Question Answering Using Structured and Unstructured Data

Doctoral thesis proposal

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

April, 2016

Abstract

Over more than half a century of research, the area of automatic Question Answering (QA) has progressed from small single domain systems to IBM Watson, who defeated best human competitors in the Jeopardy! TV show [53]. However, many of our questions are still left unanswered, and we still have a lot to do to move beyond 10 blue links in search results [48] as for most of the questions users still have to dig into the retrieved documents or post questions to the community question answering (CQA) websites. Questions come in different flavors, some are asking about a certain fact and can be answered with a short phrase, such as entity name, date or number. Such questions are typically referred to as *factoid*, as opposed to rest of the questions, which are often called *non-factoid*. In my thesis I focus on three topics in QA: 1). combination of structured and unstructured data to improve factoid question answering; 2). improving question summarization, candidate scoring and answer generation using recent advances in neural network research and better utilization of source web document structure; 3). interactions between a question answering system and real people.

Text document collections and knowledge bases (KB) are very effective in answering certain types of factoid questions, but they are also complimentary to each other. I propose to combine these data sources via semantic annotation of KB entity mentions, which effectively extends the knowledge base with additional unstructured information, often missing or complimentary to the KB data.

Non-factoid question answering is somewhat harder as it deals with a more diverse set of question and answer types. In my thesis I propose to improve performance of different stages of QA system pipeline by better utilization of the structure of a web page where a candidate answer is extracted from, and using deep learning techniques, inspired by recent successes in machine translation [12], text summarization [101], automatic caption generation [74] and answer sentence scoring [128].

The focus of the last part of the thesis is on the interaction between human and a search or question answering system. Unfortunately, there always be cases, when a machine cannot provide a good answer to the question. In such cases, a QA system may come back to the user with some suggestions on how a user can solve his search problem, or with a clarification question aiming at resolving certain ambiguities. Alternatively, a machine can consult with the crowd in order to get the answer or help it decide on certain alternatives.

In summary, the goal of the proposed research is to improve the performance of question answering over a variety of different questions a user might have, and to study some reply strategies in case a system fails to deliver a good response. I believe, that results of the proposed work will be useful for the future research in improving automatic question answering.

Contents

1 Introduction

1.1 Motivation

It has long been a dream to communicate with a computer as with another human being using natural language speech and text. Nowadays, we are coming closer to this dream as natural language interfaces become more and more popular. Our phones are already reasonably good in 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 startups developing this kind of technologies has emerged in Silicon Valley and around the world[1].

Questions are a natural way of communicating ones information needs. Users of modern search engines are already benefit from direct answers on search results page in response to some of their queries [2]. However, there is still a big number of questions, for which users have to dig into the "10 blue links" and mine answers from a pile of information contained in retrieved documents. Therefore, in my thesis I focus on helping users get answers to their questions by improving question answering methods and studying interactions between a human and a QA system.

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*"). However, many other questions do not fall into this group, *i.e.* people often ask for recommendations, opinions, instructions, reasons, *etc.* The literature usually refer to all these questions by "non-factoid questions" umbrella term.

Nowadays, most of the question answering systems retrieve answers from certain data sources. 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 has certain advantages and limitations (Table 1.1). Existing approaches often use either a single type of data sources, or combine them to a limited extend. Motivated by the fact that advantages and disadvantages of structured and unstructured data sources complement each other, In my thesis I propose to study methods of combining different data sources for joint reasoning for factoid and non-factoid questions.

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 is quite problematic. On the other hand, modern large scale knowledge bases, such as Freebase [25], dbPedia [10], YAGO [89], WikiData [126], 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

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

[2] https://www.stonetemple.com/rich-answers-in-search/

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

	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

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.

The main challenge in non-factoid question answering lies in the diversity of question and answer types. One of the most effective strategies is to reuse answers to previously asked questions, which could be found, for example, in CQA archives [110]. Unfortunately, it's not always possible to find a similar question, that has already been answered. Many information needs are unique or contain unique details, which makes it impossible to reuse old answers. 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 [114]. 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, one of the questions I'm going to focus in my thesis is how to effectively use the structure of web page to predict whether an extracted passage of text answer the given question.

