2. You will have 5 minutes to answer each question
3. If you don't know the answer yourself you are allowed to browse the internet
4. If you found the answer on the internet you must provide the source (otherwise write N/A for source)
5. Use this specific link below to search for an answer, DO NOT OPEN ANOTHER SEARCH ENGINE: WWW.GOOGLE.COM

It is possible to receive a question that is in poor taste or a question that does not make sense. Please rate each question accordingly.

Question: 39

How to clean a wrapped hand?

I broke my finger and I had to have suregery therefore they wrapped my entire handhow do I clean under the wrap since I cant take it off

Does the way the question is worded make sense?
○ yes
○ no

Are you familiar with this topic?
○ yes
○ no

Write Your Answer Below:
1000 Character Limit

Answer Source:
Answer Source

(a) Answer validation form

(b) Answer crowdsourcing form

Figure 5.1: Crowdsourcing user interfaces

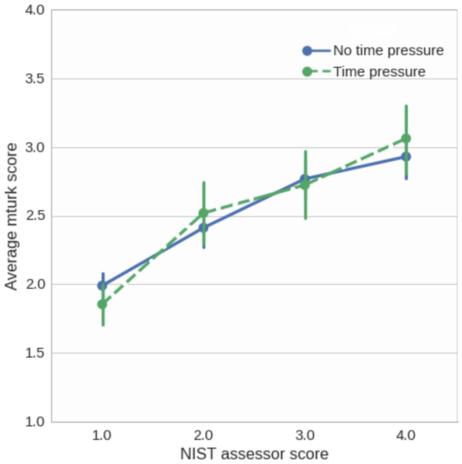


Figure 5.2: Correlation between NIST assessor scores and crowdsourced ratings with and without time limit on the work time

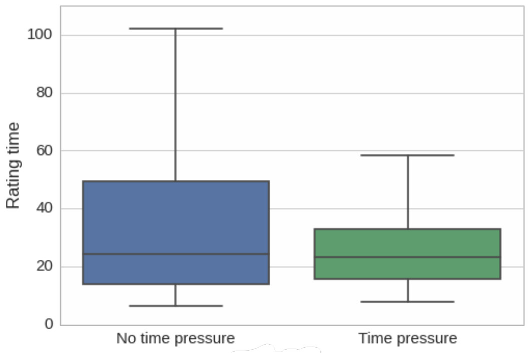


Figure 5.3: Box plot of answer rating time with and without time pressure

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 5.2 shows correlation between official NIST assessor relevance judgments and 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 5.3 shows that even 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. We 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 each answer. The form for answer crowdsourcing is shown in Figure 5.1b, 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[5]. 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[6]. 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 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
- *LiveQA top10* - answers from another top 10 TREC LiveQA'15 system.

Table 5.1 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 don't 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 ($\sim 90\%$) didn't use the web search and provided answers based on their experience, opinions and common knowledge.

---

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

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

Statistic	Y!A	mTurk	mTurk-time	LiveQA'15 winning system
% 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 5.1: Statistics of different types of answers for Yahoo! Answers questions

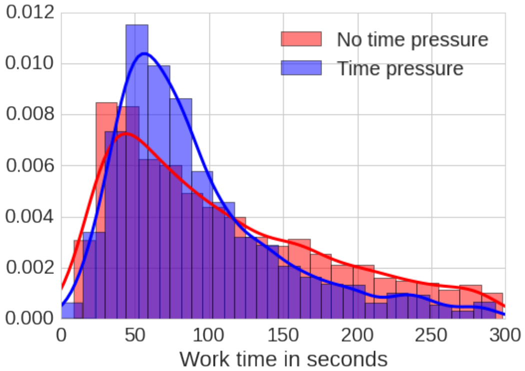


Figure 5.4: Distribution of answering times for experiments with and without time pressure

From Figure 5.4 we can see that adding time pressure shifts the distribution of answering times[7]. The tail of longer work times for no time limit experiment becomes thin with time restrictions and the distribution peaks around one minute.

Answer quality comparison

Finally, to compare the quality of the collected answers with automatic system and CQA responses we pooled together the crowdsourced answers, the answers from the winning and other top-10 LiveQA'15 systems, 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 5.5 displays the plot with average score for answers from different sources. Quite surprisingly the quality of collected answers turned out 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 quality of an average top 10 QA system.

Analysis of the score distribution (Figure 5.6) 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 distinguish 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

---

[7] 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

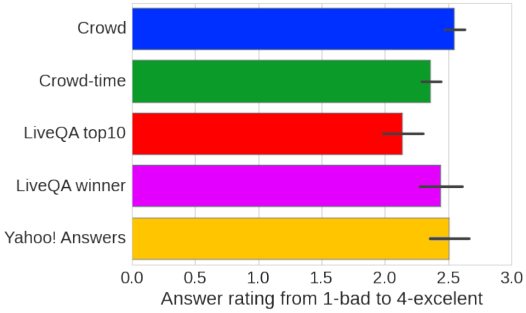


Figure 5.5: Average scores of different types of answers to Yahoo! Answers questions

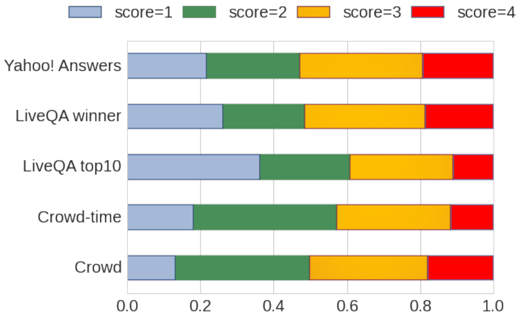


Figure 5.6: Distribution of scores for different types of answers to Yahoo! Answers questions

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 web search to find answers, if they were unfamiliar with the topic; more effective search interfaces may be helpful.

The findings of these experiments were used to implement our CRQA system, 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-rank framework for selecting answers to complex questions. We report extensive experiments of stress-testing the CRQA system, by participating in the TREC LiveQA 2016 evaluation challenge, which provided us with a realistic evaluation setup.

5.2.2 CRQA: Crowd-powered Real-time Automated Question Answering System

Our CRQA system represents a hybrid system, that includes an automated question answering and crowdsourcing modules. The high level architecture is presented in Figure 5.7.

The automated part of the CRQA system follows an Information Retrieval (IR) approach to question answering, and generates a set of candidate answer passages from multiple data sources. After candidates are generated, they are ranked by a trained model, and in the fully automated mode the top candidate could be returned as the answer. 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. Worker contributions are then used by a trained re-ranking

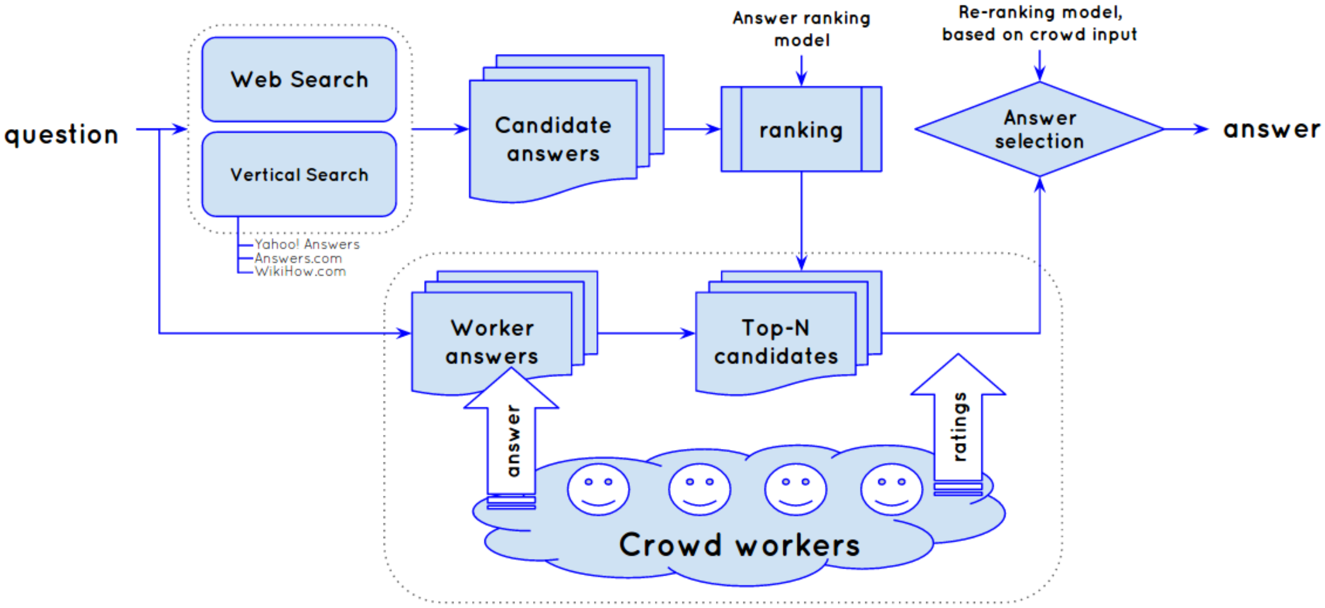

question
Web Search
Vertical Search
Yahoo! Answers
Answers.com
WikiHow.com
Candidate answers
Answer ranking model
ranking
Re-ranking model, based on crowd input
Answer selection
answer
Worker answers
Top-N candidates
answer
ratings
Crowd workers

Figure 5.7: 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

model, that selects the best candidate answers using all the information available. The next two sections describe the architectures of the automated and crowdsourcing modules of our system.

Automated question answering module

When CRQA receives a question, it generates a set of search queries to retrieve a set of relevant documents and extract candidate answer passages. Search queries are generated using the following strategies:

- 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[8]. This strategy targets over-specific questions, which often retrieve few if any search results.

To retrieve a set of potentially relevant documents and extract candidate answer passages, CRQA relies on multiple different generic and CQA document collections. Previous research [158] has shown that many of the user information needs are repeated, and reusing answers to previously posted similar questions is an effective strategy for answering new questions. Therefore, CRQA uses multiple different CQA data sources, which potentially contain a diverse set of questions. However, quite often it is hard to find a similar question in an archive, and many of the information needs are unique. Therefore, we include a web search component, that can retrieve regular web documents, from which our system can extract candidate answers. More specifically, CRQA queries the web using Bing Web Search API[9], Yahoo! Answers, Answers.com and WikiHow.com using their respective search interfaces. For each query we retrieve top-10 relevant documents and question-answer pairs from CQA archives. As candidate answers CRQA extracts answers to the questions retrieved from CQA results and paragraphs of text from the main content of regular web documents, as detected by a method based on [103]. Quite often, it's hard to estimate the quality of a candidate answers from its text only. The task becomes much easier if a system has access to additional data, *e.g.* for the candidates generated from the CQA archives, it is useful to know the text of the original question the candidate answers. For candidates, generated from regular web pages, it's useful to know the topic of the page (*e.g.* from its title), additionally the context of the candidate, such as text that immediately precedes it in the document was shown to provide a useful information for QA. Therefore, besides the text of a candidate answer, we keep some potentially useful metadata, *e.g.,* for QnA candidates we include text and category of the retrieved question, and web page title and text block preceding the answer passage in the document for web-search based candidates. For convenience, we will call the title of a retrieved question or web page *"the answer topic"*, and the body of the retrieved question or the preceding text block the *"the answer context"*.