However, ranking isn't the only important part of the question answering pipeline. A system can only rank and return a good answer if it was able to retrieve relevant information from a collection. Non-factoid questions, especially those that people post on CQA websites are often long, which makes it problematic to use directly as search queries. Previous research has studied certain question transformation strategies [3, 30, 86], however the focus was on shorter factoid questions. In my thesis I would like to focus on the problem of query generation for non-factoid questions using some recent advances in deep learning. Another promising direction of research, which I'm going to explore in my thesis, is answer generation, *i.e.* by summarizing the information a system could retrieve. Different answer candidates might by complimentary to each other, answer different parts of the question or have different opinions on the subject.

No matter how good a QA system we have, there will be cases, when it will be unable to answer a question user asked. It may be caused by various reasons, *e.g.* the question might be hard to answer with the existing data sources or it may be ambiguous. In such cases, it make sense to either appeal to another external resources, *e.g.* such as wisdom of a crowd, or interact with the asked and give him some suggestions on how to tackle her search problem or ask for some additional information.

1.2 Research Questions

Research questions I proposed addressed 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. What kind of information about a web page can help scoring a passage extracted from it as a possible answer to the given question?

3. RQ3. How to build question and answer summarization models to improve candidate retrieval and answer generation for non-factoid question answering?

4. RQ4. How we can improve user experience with question answering systems for complex informational tasks?

1.3 Research Plan

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

The goal is to study a problem of using multiple structured KB and unstructured data together to improve factoid question answering. Two major issues with KBQA is knowledge base incompleteness and complexity of translating natural language question into a structured query. Text documents on the other hand are easier to match against the question, contain more information than a typical knowledge base, but aggregating information across multiple statements and documents is complicated.

One way to improve the situation with knowledge base incompleteness is to extract missing information from other data sources, *e.g.* [32, 33, 46, 49, 61, 79]. I propose to explore one additional data source, that wasn't used for relation extraction before, namely question-answer pairs. Section 3.1 will describe our experiments and results in utilizing this data to improve knowledge base coverage. Unfortunately, relation extraction isn't perfect either and there are both precision and recall losses. Therefore, 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. Section 3.2 describes the approach in more detail.

1.3.2 Using Web Page Information to Improve Passage Ranking in Non-factoid Question Answering (Chapter 4.1.2)

To answer RQ2 I'm planning to study what kind of information from web pages can be useful to predict whether a passage of text answer the given questions. First, I'll derive a dataset of questions from TREC LiveQA'15 with passages, labeled by TREC assessors, and extract the corresponding web pages. This allows us to set the problem as passage ranking problem using a set of passage and web page context features. I will design a set of features representing a passage and some key elements from the web page and train an answer ranking model. The feature ablation experiments will reveal the relative importance of different model components.

1.3.3 Question and Answer Summarization for Non-factoid Question Answering (Chapter 4.1.1)

The goal is to develop models for question summarization, which should improve candidate answer retrieval performance, and answer summarization to generate the final response of the system. The plan is to explore recent advances in deep learning for text summarization [101] and generation [74] and apply these techniques for the above mentioned problems. The effectiveness of these models will be evaluated using the data from TREC LiveQA'15 and tested inside this year challenge model.

1.3.4 User Interaction with a QA system (Chapter 5)

The goal is to explore some techniques of involving human in the question answering problem. I'm going to focus on three scenarios:

- showing users hints on how to approach her complex informational search task.
- communicating with the user by asking certain clarification questions for ambiguous questions
- involving crowdsourcing to help a QA system to answer the given question

1.3.5 Research Timeline

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

- Completing the work proposed in Sections 1.3.2 and 1.3.3 (4/2016 - 5/2016): develop individual components for question analysis, answer candidate retrieval and answer generation and integrate them into a system to participate in TREC LiveQA'16.
- Completing the work proposed in Section 1.3.1 (6/2016 - 7/2016): Develop a model to use annotation of entity mentions for factoid question answering and compare it against existing techniques. I'm also planning to develop a new QA dataset, which will include more diverse and realistic set of questions than existing KBQA datasets and larger than available TREC QA datasets.

- Completing the work proposed in Section 1.3.4 (4/2016 - 5/2016): Integrate a crowdsourcing module into my TREC LiveQA'16 system as one of the options, develop a model to predict ambiguous questions on a CQA website and propose certain clarification questions.
- Thesis writing (07/2016 - 08/2016)
- Thesis defense (08/2016)

1.4 Contributions and Implications

The key contributions of the proposed research are: 1. New hybrid KB-text question answering algorithm, that is based on graph search, which includes both KB links as well as text search edges to follow. 2. New labelled dataset for question answering (???) 3. New features for ranking answer candidates ???

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 [111, 112]. 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 [9, 109, 135, 134]. 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 [125]. A comprehensive survey of the approaches from TREC QA 2007 can be found in [42]. An interested reader can refer to a number of surveys to track the progress made in automatic question answering over the years [63, 8, 129, 77, 98, 5, 60].

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.

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) [111]. 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 [75], 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 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

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

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

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

ways before querying a collection [3, 30]. 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 [24, 149].

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 [131], is based on TREC QA questions and sentences retrieved by participating systems[4]. The early approaches for the task used simple keyword match strategies [68, 115]. 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 [99, 108, 62, 148, 130]. 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 [69], convolutional neural networks [152, 103], recurrent neural networks [121, 128]. Another dataset, called WikiQA [144], 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 his 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 [148], or as a node labeling problem in an answer sentence parse tree [90].

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 [66, 65] and predicting and matching expected and candidate answer types from the available data [83, 84, 97]. Many approaches exploited external data for this task, I will describe some of this 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 [29]. 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 [123] 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`[80], `Aranea` [86], and a detailed analysis of what affects the performance of the redundancy-based question answering systems can be found in [85].

[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 statistics [59], lunar geology [135], geography [153]. However, one of the main challenges in KBQA is mapping between natural language phrases to the database concepts, which raises a problem of domain adaption of question answering systems.

Recent development of large scale knowledge bases (*e.g.* dbPedia [10], Freebase [25], YAGO [117], 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 [35] and WebQuestions [15]. A survey of some of the proposed approaches can be found in [124].

A series of QALD evaluation campaigns has started in 2011, and since then a number of different subtasks have been offered, *i.e.* since QALD-3 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 correct parse logical form, which is expensive to obtain. Such an approach is 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 [36]. However, most of the parses of the question produce different results, which means that it is possible to use question-answer pairs directly [15]. PARALEX system ([52]) construct a lexicon from a collection of question paraphrases from WikiAnswers[7]. A somewhat backwards approach was proposed in ParaSempre model of [16], 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 [93] labeling of a large text corpus, such as ClueWeb [146]. Such labelled collections can also be used to train a KBQA system, as demonstrated by [100]. Such an 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 surface forms for their predicates and entities, which makes it possible to convert KB RDF triples into questions and use them for training [26]. 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 a joint embedding of text and knowledge base concepts. 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 [150]. 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 ([26]). Memory Networks, developed by the

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

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

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

Facebook AI Lab, can also be used to return triples stored in network memory in a response to the user question [27]. This approach uses embeddings of predicates and can answer relatively simple questions, that do not contain any constraints and aggregations. To extend deep learning framework to more complex questions, [45] use multi-column convolutional neural network to capture the embedding of the entity path, context and type.

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 [15] 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 [17], which maintains current best parses across all levels. An alternative information extraction strategy was proposed by [147], which can be very effective for relatively simple questions. A comparison of this approaches can be found in [146]. 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.

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 [15, 16, 140]. However, NER can easily miss the right span and the whole system would fail to produce the answer. Recently, most of the state-of-the-art system on WebQuestions dataset used a strategy to consider a reasonable subset of token n-grams, each of which can map to zero or more KB entities, which are disambiguated on the answer ranking stage [145, 13, 122]. Ranking of candidates can be done using a simple linear classification model [145] or a more complex gradient boosted trees ranking model [13, 122].

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 needs 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 [122]. 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 [151]. In [133] authors propose to start from single KB facts and build more complex logical formulas by combining existing ones, while scoring candidates using paraphrasing model. Such a 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. WordNet lexical database [92] was among the first resources, that were adapted by QA community [67, 96], and it was used for such tasks as query expansion and definition extraction. Wikipedia[8], which can be characterized as an unstructured and semi-structured (infoboxes) knowledge base, quickly became a valuable resource for answer extraction and verification [4, 31]. Developers of the Aranea QA system noticed that structured knowledge bases are very effective in answering a significant portion of relatively simple questions [86]. 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[93, 72, 61], web tables [32], *etc.* 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 ([49]) can be used to extract a surface form-based knowledge base, which can be very effective for question answering. Open question answering approach of [51] 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.