---

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

Next, for each candidate answer we compute a set of features, described in Table 5.2.

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 5.2: The list of candidate answer ranking features used by the automated module of our CRQA system

The final stage of the module is answer ranking, where a trained LambdaMART [34] model sorts the candidates by their predicted quality. This model was trained using the RankLib library[10] on the data from last year TREC LiveQA task[11], which 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. In a fully automated setup the top candidate could be returned as the final answer to the question, but in this work we explore if we can further improve the performance of the system using crowdsourcing.

---

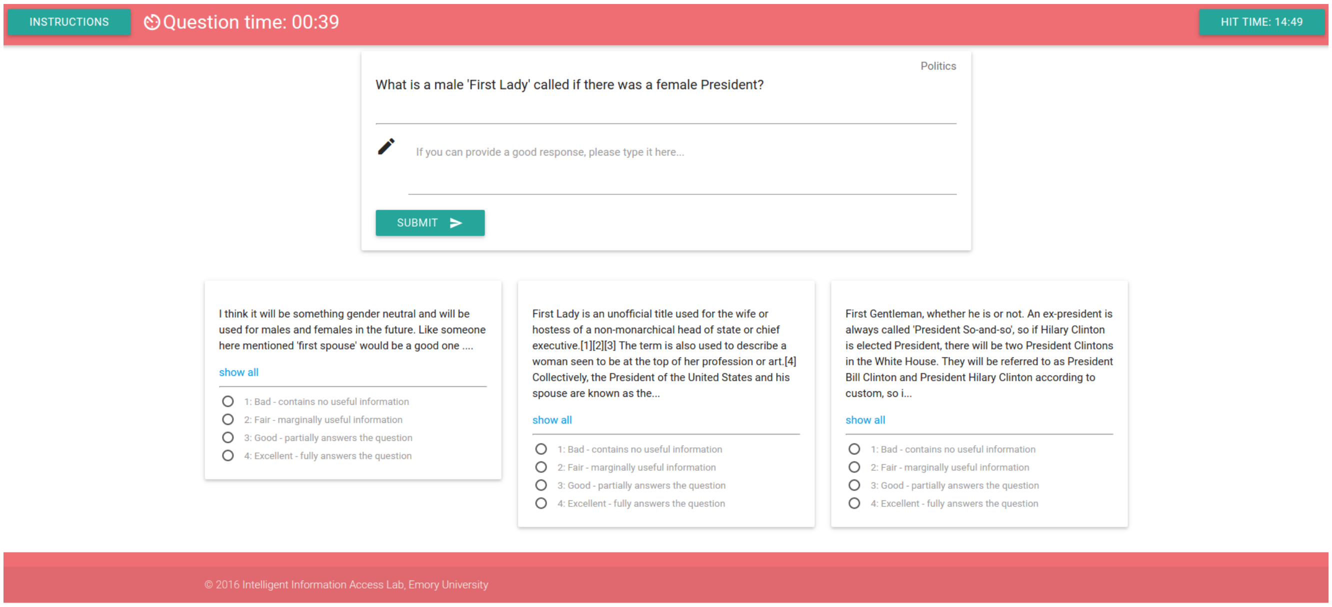
[9] https://datamarket.azure.com/dataset/bing/searchweb

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

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

Crowdsourcing module

As we mentioned before, two of the main problems with the fully automated system responses are lack of good candidates and problems with ranking, therefore, we decided to explore if crowdsourcing can be helpful to overcome these challenges in near real-time scenario. Instead of returning the final answer, the system sends the question and top-7 ranked candidates to the crowd workers and waits for the responses. We chose to give 7 answers based on the average number of rated answers in our preliminary studies. Since systems had only 60 seconds to answer each question, we start a timer when a question arrives, and the system waits to receive all worker contributions until the timer reaches 50 seconds to leave the system some time to generate the final answer and minimize the chances of going above the time limit. CRQA asks workers to write their own answers to questions if possible, which should provide additional candidates in case there are no good automatically generated ones. Additionally, systems expects workers to rate the quality of top-7 ranked candidates, which should help to fix the ranking problems and select a better final answer. Figure 5.8 presents the user interface of our crowdsourcing module.



What is a male 'First Lady' called if there was a female President?

I think it will be something gender neutral and will be used for males and females in the future. Like someone here mentioned 'first spouse' would be a good one ....

○ 1: Bad - contains no useful information
○ 2: Fair - marginally useful information
○ 3: Good - partially answers the question
○ 4: Excellent - fully answers the question

First Lady is an unofficial title used for the wife or hostess of a non-monarchical head of state or chief executive.[1][2][3] The term is also used to describe a woman seen to be at the top of her profession or art.[4] Collectively, the President of the United States and his spouse are known as the...

○ 1: Bad - contains no useful information
○ 2: Fair - marginally useful information
○ 3: Good - partially answers the question
○ 4: Excellent - fully answers the question

First Gentleman, whether he is or not. An ex-president is always called 'President So-and-so', so if Hilary Clinton is elected President, there will be two President Clintons in the White House. They will be referred to as President Bill Clinton and President Hilary Clinton according to custom, so i...

○ 1: Bad - contains no useful information
○ 2: Fair - marginally useful information
○ 3: Good - partially answers the question
○ 4: Excellent - fully answers the question

© 2016 Intelligent Information Access Lab, Emory University

Figure 5.8: User Interface for workers in our Crowd-Powered Question Answering system

The overall algorithm for obtaining crowd input for real-time question answering is the following:

1. When a 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's optional)
3. Otherwise they are waiting for the answer candidates to arrive

4. When a 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[12] 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
   - 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

To hire the workers we used Amazon Mechanical Turk platform[13]. Since the challenge was to run the system "live" over the period of 24 hours, we adapted the "retainer" model to our question-answering task, inspired by the success of this model reported in previous work [20, 24]. Specifically, to obtain an even distribution of workers over the 24-hour period of the TREC LiveQA shared task, we posted 10 tasks every 15 minutes, and they expired after the next set of tasks became available. Since not all assignments were accepted by some worker 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 5.3) 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[14].

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 5.3: Crowdsourcing task instructions, displayed to the user when she first gets to the task

---

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

[13] http://mturk.com

[14] 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

Answer re-ranking and selection. 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 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 [68]. 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 5.4.

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 5.4: The list of features used for answer re-ranking based on crowdsourcing input

CRQA sorts the candidates by the quality score predicted by the model, as returns the top candidate as the final answer.

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 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.

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 5.5: Aggregate statistics of crowdsourcing tasks

Overall statistics are provided in Table 5.5. As we can see, on average workers were able to provide at least one answer for each question, and each of the provided answers got 6 ratings. Since the interface could show only 3 answers at a time, answers had different chances of being rated. To investigate the effect of the order of the candidates posted for ratings on the quality of the final answer, in CRQA for each worker and each question we randomly selected one of the strategies: ordering answer candidates by their model rank or random shuffling of the candidates. The system variant with shuffled candidates for rating was expected to obtain more diverse and comprehensive set of ratings, as we will investigate in the Analysis section.

The official results of the shared task will be available in November 2016 during the TREC conference[15]. Meanwhile, we used traditional (batch-mode) 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, we 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 correlates 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:

- *Automated QA*: automated QA system described in Section 5.2.2.
- *CRQA*: Automated QA system with Crowdsourcing, described in Section 5.2.2
- *Re-ranking by score*: a simplified version of CRQA 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 originated. 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:

- **avg-score**: average score over all questions
- **avg-prec**: average score over all answered questions
- **s@i+**: the fraction of answers with score i or greater (i=2..4)
- **p@i+**: the number of answers with score i or greater (i=2..4) divided by the number of answered questions[16]

Table 5.6 summarizes the performance of the baselines and our system. As we can see, the average score and precision of answers generated by CRQA system is higher than the baseline ranking and even community answers on the Yahoo! Answers platform. However, Yahoo! Answers community answers have higher percentage of *"4 (excellent)"* scores. Figure 5.9 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

---

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

[16] Since for each answer we averaged 3 ratings by different workers, the number of answers with the average score of 4 is low

Method	avg-score	avg-prec	s@2+	s@3+	s@4+	p@2+	p@3+	p@4+
Automated QA	2.321	2.357	0.697	0.297	0.026	0.708	0.302	0.026
Re-ranking by score	2.416	2.421	0.745	0.319	0.031	0.747	0.320	0.031
Yahoo! Answers	2.229	2.503	0.656	0.375	**0.045**	0.737	**0.421**	**0.050**
CRQA	**2.550**	**2.556**	**0.799**	**0.402**	0.034	**0.800**	0.402	0.034
worker ratings only	2.432	2.470	0.750	0.348	0.030	0.762	0.354	0.031
worker answers only	2.459	2.463	0.759	0.354	0.029	0.760	0.355	0.029

Table 5.6: 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

the question. In addition, around 20% of the questions didn't 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 doesn't answer the questions, and therefore rated *"2 (fair)"*. The introduction of the crowdsourcing module allowed CRQA to cover a couple of percent of the questions, for which the automated system wasn't able to generate any candidates, as well as select better candidates when it was possible using crowd ratings.

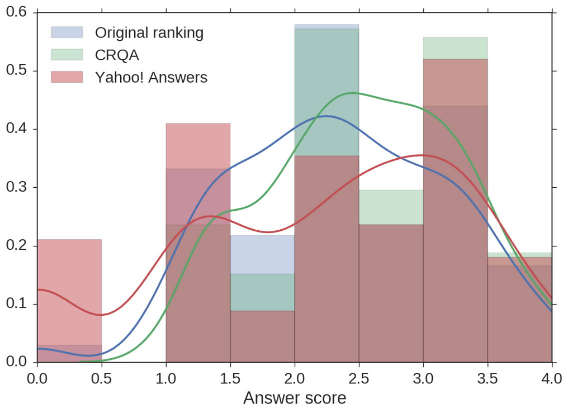


Figure 5.9: Histogram and kernel density estimation of answer scores for original candidate ranking, CPQA model re-ranking and Yahoo! Answers answers

Therefore, we can conclude, that crowdsourcing can effectively help automated QA system to improve the performance of question answering, by providing worker generated answers and rating existing candidates.

Analysis

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 attempts to solve different problems. Table 5.6 shows the performance of our question answering system using each of these types of feedback independently. The results demonstrate that both answers and ratings have 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 doesn't help the system in cases, when it wasn't 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 don't always perfectly answer the question. Table 5.7 gives some examples of worker generated answers with low and high quality scores.

To summarize, ratings of answer candidates and worker generated answers both have similar positive effect on the performance of our question answering system. What is more important, the contributions are independent and therefore it is beneficial to use both of them in the final system.