SPOX tuples, proposed in [141], encode subject-predicate-object triples along with certain keywords, that could be extracted from the same place as RDF triple. These keywords encode the context of the triple and can be used to match against keywords in the question. The method attempts to parse the question and uses certain relaxations (removing SPARQL triple statements) along with adding question keyphrases as additional triple arguments. As an extreme case of relaxation authors build a query that return all entities of certain type and use all other question terms to filter and rank the returned list.

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. In [139], authors propose 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 [118] 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.

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 [53] or in the full special issue of the IBM Journal of Research and Development [54]. YodaQA [14] is an open source implementation of the ideas behind the IBM Watson system.

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

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

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 in [114] 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, [88] studies 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 [18, 110, 47, 70]. WebAP is a dataset for non-factoid answer sentence retrieval, which was developed in [143]. 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.

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. One of the first extensive studies of different features for non-factoid answer ranking can be found in [119], 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 [132]. 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 [57] 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 [107].

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 [154]. 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 [113]. 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 [39, 71, 142, 44, 82].

Most of the approaches from TREC LiveQA 2015 combined similar question retrieval and web search techniques [136, 104, 127]. Answers to similar questions are very effective for answering new questions [104]. 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 [127] 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.

2.3 User interactions

IN PROGRESS...

There has been considerable amount of work on search assistance and improving user experience with feedback, suggestions and hints. Results of the study in [137] demonstrate that in 59.5% of the cases users need help to refine their searches or to construct search statements. Individual term ([102]) or query suggestion ([22, 37, 73]) are among the most popular techniques for helping users to augment their queries. The study in [76] 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 [22].

When query or term suggestions are not efficient, it is still possible to help users by providing potentially useful search hints. An adaptive tool providing tactical suggestions was presented in [78] 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 [94]. The study demonstrated the potential effectiveness and teaching effect of hints. The major difference of our work from [94] is the type of search hints used. Rather than suggesting to users the available search functionality, this work focuses on *strategic* search hints, designed to solve difficult informational questions.

Using the wisdom of a crowd to help users satisfy their information needs has been studied before in the literature. [20] explored the use of crowdsourcing for offline preparation of answers to tail search queries. In this work 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 our work, however, the focus is on online question answering, which requires fast responses to the user, who is unlikely to wait more than a minute. Another related work is targeting a different domain, namely SQL queries. The CrowdDB system [55] 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 [11] 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.

Using crowdsourcing for relevance judgments has been studied extensively in the information retrieval community, e.g., [7, 6, 58] to name a few. The focus in these works is on document relevance, and the quality of crowdsourced judgments. Whereas in our paper we are investigating the ability of a crowd to quickly assess the quality of the answers in a nearly real-time setting.

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. [19] 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. We are planning to build on these ideas and integrate a crowd into a real-time question answering system. The work of [81] showed how multiple workers can sit behind a conversational agent named Chorus, where human input is used to propose and vote on responses. Another use of a crowd for maintaining a dialog is presented in [21], who let the crowd handle difficult cases, when a system was not able to automatically retrieve a good response from the database of twitter data. In this paper, we focus on a single part of the human-computer dialog, i.e. question answering, which requires a system to provide some useful information in a response to the user.

An interesting approach for knowledge base construction through dialog with the user has been proposed by [64].

A very nice crowdsourcing method to obtain answers to tail information needs was proposed by [20]. Question query-url pairs are first mined from query logs, and then the wisdom of a crowd is used to extract and save answers to these questions.

Certain questions cannot be answered by machines only due to reasons such as lack of the appropriate data. Crowdsourcing was explored as one of the options to bridge this knowledge gap and assist the machine in matching, ranking and result aggregation [56]. Wisdom of a crowd can also be exploited to answer more difficult quiz questions [11].

[28] retrieves Wikipedia statements that support user answers for opinion questions.

A number of works have focused on studying and estimating different factors of user satisfaction with question answering systems [95, 87]

Interactive TREC ciQA...