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 and not have to pay taxes at the end of the year...	W2	1.33
I need help with my mum? Something traumatic happened to me about 4 years ago 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 and the younger kids that lived there called me there cousins and one boy called me his sister?	yes	1.0
Is it bad not wanting to visit your family?	It's nt bad. Just be honest with them. They may be upset but they should understand	3.0
Any health concerns with whey protein? So I workout 3-5 days a week and i drink a whey protein isolate after each workout. Since I workout almost everyday, is it ok for me to just drink a shake everyday?..	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, but I would not be too concerned.	3.0
Foot pain unable to walk? Hi so today woke with some pain, I'm able to put weight on my heel with no problem or pain. But the area between my heel and toes hurts really bad when I try to go with the motion of taking a step. Its not swollen and I don't remember hurting it at all	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 5.7: Examples of answers provided by the crowd workers and their average quality scores

Selection of answer candidate for rating. Predicting the quality of answers and ranking them to select the best is one of the main challenges in automated question answering [169]. External feedback, such as noisy answer ratings, obtained from the crowd workers, provide valuable information, which, as our results demonstrate, can help a QA system to better re-rank the answers. However, the capacity of crowdsourcing for answer ratings are obviously limited, as systems often are dealing with hundreds and thousands of answer candidates for a given question. In this work, we made a choice to rate only top-7 answers according the automated system ranking. This decision was made based on the average number of ratings workers could provide in the allotted time[17]. 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's 2.539 when the candidates are sorted according to their model ranking score. This suggests, that it's beneficial to allocate more of the workers 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 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 certain percentage of workers and removed all contributions (answers and ratings) of others. Figure 5.10 plots the dependency of the performance of our QA system on the number of workers.

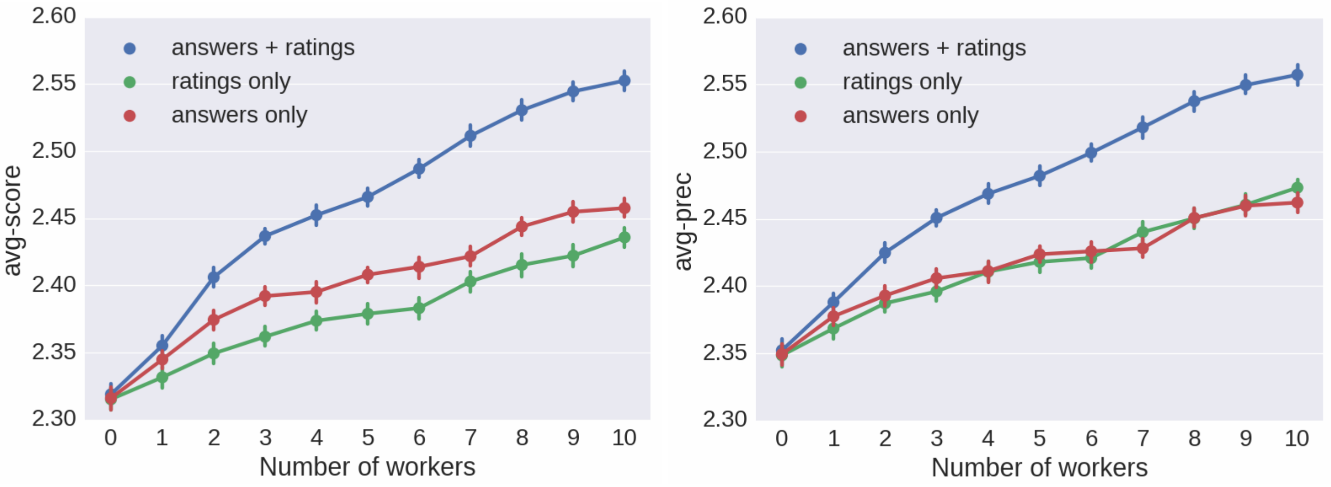
Obviously more workers mean more reliable answer ratings and more answer candidates, which improves the performance of the question answering system. However, we can observe diminishing returns, as the cost per extra gain in performance metrics decreases as the number of workers grow. Half of the overall 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 answering 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 didn't generate enough candidate answers or the predicted quality of the top scoring candidates is low [42, 78]. 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's possible to utilize crowd input for every questions.

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

---

[17] The answers were posted for rating automatically after an automated system was done with candidate generation and ranking. On average users had ~ 35 seconds to provide the ratings.



(a) avg-score: Average score per question

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

Figure 5.10: Plot showing how the quality of the final answer depends on the number of workers per question

suggests that paying 30% of the original cost would give 50% of the performance improvement.

5.2.3 Summary

In this section we presented CRQA, the first, as far as we know, real-time question answering system that integrated crowd work within an automated question answering system. Specifically, we explore different methods of obtaining input from the crowd, and use a machine-learned answer re-ranking model that incorporates the crowd input as features to select the final system answer to return to the user.

We report a large-scale experiment in which over a thousand real user questions were submitted to the CRQA system in real-time, as part of the LiveQA challenge. CRQA was able to successfully answer these questions in under 1 minute, with over 80% of the answers subsequently rated to be fair or better. Importantly, CRQA significantly improved question quality and coverage compared to the starting automated-only system, and, surprisingly, was able to return better answers on average compared to the traditional CQA system with millions of users (Yahoo! Answers) with answers collected more than *two days* after the original posting time.

The described CRQA implementation is a promising step towards efficient and close integration of crowd work and automated analysis for real-time question answering. It raises many promising issues and opens directions for future work, such as selective crowdsourcing for only the questions deemed “difficult” for the automated system; more efficient online learning for obtaining ratings from the crowd and integrating them into the ranking model; and investigating additional features and sources of evidence for improving the joint ranking of the system and crowd input. This paper provides a flexible and powerful framework for combining the powers of crowdsourcing with automated question answering techniques, for building the next generation of real-time question answering systems.

5.3 Proposed Directions for Future Research

The results described above demonstrate how effective can crowdsourcing be for real-time question answering. However, it can also be quite expensive, which is one of the most critical problems for scalability of hybrid system. Such a system needs to have a good strategy to deal with increasing volume and velocity of the input data, *i.e.* user questions. CRQA system, developed to participate in TREC LiveQA cost us \$0.81 per question, which is quite expensive, and according to our analysis in Section 5.2.2 it is possible to get half of the performance gain for 30% of the costs by hiring less workers. However, in general such a strategy isn't agile enough, as some questions might be very easy and system don't actually need any worker input, while some others are very hard and would benefit from more human feedback. In addition, in different situations our hybrid system might need different types of feedback, *e.g.* for some questions some help with answer ranking is more important than additional candidates, and sometimes the opposite.

The research proposed below targets the problem of designing an optimal crowdsourcing plan, *i.e.* allocation of crowdsourcing resources across different tasks. The first step towards this goal is to develop a model for selective crowdsourcing, *i.e.* identification of cases when an automated system would benefit from crowdsourcing the most and when it might be redundant. Next, if a system decides to request feedback from crowdsourcing, we need to decide how much resources to allocate (number of workers and time) and where to target it (what kind of feedback to request from how many users).

5.3.1 Method

First, we need to formulate the conditions and objectives of the problem we are trying to solve, *i.e.* resource and quality conditions, etc. An important problem to focus on in the future research is optimizing the quality of hybrid question answering under the fixed crowdsourcing budget.

A possible approach on the problem is to stick with the real-time scenario, proposed in TREC LiveQA challenge, and set the maximum response time to 1 minute after the question is posted. In such scenario, for the selective crowdsourcing problem one can train a machine learning model to predict the expected performance of the answer, given all the current candidates. This problem is very similar to search query performance prediction [42, 78, 221], and therefore we can adapt the ideas developed in this line of research. More specifically, a machine learning model (*e.g.* random forest or gradient boosted regression trees) can be trained to predict the expected relevance score of the answer a system is about to return, given the following set of features:

- single answer features: answer length, information retrieval score given the question text and other features used for answer ranking
- results list features: syntactic and semantic similarity of the top answer to other retrieved candidates
- corpus dependent features: query clarity score [50], which measures the coherence of the retrieved results (answer candidates) with respect to the whole corpus

In the simplest case, the score, returned by this model, can be thresholded to determine if the answers is expected to be good and no crowd feedback is needed, or if the system would benefit from worker contributions. Alternatively, it is possible to experiment with adaptive thresholding, which can be implemented by putting all the questions into a single priority queue, based on their expected quality from lowest to highest. The workers will be assigned to the tasks from the top of the queue, *i.e.* starting from the lowest expected performance. Decision on how many workers assign to which tasks depends on many factors, such as total number of workers available, number and expected performance of other questions expecting crowdsourcing feedback in the queue. Therefore, to design a resource allocation strategy I will train another machine learning model, that will predict how much workers to assign to each question, given the total number of workers and information about the other questions waiting in the queue, *i.e.* expected performances of 2nd, 3rd, ... questions in the queue, waiting time for the other questions in the queue, *etc.* Such a model will allow a system to adapt not only to varying number of workers available, but also to changing velocity of the arriving questions.

Finally, another important aspect of crowdsourcing for real-time question answering is how much time do we need to give to workers in order to receive enough feedback. To answer it, we can analyze the data we already collected and determine the dependency of performance gain vs time workers spend on the task, which can be done by filtering out feedback items received after certain time threshold.

5.3.2 Experimentation

To train the models and conduct the analysis of the work proposed in the previous section we can use the crowdsourcing data we already collected during TREC LiveQA 2016. The dataset includes 1088 questions, top 7 automatically and worker generated answers, each of which was judged on a scale from 1 - 4 based on its relevance to the question. This setup allows us to apply different reranking and filtering strategies and estimate the final quality of the answer our hybrid system would return.

To train the selective crowdsourcing model the dataset can be split into training, development and test sets, and the training portion can be used to build a regression model to predict the relevance of top answer given the question and other candidates, represented by a set of features described above. To test this model, we can apply it on a test set and compute Pearson and rank correlation coefficients. In addition, by varying the threshold we can save certain percentage of crowdsourcing resources by discarding the corresponding feedback data and draw the performance vs crowdsourcing cost plot.

Finally, to model a real scenario of questions arriving non-uniformly in time, we can modify the dataset and set question posting times based on some random process, *e.g.* Poisson point process. A couple of different datasets can be generating to have different properties (*e.g.* peak load) and we can conduct the experiments on all of them. Each dataset will be split by time into training and test, and I will train the resource allocation model (predicting how many workers to allocate to the task) on the training set and use it on training set. The output of the model will be used to subsample the feedback of the users we already have, *e.g.* if the model suggests to use 4 workers, we can sample them from a total pool of 10 workers we already had for this task. Unfortunately, this experiment does not allow us to go beyond 10 workers per question, but this scenario is quite expensive anyway, therefore it seems more reasonable to explore only plans where we don't need to go beyond 10 workers. In addition, in the original experiment there was no need to share workers between tasks, as all of the tasks arrived sequentially, while in real scenario we will have more than one question requiring worker feedback.

5.4 Summary

This section described my prior research and proposed some additional work in developing a crowdsourcing module for real-time question answering. I developed a crowd-powered automated question answering system, called CRQA, that participated in TREC LiveQA 2016 shared task and by preliminary results significantly improved the performance compared to fully automated scenario. Additional research I proposed targets mostly the scalability of the system, *i.e.* reducing total costs by applying crowdsourcing selectively to questions that would benefit from it the most and allocating the number of workers according to the total available resources and the volume of the incoming questions. This work will be included into my thesis if time permits.

The next section will look into another important aspect of question answering, namely interactions with the user. Despite our best effort to cater to the user information needs, they may be expressed in an ambiguous way or miss some important details, which makes it simply impossible to answer. In other cases, complex informational tasks may be reduced a number of easier questions, which a system can solve automatically. To handle all these situations we need to study and improve the ways a question answering system communicates with the user.

6 User Interactions with Question Answering Systems

6.1 Problem

Most of the research has considered question answering as a one side communication process: a user issues a question and a computer system provides an answer. Thus, the only input a system receives is the text of the question and the only output is the text of the answer. This setup is quite limited, because it doesn't provide any means for the user to affect the behavior of a question answering system except by issuing new questions. Similarly, it doesn't allow QA systems to request any additional information from the user. However, with the growth of mobile personal assistants and conversational agents the popularity of dialog-based interface increases, which opens up new opportunities for question answering.

TREC complex interactive question answering (ciQA) track, which were run in 2006 and 2007, was designed specifically to motivate research in this area. In this task participant system could request some input from assessors before submitting the final answer. However, the setup of the task wasn't exactly based on a dialog between systems and their users. The most typical approach to the task was to give assessors a set of sentences or passages to judge as relevant/irrelevant and use this feedback to improve the quality of the final answer. Such scenario is quite problematic to implement in todays personal assistants and chat bots, as they require relatively significant effort and time from users.

In my thesis I'm going to focus on improving the user success solving complex informational tasks in a question answering dialog scenario. Section 6.2 describes my prior research on assisting users by providing strategic search hints, designed to help split a complex task into smaller ones that an automatic system will likely handle much better. And section 6.3 proposes additional research to incorporate user feedback into a question answering system in a chat bot scenario.

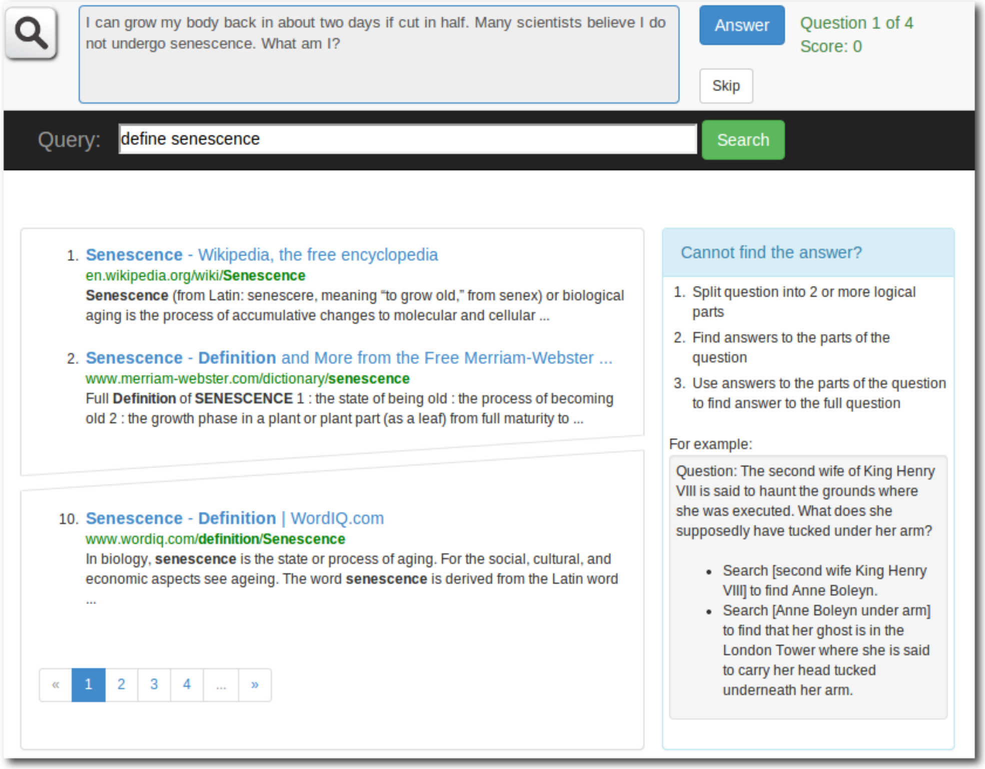
6.2 Approach

6.2.1 Search Hints for Complex Informational Tasks

Search engines are ubiquitous, and millions of people of varying experience use them on daily basis. Unfortunately, not all searches are successful. Bilal and Kirby [25] reported that about half of the participants of their user study felt frustration when searching. Xie and Cool [199] 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[1] 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. [131] demonstrated that

---

[1] http://www.powersearchingwithgoogle.com



I can grow my body back in about two days if cut in half. Many scientists believe I do not undergo senescence. What am I?

Question 1 of 4
Score: 0

Query: define senescence

1. **Senescence** - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Senescence**
Senescence (from Latin: senescere, meaning "to grow old," from senex) or biological aging is the process of accumulative changes to molecular and cellular ...

2. **Senescence** - **Definition** and More from the Free Merriam-Webster ...
www.merriam-webster.com/dictionary/**senescence**
Full **Definition** of **SENESCENCE** 1 : the state of being old : the process of becoming old 2 : the growth phase in a plant or plant part (as a leaf) from full maturity to ...

10. **Senescence** - **Definition** | WordIQ.com
www.wordiq.com/**definition**/**Senescence**
In biology, **senescence** is the state or process of aging. For the social, cultural, and economic aspects see ageing. The word **senescence** is derived from the Latin word ...

Cannot find the answer?

1. Split 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:

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?

- Search [second wife King Henry VIII] to find Anne Boleyn.
- 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.

Figure 6.1: 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, 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, we chose 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 hints can improve search success rate. In contrast, generic search hints, which were too general and harder to follow, had negative effect on user performance and satisfaction.

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"[2] and uFindIt [1]. Participants were hired using Amazon Mechanical Turk[3].

The goal of the web search game used in the user study is to find answers to several questions with the provided web search interface (Figure 6.1). 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 popup box notifies a player if the answer is incorrect (since the answer can be formulated differently, 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[4] 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.

---

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

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

	Question	Correct Answer	Specific hints
Task 1	I can grow body back in about two days if cut in half. Many scientists think I don't 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
Task 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 didn't 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
Task 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"
Task 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 6.1: Search tasks used for the study, and specific search hints shown to one of the user groups

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 with 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 6.1 and were presented to all participants in the same order to ensure comparable learning effects.

---

[4] http://www.bing.com/toolbox/bingsearchapi

The questions have multiple parts and to solve them it is helpful to search for answers to parts of the questions and then combine them. In one of the previous studies we observed, that most of the users didn't 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 6.1). 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 [131], each user was assigned to one of the three groups:

1. users who didn't get any hints
2. users who got task-specific hints
3. users who got the generic hints

Results

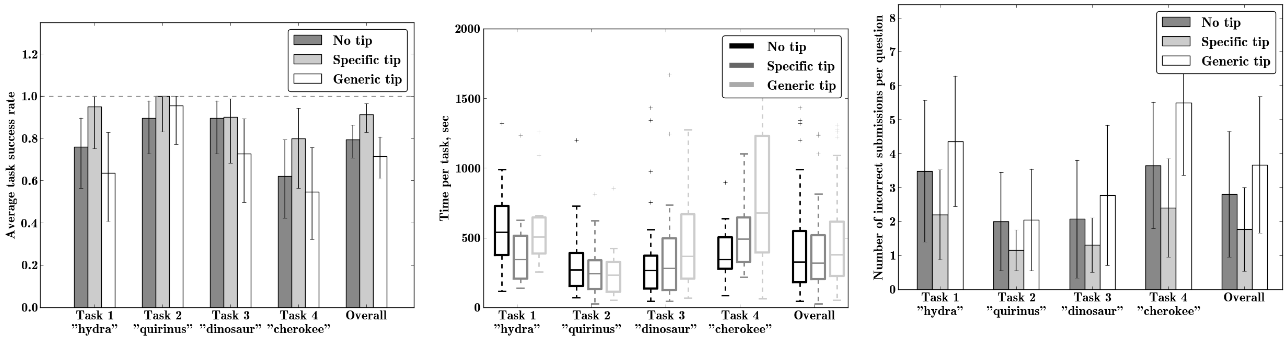
From 199 unique participants, who clicked the HIT on Amazon Mechanical Turk only 90 players finished the game. We further examined all games manually and filtered out 9 submissions for one of the following reasons: lack of effort (e.g. skipped several tasks after none or a single query) or usage of external resources (e.g. the answer was obtained without submitting any queries or results explored didn't contain the answer). Furthermore, 10 players from the group which received hints indicated in the survey that they didn't 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[5]. Figure 6.2a shows that success rate is higher for users who saw task-specific hints compared to users who didn't get such assistance. Surprisingly, having the generic hint decreased the success rate, although users could easily ignore a hint they didn't like. A possible explanation is: generic hints were harder to follow and users who tried and failed became frustrated and didn't restart their searches.

---

[5] 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

The plot of average time to answer a question on Figure 6.2b doesn't 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 6.2c 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 less 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) Success rate per task for each group of participants

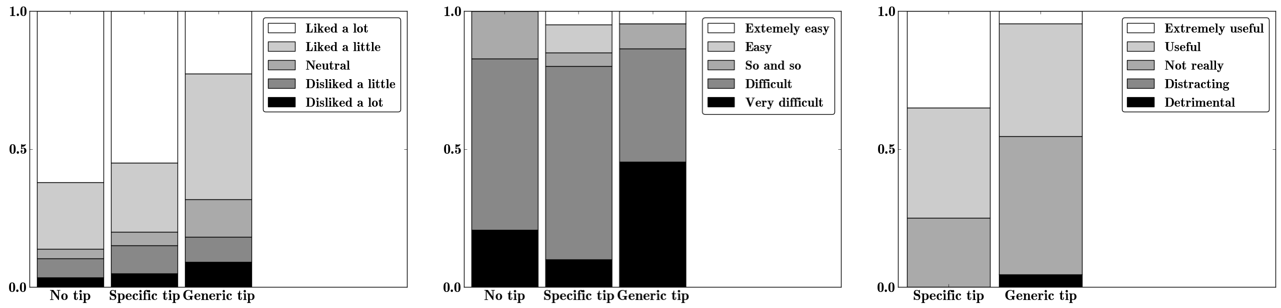
(b) Task completion time for each group of players

(c) The number of incorrect submission attempts per question for all groups of users

Figure 6.2: Results of the user study on the effectiveness of strategic search tips on search task success rate

We also looked at other search behavior characteristics: number of queries submitted, number of clicks made, 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 6.3 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 didn't 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 6.3c).



(a) How did you like the game? (b) How difficult was the game? (c) Were search hints useful to you?

Figure 6.3: Proportions of replies to some of the survey question for each group of users

To summarize, the results of the conducted user study suggest that specific search hints can be helpful, which is indicated by higher success rate, lower number of incorrect attempts and positive feedback in the end of study survey. In contrast, generic hints can have negative effect on user experience, which is indicated by lower success rate, increased number of incorrect attempts and higher perceived tasks complexity according to the survey.

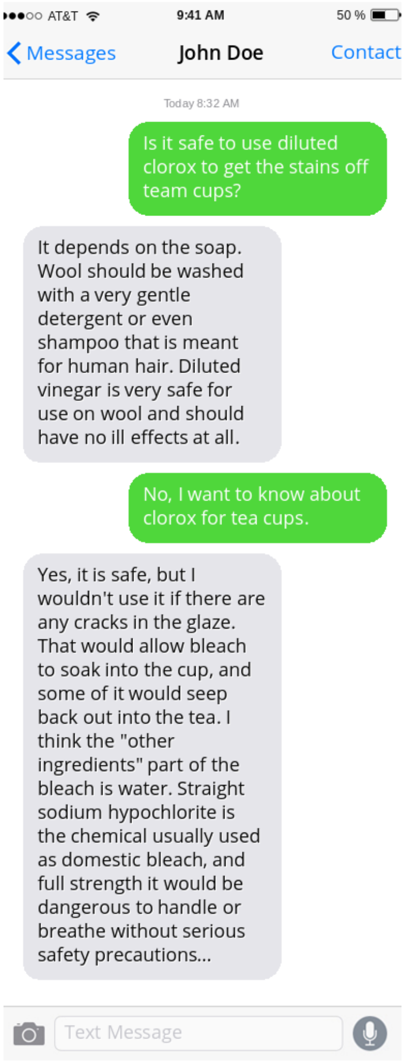
Summary

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.

Even though strategic search hints can improve success rate for complex informational tasks, they represent a rather one-way form of communication, where a system doesn't accept any feedback from the user and doesn't adapt to it. Hints essentially outsource all the heavy-lifting of decision making to the user, which often makes the experience less enjoyable, as demonstrated by the post-study survey results. An alternative strategy is to let the system adapt based on the implicit and explicit feedback from the user. Such a scenario looks even more compelling given a proliferation of personal assistants and chat bots, which makes it easy for users to respond to the user answer with either positive or negative feedback message.

6.3 Proposed Research

Question answering is one of the major components of modern personal assistants like Apple Siri, Amazon Alexa and Microsoft Cortana. However, these systems still operate in one sided communication mode, where a user formulates queries and assistant responds with an answer or search results. In my thesis I propose to incorporate feedback from the user to improve the performance of the question answering system. To make it more concrete, consider an Example on Figure 6.4. It's not uncommon for a question answering system to be incorrect. Web search engines typically decide whether to show direct answers or not based on the confidence in the



Is it safe to use diluted clorox to get the stains off team cups?

It depends on the soap. Wool should be washed with a very gentle detergent or even shampoo that is meant for human hair. Diluted vinegar is very safe for use on wool and should have no ill effects at all.

No, I want to know about clorox for tea cups.

Yes, it is safe, but I wouldn't use it if there are any cracks in the glaze. That would allow bleach to soak into the cup, and some of it would seep back out into the tea. I think the "other ingredients" part of the bleach is water. Straight sodium hypochlorite is the chemical usually used as domestic bleach, and full strength it would be dangerous to handle or breathe without serious safety precautions...

Figure 6.4: An example of a dialog-based question answering system, that accepts feedback from the user and returns an updated answer

answer [208]. However, in a dialog a system needs to provide some kind of response, and why not take a chance and return the top candidate answer. If the user doesn't like the answer, she can respond accordingly, and this feedback needs to be taken into account by the question answering system, which can then re-rank the candidate answers and provide an alternative response, that will hopefully satisfy the user.

Relevance feedback has been studied extensively in the information retrieval community for ad-hoc document retrieval [150, 146, 190], but not for question answering. Thus, I propose to explore the methods for relevance feedback in question answering.

6.3.1 Method

In my thesis I propose to focus on both positive and negative relevance feedback for question answering. A user might give a system a positive feedback if the provided answer was relevant, but either alternative view or additional information is needed. Alternatively, if an answer is not helpful, a system can take negative feedback to try to generate a good answer.

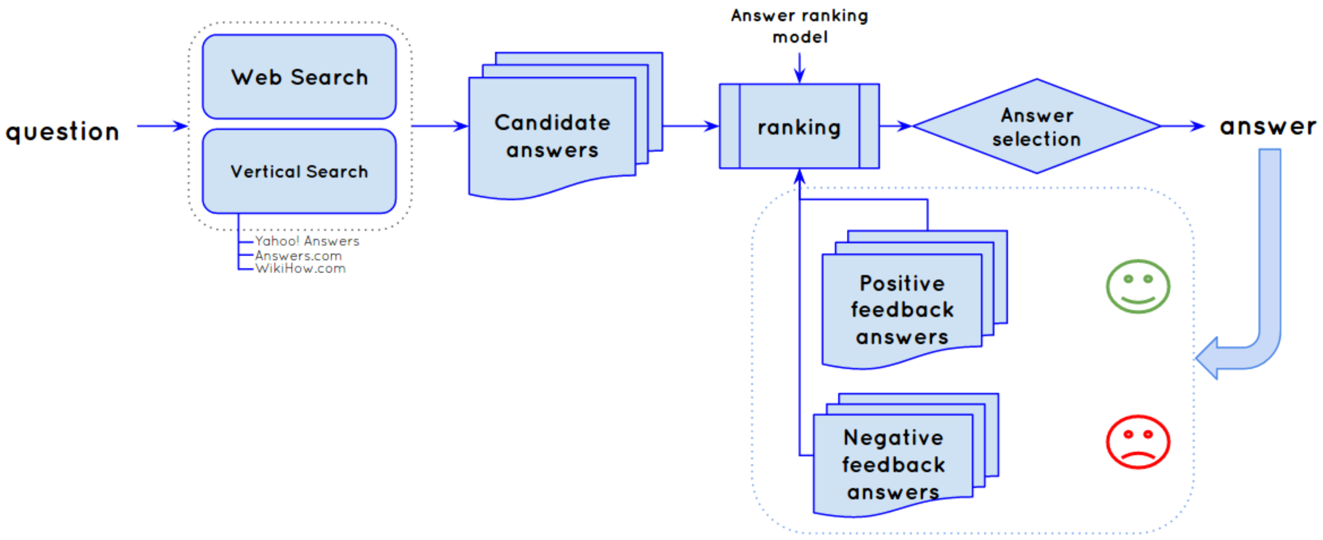


Figure 6.5: Architecture of the question answering system, that incorporate positive and negative relevance feedback

As a baseline for the experiments I'm planning to take the system I developed for TREC LiveQA shares task, described in Section **??**, and extend it with the relevance feedback module. This module will allows the system to adjust and return an alternative answer given either positive or negative feedback. I'm going to integrate relevance feedback into the system's candidate ranking model. More specifically, a set of features representing a candidate answer will be extended with various scores, that measure syntactic and semantic similarities between a candidate and positive and negative feedback examples. This additional set of features will include various vector space, distributed semantics [107] and language model similarity scores [190], as well as some more simple statistics, such as common and missing n-grams, longest common substrings, *etc.* The overall architecture of the proposed system is sketched on Figure 6.5.

There are other alternative stages in the QA pipeline, where we could incorporate the relevance feedback, *e.g.* a system can adjust a set of search queries used to retrieve a set of candidates as done for ad-hoc information retrieval [146], but I will have to leave this for future work.

6.3.2 Experimentation

To test the system experimentally I propose to reuse the data from TREC LiveQA 2016, which we collected for the crowdsourcing experiment (see Section 5.2.2). As a remainder, during the official run of TREC LiveQA 2016 we collected 1087 questions, issued to our system. For each question, top 7 generated candidate answers were rated by crowd workers on a scale from 1 (bad) to 4 (excellent).

To test the proposed relevance feedback model I'm proposing to conduct the following simulation experiment. For each question, top 7 answers will be arranged according to their model score. Assuming the model returns the top candidate, we collect cases when the top candidate has a low relevance score (*e.g.* $score < 2$), and use these answers as negative feedback. The goal of the model will be to re-rank the rest of the candidates and return a better answer. A baseline in this experiment will be a system, that doesn't use the feedback and simply returns the next best answer according to the model score. Positive relevance feedback experiment will be conducted in a similar way: all answers with good relevance score (*e.g.* from 2 to 3) will be given back to the system as positive feedback with the goal to return another good answer. This experiment will help us confirm or reject the hypothesis, that it's possible to incorporate relevance feedback for question answering.

To train our ranking model to accept additional answer candidates I will split the original dataset into training and test parts again. Using the original model I will collect cases with negative and positive feedback as described above, and train the model to retrieve the best possible candidate from the rest of the answers.

However, such simulation experiment won't allow us to test if relevance feedback improves the user experience. To answer this question we will conduct a user study using either Mechanical Turk or hire some student workers. We are building a chatbot for Facebook messenger, that will serve as a front-end to our question answering system. The users will receive a set of questions, e.g. sampled from Yahoo! Answers, and will need to get a satisfactory answer using our chatbot. All the users will be split into two groups: with and without an option to leave relevance feedback. The relevance feedback group will be asked to mark each answer as good or bad, after which a new candidate will be provided until the user will decide to stop and declare either fail or success. The group without relevance feedback will be given an option to receive another answer, which will simply be the next candidate in the original ranked list. Upon completion of all tasks, the users will complete the questionnaire, where I will ask questions about their experience and satisfaction with the chatbot. Task success rate and the feedback will help me answer the question if relevance feedback for question answering improves user experience.

6.4 Summary

In this chapter I described my prior and proposed research in improving user experience with question answering systems. Strategic hints, that a system might display to help the user better split the original problem, were shown to be able to improve the success rate, however they need to be designed carefully. Hints, that are too generic and hard to follow can be distracting and detrimental to the user experience. However, such hints put all the hard work on the user, who will go through the trial and error process until a system is able to retrieve a good answer. The second part of the section described the proposed research in engaging in a dialog with the user, and using these interactions to receive relevance feedback, which can be used to refine the answer. I believe that this feedback is one of the first step, that can lead to deeper understanding of user information needs through dialog.

7 Summary & Research Timeline

This chapter summarized the research I propose to do in my thesis to improve different aspects of question answering.

7.1 Research Objectives

The main research objectives I plan to target in my thesis are the following:

1. RQ1. How to effectively combine unstructured text and structured knowledge base data to improve factoid question answering?
2. RQ2. How to improve candidate retrieval and answer generation for non-factoid question answering?
3. RQ4. How to use crowdsourcing to improve question answering performance in near real-time scenario?
4. RQ3. How to utilize the dialog between the user and question answering system to improve users success rate in solving informational tasks?

7.2 Research Plan

7.2.1 Combining KB and Text Data for Factoid Question Answering (Chapter 3)

I propose to explore semantic annotation of entity mentions as a way to bridge the gap between KB entity graph and text documents. Such representation will allow us to do simple string matching on text documents and at the same time explore the knowledge about the mentioned entities in KB and vice versa. The idea of the proposed approach is based on extending a set of links in a knowledge base with edges connecting entities to their mentions in text documents. Therefore, text fragments, that mention two or more entities, essentially create additional edges in the knowledge base connecting these entities, and these edges are “labelled” with textual representation of the fragment (e.g. bag of words or embedding). Section 3.3 described the proposed approach in more detail.

Expected contributions

- A novel model for relation extraction from archives of question-answer pairs.

  The developed model allows to extract relational triples for KB completion from a novel data source. The experiments demonstrate, that together with existing sentence-based relation extraction techniques it is now possible to get more information, which eventually benefits the performance of knowledge base question answering and other downstream KB applications.

- New hybrid KB-Text question answering approach, that improves knowledge base question answering by using information from unstructured text data sources, annotated with KB entity mentions, which essentially introduces a new types of edges into a knowledge graph

  The proposed model has a potential to overcome many of the issues of KB and text question answering, and allow to answer complex factual questions, that couldn't be answered before. This research could be useful for future work on integrating different unstructured, semi-structured and structured data sources for joint reasoning in question answering and other applications.

Risks

- The dataset I propose to derive from CQA data might be hard to build due to potentially low number of questions, that survive the filtering. In case this happens I will have to use one of the other existing datasets, such as TREC QA.
- Abundant textual information might add a lot of noise to our knowledge graph extended with text-based edges. More connections means more candidates for our search algorithm, and increased level of noise might be hard to deal with in this scenario, which could actually drop the performance compared to clean, but incomplete knowledge base only model.

7.2.2 Answer Summarization for Non-factoid Question Answering (Chapter 4)

In Section 4.3 I proposed to develop answer summarization techniques, inspired by the recent advances in deep learning for text summarization [147, 45] and generation [97]. More particularly, I'm planning to test both extractive and abstractive text summarization approaches on the task of generating responses to non-factoid questions.

Expected contributions

- An open source non-factoid question answering system, designed to participate in TREC LiveQA shared task, and which can be used as a baseline in various experiments in improving different aspects of the QA process.

  The system I developed was ranked top-7 in TREC LiveQA 2015 shared task (the results of the 2016 track aren't yet available).
- A novel answer summarization module for non-factoid question answering system. Unlike some prior work, which focused on summarizing answers posted by different users on CQA platforms, the model I propose to develop will operate inside a real question answering system, which means it will have to deal with some additional challenges, such as higher rate of irrelevant passages. This model will be useful as a first step towards more sophisticated answer distillation techniques [130].

Risks

- The amount of training data available for the task isn't that big compared to some other datasets, used to train deep learning models. There is a risk, that it won't be enough to find the optimal parameters for the model and beat some traditional techniques

7.2.3 Crowdsourcing for near Real-time Question Answering (Chapter 5)

The developed near real-time question answering system showed significant improvements in performance compared to a system without crowdsourcing. As a future research I propose to optimize the costs of this approach. To make our system more scalable and cut expenses associated with the crowdsourcing module, I propose to incorporate a model to predict the expected performance of the answer, generated by the automated system. This score can be used to prioritize tasks when the load is high, and even skip some tasks altogether, if the expected quality of automatically generated answer is high.

Expected contributions

- New method for answer collection and rating using crowdsourcing for a near real-time question answering system.

  I believe that this piece of work could be very useful for future research on the interaction between automated QA and dialog systems and human experts, which could be essential to modern intelligent assistants. Existing systems works great for a subset of user tasks, but encounter certain problems for a different subset. To provide a smooth user experience it might be crucial to have a fall back option of human experts, who can help systems to overcome the challenge.

- A novel hybrid question answering system, that incorporates crowdsourcing, but still operates in near real-time, *i.e.* providing a response within one minute after the question is posted. This is much faster than on existing community question answering platforms, where a quarter of the questions are left completely unanswered.

Risks

- High overall costs of the developed crowdsourcing module. In the current version of our hybrid question answering system, developed for TREC LiveQA 2016 the cost of a single question turned out to be $0.81, which is quite a lot. The proposed research targets the problem of reducing these costs by applying crowdsourcing selectively, but there is a risk that the cost of getting a significant quality improvement will still be high to make our system practically useful.

7.2.4 User Interactions with Question Answering Systems (Chapter 6)

As one of the first steps towards a more intelligent question answering dialog I propose to develop methods for positive and negative relevance feedback. After a system returns an answer, a user might respond with certain feedback, whether positive (*e.g.* “give me more information on the topic”) or negative (*e.g.* “this answer is bad”). The system needs to take this feedback into account and come up with a better answer. To implement this functionality I propose to extend a set of features representing each answer candidates with features, measuring similarities and dissimilarities between the candidate and provided positive and negative feedback answers. More details on the architecture of the proposed approach are given in Section 6.3.1.

Expected contributions

- A study of the effect of strategic search hints on the user experience and success rate for complex informational tasks.

  The results of this work suggest, that well designed search hints can have a positive effect on users struggling with a complex search task. These observations provide some insights for future research in user assistance for complex information needs.

- A novel method for positive and negative relevance feedback in a question answering dialog scenario. I believe that incorporating user feedback is a first step towards a richer dialog between a computer system and a user. In future, it's possible to extend a set of dialog actions and allow system ask clarification questions or confirm certain pieces of information.

Risks

- Negative relevance feedback is usually harder to implement that positive feedback [190]. Negative feedback tells the system some information on what user didn't like. However, the space of irrelevant information is much larger than relevant, *i.e.* another candidate answer might be different from the one that received negative feedback, but still doesn't provide any useful information. Therefore, if in our scenario bad questions turns out to be quite diverse, negative feedback might not work efficiently, and the overall user experience would suffer.

7.3 Research Timeline

A tentative timeline for the work that needs to be done is shown below:

- Joint model for question answering over KB and text (Section 3.1): *8/2016 - 9/2016*
  1. collect and index textual data about entity mentions from ClueWeb12 dataset
  2. derive a factoid question answering dataset from Yahoo!Answers WebScope collection
  3. build a joint KB and text question answering system
  4. test it on existing (TREC QA) as well is derived datasets and compare the results with the baselines
- Answer summarization for non-factoid question answering (Section 4.3): *9/2016 - 10/2016*
  1. build a model for extractive summarization for non-factoid question answering
  2. if time permits, build a model for abstractive summarization for non-factoid question answering
  3. test the model on Yahoo! Answers datasets from [133, 174]
  4. integrate the model into my TREC LiveQA system, test it on a set of questions and manually judge the quality compared to a single returned answer.
- Relevance feedback for dialog-based question answering (Section 6.3): *9/2016 - 11/2016*
  - implement features to measure similarity between an answer candidate and negative and positive feedback answers

  - train candidate answer ranking model, that incorporates relevance feedback if available
  - test the model on the data from TREC LiveQA 2016
  - conduct a small user study to see how the system behaves in a real scenario and whether or not it improves the user experience

- If time permits, optimizing crowdsourcing for near real-time question answering (Section 5.3): *11/2016*

  1. build a model to predict the expected quality of automatically generated answer candidate
  2. develop a task scheduling system based on priorities estimated in step 1, number of workers available and the current load of the system.
  3. test the system by simulating arrivals of the question using different models

- Thesis writing: *10/2016 - 11/2016*

- Thesis defense: *12/2016*

7.4 Summary

In my thesis I proposed several pieces of work towards improving user satisfaction with question answering systems for factoid as well as non-factoid information needs. The proposed research spans multiple directions: from improving the data sources for factoid question answering, better answer representation by summarizing available pieces of information, to engaging the human input, either from a crowd of external workers or the user herself in a form of feedback. I believe, that the results of the proposed research directions will be useful for the developing field of intelligent personal assistants and chat bots, as question answering is arguably one of their most important applications.

Bibliography

[1] M. Ageev, Q. Guo, D. Lagun, and E. 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] E. Agichtein, D. Carmel, D. Harman, D. Pelleg, and Y. Pinter. Overview of the trec 2015 liveqa track. In *Proceedings of TREC 2015*, 2015.

[3] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL '00, pages 85–94, New York, NY, USA, 2000. ACM.

[4] E. Agichtein, S. Lawrence, and L. Gravano. Learning search engine specific query transformations for question answering. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10*, pages 169–178, 2001.

[5] E. Agichtein, S. Lawrence, and L. Gravano. Learning search engine specific query transformations for question answering. In *Proceedings of WWW 2001*, WWW '01, pages 169–178, New York, NY, USA, 2001. ACM.

[6] D. Ahn, V. Jijkoun, G. Mishne, K. Müller, M. de Rijke, and K. Schlobach. Using wikipedia at the trec qa track. In *The Thirteenth Text Retrieval Conference (TREC 2004)*, 2005.

[7] A. M. N. Allam and M. H. Haggag. The question answering systems: A survey. *International Journal of Research and Reviews in Information Sciences (IJRRIS)*, 2(3), 2012.

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

[9] O. Alonso, D. E. Rose, and B. Stewart. Crowdsourcing for relevance evaluation. *SIGIR Forum*, 42(2):9–15, Nov. 2008.

[10] A. Andrenucci and E. Sneiders. Automated question answering: Review of the main approaches. In *null*, pages 514–519. IEEE, 2005.

[11] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. Natural language interfaces to databases–an introduction. *Natural language engineering*, 1(01):29–81, 1995.

[12] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. *Dbpedia: A nucleus for a web of open data*. Springer, 2007.

[13] B. I. Aydin, Y. S. Yilmaz, Y. Li, Q. Li, J. Gao, and M. Demirbas. Crowdsourcing for multiple-choice question answering. In *AAAI*, pages 2946–2953, 2014.

[14] H. Bast and E. Haussmann. More accurate question answering on freebase. In *CIKM*, 2015.

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

[16] J. Berant, A. Chou, R. Frostig, and P. 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.

[17] J. Berant and P. Liang. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual*

Meeting of the Association for Computational Linguistics, ACL 2014, pages 1415–1425, 2014.

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

[19] D. Bernhard and I. 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-Volume 2*, pages 728–736. Association for Computational Linguistics, 2009.

[20] M. S. Bernstein, J. Brandt, R. C. Miller, and D. 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.

[21] M. S. Bernstein, J. Teevan, S. Dumais, D. Liebling, and E. 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.

[22] F. Bessho, T. Harada, and Y. 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 '12, pages 227–231, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[23] S. Bhatia, D. Majumdar, and P. 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. ACM.

[24] J. P. Bigham, C. Jayant, H. Ji, G. Little, A. Miller, R. C. Miller, R. Miller, A. Tatarowicz, B. White, S. 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.

[25] D. Bilal and J. Kirby. Differences and similarities in information seeking: Children and adults as web users. *Inf. Process. Manage.*, 38(5):649–670, Sept. 2002.

[26] M. W. Bilotti, P. Ogilvie, J. Callan, and E. 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.

[27] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA, 2008. ACM.

[28] A. Bordes, S. Chopra, and J. 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.

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

[30] A. Bordes, J. Weston, R. Collobert, and Y. Bengio. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*, number EPFL-CONF-192344, 2011.

[31] A. Bozzon, M. Brambilla, and S. Ceri. Answering search queries with crowdsearcher. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 1009–1018, New York, NY, USA, 2012. ACM.

[32] E. Brill, S. Dumais, and M. 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.

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

[34] C. J. Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11:23–581, 2010.

[35] D. Buscaldi and P. 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.

[36] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: Exploring the power of tables on the web. *Proc. VLDB Endow.*, 1(1):538–549, Aug. 2008.

[37] Q. Cai and A. Yates. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL (1)*, pages 423–433. Citeseer, 2013.

[38] Q. Cai and A. 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.

[39] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. 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.

[40] Z. Cao, F. Wei, L. Dong, S. Li, and M. Zhou. Ranking with recursive neural networks and its application to multi-document summarization. In *AAAI*, pages 2153–2159, 2015.

[41] D. Carmel, M. Shtalhaim, and A. Soffer. eresponder: Electronic question responder. In *International Conference on Cooperative Information Systems*, pages 150–161. Springer, 2000.

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

[43] W. Chan, X. Zhou, W. Wang, and T.-S. Chua. Community answer summarization for multi-sentence question with group l 1 regularization. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 582–591. Association for Computational Linguistics, 2012.

[44] A. X. Chang, V. I. Spitkovsky, E. Agirre, and C. D. Manning. Stanford-ubc entity linking at tac-kbp, again. In *Proceedings of Text Analysis Conference*, TAC'11, 2011.

[45] S. Chopra, M. Auli, A. M. Rush, and S. Harvard. Abstractive sentence summarization with attentive recurrent neural networks. 2016.

[46] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[47] G. Cong, L. Wang, C.-Y. Lin, Y.-I. Song, and Y. 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.

[48] M. Cornolti, P. Ferragina, M. Ciaramita, H. Schütze, and S. Rüd. The smaph system for query entity recognition 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.

[49] M. Cornolti, P. Ferragina, M. Ciaramita, H. Schütze, and S. Rüd. The smaph system for query entity recognition and disambiguation. In *Proceedings of the first international workshop on Entity recognition & disambiguation*, 2014.

[50] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 299–306. ACM, 2002.

[51] H. T. Dang, D. Kelly, and J. J. Lin. Overview of the trec 2007 question answering track. In *TREC*, volume 7, page 63. Citeseer, 2007.

[52] H. T. Dang, D. Kelly, and J. J. Lin. Overview of the trec 2007 question answering track. In *TREC*, 2007.

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

[54] S. Ding, G. Cong, C.-Y. Lin, and X. Zhu. Using conditional random fields to extract contexts and answers of questions from online forums. In *ACL*, volume 8, pages 710–718. Citeseer, 2008.

[55] L. Dong, F. Wei, M. Zhou, and K. 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.

[56] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. 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 '14, pages 601–610, New York, NY, USA, 2014. ACM.

[57] H. Duan, Y. Cao, C.-Y. Lin, and Y. Yu. Searching questions by identifying question topic and question focus. In *ACL*, pages 156–164, 2008.

[58] S. Elbassuoni, M. Ramanath, R. Schenkel, M. Sydow, and G. 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.

[59] G. Erkan and D. R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479, 2004.

[60] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld. Open information extraction from the web. *Commun. ACM*, 51(12):68–74, Dec. 2008.

[61] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[62] A. Fader, L. Zettlemoyer, and O. 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.

[63] A. Fader, L. S. Zettlemoyer, and O. Etzioni. Paraphrase-driven learning for open question answering. In *ACL*. Citeseer, 2013.

[64] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

[65] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, et al. This is watson. *IBM Journal of Research and Development*, 56, 2012.

[66] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 61–72. ACM, 2011.

[67] D. Fried, P. Jansen, G. Hahn-Powell, M. Surdeanu, and P. Clark. Higher-order lexical semantic models for non-factoid answer reranking. *Transactions of the Association for Computational Linguistics*, 3:197–210, 2015.

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

[69] E. Gabrilovich, M. Ringgaard, and A. Subramanya. Facc1: Freebase annotation of clueweb corpora, 2013.

[70] R. Gangadharaiah and B. Narayanaswamy. Natural language query refinement for problem resolution from crowdsourced semi-structured data. In *IJCNLP'2013*, pages 243–251, 2013.

[71] M. Gardner and T. 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.

[72] C. Grady and M. 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.

[73] B. F. Green Jr, A. K. Wolf, C. Chomsky, and K. 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.

[74] P. Gupta and V. Gupta. A survey of text question answering techniques. *International Journal of Computer Applications*, 53(4), 2012.

[75] R. Gupta, A. Halevy, X. Wang, S. E. Whang, and F. Wu. Biperpedia: An ontology for search applications. *Proc. VLDB Endow.*, 7(7):505–516, Mar. 2014.

[76] F. M. Harper, J. Weinberg, J. Logie, and J. A. Konstan. Question types in social q&a sites. *First Monday*, 15(7), 2010.

[77] C. G. Harris and P. 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.

[78] B. He and I. Ounis. Query performance prediction. *Information Systems*, 31(7):585–594, 2006.

[79] M. Heilman and N. 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.

[80] D. Hiemstra and S. E. Robertson. Relevance feedback for best match term weighting algorithms in information retrieval. 2001.

[81] W. Hildebrandt, B. Katz, and J. J. Lin. Answering definition questions using multiple knowledge sources. In *HLT-NAACL*, pages 49–56, 2004.

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

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

[84] E. Hovy, U. Hermjakob, and D. 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.

[85] E. H. Hovy, L. Gerber, U. Hermjakob, M. Junk, and C.-Y. Lin. Question answering in webclopedia. In *TREC*, volume 52, pages 53–56, 2000.

[86] E. H. Hovy, U. Hermjakob, and C.-Y. Lin. The use of external knowledge of factoid qa. In *TREC*, volume 2001, pages 644–52, 2001.

[87] T.-H. K. Huang, W. S. Lasecki, and J. P. Bigham. Guardian: A crowd-powered spoken dialog system for web apis. In *Third AAAI Conference on Human Computation and Crowdsourcing*, 2015.

[88] K. Ignatova, C. Toprak, D. Bernhard, and I. Gurevych. Annotating question types in social q&a sites. In *Tagungsband des GSCL Symposiums Sprachtechnologie und eHumanities*, pages 44–49. Citeseer, 2009.

[89] A. Ittycheriah, M. Franz, and S. Roukos. Ibm's statistical question answering system-trec-10. In *TREC*, 2001.

[90] M. Iyyer, J. L. Boyd-Graber, L. M. B. Claudino, R. Socher, and H. Daumé III. A neural network for factoid question answering over paragraphs. In *EMNLP*, pages 633–644, 2014.

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

[92] J. Jeon, W. B. Croft, and J. H. Lee. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 84–90, New York, NY, USA, 2005. ACM.

[93] V. Jijkoun and M. 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.

[94] V. Jijkoun, M. De Rijke, and J. 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.

[95] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proceedings of the 15th International Conference on World Wide Web*, pages 387–396, New York, NY, USA, 2006.

[96] M. Kågebäck, O. Mogren, N. Tahmasebi, and D. Dubhashi. Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)@ EACL*, pages 31–39. Citeseer, 2014.

[97] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[98] M. Keikha, J. H. Park, W. B. Croft, and M. Sanderson. Retrieving passages and finding answers. In *Proceedings of the 2014 Australasian Document Computing Symposium*, page 81. ACM, 2014.

[99] D. Kelly, K. Gyllstrom, and E. 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.

[100] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[101] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[102] J. Kiseleva, K. Williams, J. Jiang, A. Hassan Awadallah, A. C. Crook, I. Zitouni, and T. 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.

[103] C. Kohlschütter, P. Fankhauser, and W. Nejdl. Boilerplate detection using shallow text features. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 441–450, New York, NY, USA, 2010. ACM.

[104] O. Kolomiyets and M.-F. Moens. A survey on question answering technology from an information retrieval perspective. *Information Sciences*, 181(24):5412–5434, Dec. 2011.

[105] A. Kotov and C. Zhai. Towards natural question guided search. In *WWW'2010*, pages 541–550, 2010.

[106] S. Kriewel and N. 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.

[107] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, pages 957–966, 2015.

[108] C. Kwok, O. Etzioni, and D. S. Weld. Scaling question answering to the web. *ACM Transactions*

on Information Systems (TOIS), 19(3):242–262, 2001.

[109] N. Lao, A. Subramanya, F. Pereira, and W. 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.

[110] W. S. Lasecki, R. Wesley, J. Nichols, A. Kulkarni, J. F. Allen, and J. P. Bigham. Chorus: A crowd-powered conversational assistant. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 151–162, New York, NY, USA, 2013. ACM.

[111] V. Lavrenko and W. B. Croft. Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127. ACM, 2001.

[112] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014.

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

[114] B. Li, X. Si, M. R. Lyu, I. King, and E. Y. Chang. Question identification on twitter. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2477–2480. ACM, 2011.

[115] X. Li and D. Roth. Learning question classifiers. In *19th International Conference on Computational Linguistics, COLING 2002*, 2002.

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

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

[118] J. Lin and B. Katz. Question answering from the web using knowledge annotation and knowledge mining techniques. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 116–123. ACM, 2003.

[119] Y. Liu, J. Bian, and E. 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.

[120] Y. Liu, S. Li, Y. Cao, C.-Y. Lin, D. Han, and Y. Yu. Understanding and summarizing answers in community-based question answering services. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 497–504, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[121] Y. Liu, S. Li, Y. Cao, C.-Y. Lin, D. Han, and Y. Yu. Understanding and summarizing answers in community-based question answering services. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 497–504. Association for Computational Linguistics, 2008.

[122] Y. Lv and C. Zhai. Positional relevance model for pseudo-relevance feedback. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 579–586. ACM, 2010.

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

[124] C. Malon and B. Bai. Answer extraction by recursive parse tree descent. *ACL 2013*, page 110, 2013.

[125] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. 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.

[126] T. Mikolov and J. Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 2013.

[127] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*, 2016.

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

[129] M. Mintz, S. Bills, R. Snow, and D. 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.

[130] B. Mitra, G. Simon, J. Gao, N. Craswell, and L. Deng. A proposal for evaluating answer distillation from web data. 2016.

[131] N. Moraveji, D. Russell, J. Bien, and D. 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.

[132] F. Nikolaev, A. Kotov, and N. Zhiltsov. Parameterized fielded term dependence models for ad-hoc entity retrieval from knowledge graph. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 435–444. ACM, 2016.

[133] A. Omari, D. Carmel, O. Rokhlenko, and I. Szpektor. Novelty based ranking of human answers for community questions. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 215–224. ACM, 2016.

[134] C.-S. Ong, M.-Y. Day, and W.-L. Hsu. The measurement of user satisfaction with question answering systems. *Information & Management*, 46(7):397–403, 2009.

[135] V. Pande, T. Mukherjee, and V. Varma. Summarizing answers for community question answer services. In *Language Processing and Knowledge in the Web*, pages 151–161. Springer, 2013.

[136] S. Park, S. Kwon, B. Kim, and G. G. Lee. Isoft at qald-5: Hybrid question answering system over linked data and text data. CLEF, 2015.

[137] M. Pasca and S. 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.

[138] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43, 2014.

[139] J. Pound, P. Mika, and H. 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.

[140] J. Prager, J. Chu-Carroll, E. W. Brown, and K. Czuba. Question answering by predictive annotation. In *Advances in Open Domain Question Answering*, pages 307–347. Springer, 2006.

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

[142] V. Punyakanok, D. Roth, and W.-t. Yih. Mapping dependencies trees: An application to question answering. In *Proceedings of AI&Math 2004*, pages 1–10, 2004.

[143] S. Reddy, M. Lapata, and M. Steedman. Large-scale semantic parsing without question-answer pairs. *TACL*, 2:377–392, 2014.

[144] Z. Ren, H. Song, P. Li, S. Liang, J. Ma, and M. de Rijke. Using sparse coding for answer summarization in non-factoid community question-answering. 2016.

[145] S. Riedel, L. Yao, A. McCallum, and B. M. Marlin. Relation extraction with matrix factorization and universal schemas. *NAACL HLT 2013*, pages 74–84, 2013.

[146] J. J. Rocchio. Relevance feedback in information retrieval. 1971.

[147] A. M. Rush, S. Chopra, and J. Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

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

[149] H. Sajjad, P. Pantel, and M. Gamon. Underspecified query refinement via natural language question generation. In *COLING'2012*, pages 2341–2356, 2012.

[150] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Readings in information retrieval*, 24(5):355–363, 1997.

[151] C. d. Santos, M. Tan, B. Xiang, and B. Zhou. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*, 2016.

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

[153] D. Savenkov, W.-L. Lu, J. Dalton, and E. Agichtein. Relation extraction from community generated question-answer pairs. In *NAACL-HLT 2015 Student Research Workshop (SRW)*, page 96, 2015.

[154] C. Shah and J. 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.

[155] R. Sharp, P. Jansen, M. Surdeanu, and P. 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.

[156] D. Shen, G.-J. M. Kruijff, and D. Klakow. Exploring syntactic relation patterns for question answering. In *Natural Language Processing–IJCNLP 2005*, pages 507–518. Springer, 2005.

[157] E. H. Shortliffe and B. G. Buchanan. A model of inexact reasoning in medicine. *Mathematical biosciences*, 23(3):351–379, 1975.

[158] A. Shtok, G. Dror, Y. Maarek, and I. Szpektor. Learning from the past: Answering new questions with past answers. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 759–768, New York, NY, USA, 2012. ACM.

[159] R. F. Simmons. Answering english questions by computer: A survey. *Communications of ACM*, 8(1):53–70, Jan. 1965.

[160] R. F. Simmons. Natural language question-answering systems: 1969. *Commun. ACM*, 13(1):15–30, Jan. 1970.

[161] R. Snow, D. Jurafsky, and A. Y. Ng. Learning syntactic patterns for automatic hypernym discovery. *Advances in Neural Information Processing Systems 17*, 2004.

[162] P. Sondhi and C. 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.

[163] R. Soricut and E. Brill. Automatic question answering using the web: Beyond the factoid. *Information Retrieval*, 9(2):191–206, 2006.

[164] M. M. Soubbotin and S. M. Soubbotin. Patterns of potential answer expressions as clues to the right answers. In *TREC*, 2001.

[165] V. I. Spitkovsky and A. X. Chang. A cross-lingual dictionary for english wikipedia concepts. In

LREC, pages 3168–3175, 2012.

[166] V. I. Spitkovsky and A. X. Chang. A cross-lingual dictionary for english wikipedia concepts. In N. C. C. Chair), K. Choukri, T. Declerck, M. U. Doan, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

[167] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[168] H. Sun, H. Ma, W.-t. Yih, C.-T. Tsai, J. Liu, and M.-W. Chang. Open domain question answering via semantic enrichment. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1045–1055, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee.

[169] M. Surdeanu, M. Ciaramita, and H. Zaragoza. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics*, 37(2):351–383, 2011.

[170] M. Surdeanu, J. Tibshirani, R. Nallapati, and C. 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 '12, pages 455–465, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[171] M. Tan, B. Xiang, and B. Zhou. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*, 2015.

[172] Y. Tang, F. Bu, Z. Zheng, and X. Zhu. Towards interactive qa: suggesting refinement for questions. In *SIGIR'2011 Workshop on "entertain me": Supporting Complex Search Tasks*, pages 13–14, 2011.

[173] W. tau Yih, M.-W. Chang, X. He, and J. 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*. ACL Association for Computational Linguistics, July 2015.

[174] M. Tomasoni and M. 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. Association for Computational Linguistics, 2010.

[175] K. Toutanova, D. Chen, P. Pantel, P. Choudhury, and M. Gamon. Representing text for joint embedding of text and knowledge bases. *ACL Association for Computational Linguistics*, 2015.

[176] C. Tsai, W.-t. Yih, and C. Burges. Web-based question answering: Revisiting askmsr. Technical report, Technical Report MSR-TR-2015-20, Microsoft Research, 2015.

[177] C. Unger, A. Freitas, and P. 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.

[178] R. Usbeck and A.-C. N. Ngomo. Hawk@ qald5–trying to answer hybrid questions with various simple ranking techniques. CLEF, 2015.

[179] P. Verga and A. McCallum. Row-less universal schema. *arXiv preprint arXiv:1604.06361*, 2016.

[180] E. M. Voorhees. The trec question answering track. *Natural Language Engineering*, 7(04):361–378, 2001.

[181] D. Vrandečić and M. Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[182] C. Wang, Y. Song, A. El-Kishky, D. Roth, M. Zhang, and J. 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.

[183] C. Wang, Y. Song, H. Li, M. Zhang, and J. Han. Text classification with heterogeneous information

network kernels. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[184] D. Wang and E. Nyberg. Cmu oaqa at trec 2015 liveqa: Discovering the right answer with clues. In *Proceedings of TREC*, 2015.

[185] D. Wang and E. 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, ACL 2015*, pages 707–712, 2015.

[186] D. Wang and E. Nyberg. A long short-term memory model for answer sentence selection in question answering. *ACL*, 2015.

[187] M. Wang. A survey of answer extraction techniques in factoid question answering. *Computational Linguistics*, 1(1), 2006.

[188] M. Wang and C. 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. Association for Computational Linguistics, 2010.

[189] M. Wang, N. A. Smith, and T. Mitamura. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*, volume 7, pages 22–32, 2007.

[190] X. Wang, H. Fang, and C. Zhai. A study of methods for negative relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 219–226. ACM, 2008.

[191] Z. Wang and A. Ittycheriah. Faq-based question answering via word alignment. *arXiv preprint arXiv:1507.02628*, 2015.

[192] Z. Wang, S. Yan, H. Wang, and X. 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.

[193] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph and text jointly embedding. In *EMNLP*, pages 1591–1601. Citeseer, 2014.

[194] S. E. Whang, P. Lofgren, and H. Garcia-Molina. Question selection for crowd entity resolution. *Proc. VLDB Endow.*, 6(6):349–360, Apr. 2013.

[195] R. Wilensky, D. N. Chin, M. Luria, J. Martin, J. Mayfield, and D. Wu. The berkeley unix consultant project. *Computational Linguistics*, 14(4):35–84, 1988.

[196] D. C. Wimalasuriya and D. Dou. Ontology-based information extraction: An introduction and a survey of current approaches. *Journal of Information Science*, 2010.

[197] W. A. Woods and R. Kaplan. Lunar rocks in natural english: Explorations in natural language question answering. *Linguistic structures processing*, 5:521–569, 1977.

[198] G. Wu and M. Lan. Leverage web-based answer retrieval and hierarchical answer selection to improve the performance of live question answering. In *Proceedings of TREC*, 2015.

[199] I. Xie and C. 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.

[200] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2015.

[201] K. Xu, S. Reddy, Y. Feng, S. Huang, and D. Zhao. Question answering on freebase via relation extraction and textual evidence. 2016.

[202] K. Xu, S. Zhang, Y. Feng, and D. Zhao. Answering natural language questions via phrasal semantic parsing. In *Natural Language Processing and Chinese Computing*, pages 333–344. Springer, 2014.

[203] M. Yahya. Question answering and query processing for extended knowledge graphs. 2016.

[204] M. Yahya, D. Barbosa, K. Berberich, Q. Wang, and G. 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.

[205] M. Yahya, K. Berberich, S. Elbassuoni, and G. 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.

[206] J.-M. Yang, R. Cai, Y. Wang, J. Zhu, L. Zhang, and W.-Y. 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.

[207] L. Y. Yang, Q. Ai, D. Spina, R.-C. Chen, L. Pang, W. B. Croft, J. Guo, and F. Scholer. Beyond factoid qa: Effective methods for non-factoid answer sentence retrieval. In *Proceedings of ECIR'16*, 2016.

[208] Y. Yang, W.-t. Yih, and C. 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.

[209] X. Yao. Lean question answering over freebase from scratch. In *Proceedings of NAACL Demo*, 2015.

[210] X. Yao, J. Berant, and B. Van Durme. Freebase qa: Information extraction or semantic parsing? *ACL 2014*, page 82, 2014.

[211] X. Yao and B. V. 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.

[212] X. Yao, B. Van Durme, C. Callison-Burch, and P. Clark. Answer extraction as sequence tagging with tree edit distance. In *HLT-NAACL*, pages 858–867. Citeseer, 2013.

[213] X. Yao, B. Van Durme, and P. Clark. Automatic coupling of answer extraction and information retrieval. In *ACL (2)*, pages 159–165, 2013.

[214] S. W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of ACL*, 2016.

[215] W.-t. Yih, X. He, and C. Meek. Semantic parsing for single-relation question answering. In *ACL (2)*, pages 643–648. Citeseer, 2014.

[216] P. Yin, N. Duan, B. Kao, J. Bao, and M. 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.

[217] L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*, 2014.

[218] J. M. Zelle and R. J. Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055, 1996.

[219] N. Zhiltsov, A. Kotov, and F. Nikolaev. Fielded sequential dependence model for ad-hoc entity retrieval in the web of data. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 253–262. ACM, 2015.

[220] G. Zhou, T. He, J. Zhao, and P. Hu. Learning continuous word embedding with metadata for question retrieval in community question answering. In *Proceedings of ACL*, pages 250–259, 2015.

[221] Y. Zhou and W. B. Croft. Query performance prediction in web search environments. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 543–550. ACM, 2007.