[43] analyzes how clarification questions can be detected and used to improve QA performance. As far as I understand, the clarification question comes from the same user, not the system.

2.4 Summary of Related Work

Most previous work in ...

3 Structured and Unstructured Data for Factoid Question Answering

There are multiple ways to marry unstructured and structured data for joint question answering: convert structured data to unstructured format or vice versa, convert all data to certain intermediate representation or to leave them as is and link the data sources. In my thesis I focus on two approaches: relation extraction for knowledge base completion, and semantic annotation of text for hybrid question answering.

3.1 Relation Extraction for Knowledge Base Completion

The information on the web is stored in multiple different forms, such as natural language statements, tables and infoboxes, images *etc.* In this work I focus on yet another source of information: question-answer pairs. Community question answering archives contain hundreds of millions of question and corresponding answers. Information expressed in these pairs might be hard to extract or not exist at all in other formats.

3.1.1 Relation extraction from 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 [110]. 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 [105]). 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 this thesis we 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 entities mentioned in answer sentences. The conducted experiments confirm that many of such facts

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

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

[3] http://quora.com

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

cannot be extracted by existing sentence-based techniques and thus it is beneficial to combine their outputs with the output of our model.

Let us define the problem more formally. We target the problem of 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.

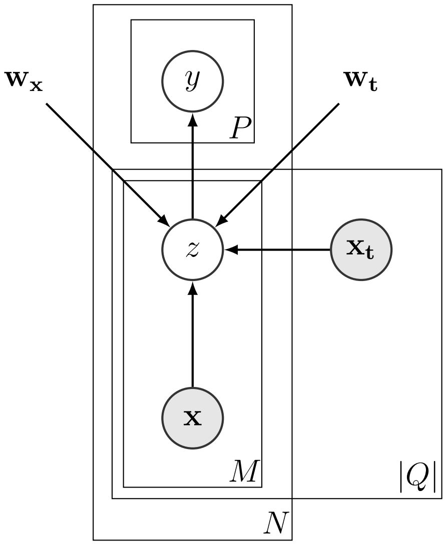


Figure 3.1: QnA-based relation extraction model plate diagram. N - number of different entity pairs, M - number of mentions of an entity pair, $|Q|$ - number of questions where an entity pair is mentioned, $\mathbf{x}$ and $\mathbf{x_t}$ - mention-based and question-based features, $\mathbf{w}$ and $\mathbf{w_t}$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

Our 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 [93], 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 [120], which was shown to be superior to the original model of [93], is easier to train than some other state of the art distant supervision models and produces comparable results.

Sentence-based model with question features

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

Sentence-based model	
Dependency path between entities	[PERSON]→nsubjpass(born)tmod←[DATE]
Surface pattern	[PERSON] be/VBD born/VBN [DATE]

Question features for sentence-based model	
Question template	when [PERSON] born
Dependecy path from a verb to the question word	(when)→advmod(born)
Question word + dependency tree root	when+born

QnA-based model	
Question template + answer entity type	Q: when [PERSON] born A:[DATE]
Dependency path from question word to entity	Q:(when)→advmod(born)nsubj←[PERSON]
and answer entity to the answer tree root	A: (born)tmod←[DATE]
Question word, dependency root and answer pattern	Q: when+born A:born [DATE]

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 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 [51]. 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).

Table 3.2: Yahoo! Answers and WikiAnswers datasets statistics

	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%

For each QnA pair we applied tokenization, sentence detection, named entity tagger, parsing and coreference resolution from Stanford CoreNLP [91]. Our cascade entity linking approach is similar to [38] 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 [116] 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.

[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

The percentage is higher for WikiAnswers, which has shorter answers on average. Unfortunately, for many QnA pairs we were unable to find relations between the mentioned entities (for many of them no or few entities were resolved to Freebase). Among those QnA pairs, where some relation was annotated, we looked at the location of related entities. In Yahoo! Answers dataset 38.6% (12.0% for WikiAnswers) of related entities are mentioned in answer sentences and can potentially be extracted by sentence-based model, and 28.1% (46.4% for WikiAnswers) between entities mentioned in question and answer sentences, which are not available to the baseline model and our goal is to extract some of them.

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

Table 3.3: Extraction results for QnA- and sentence-based models on both datasets

	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%	-

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).

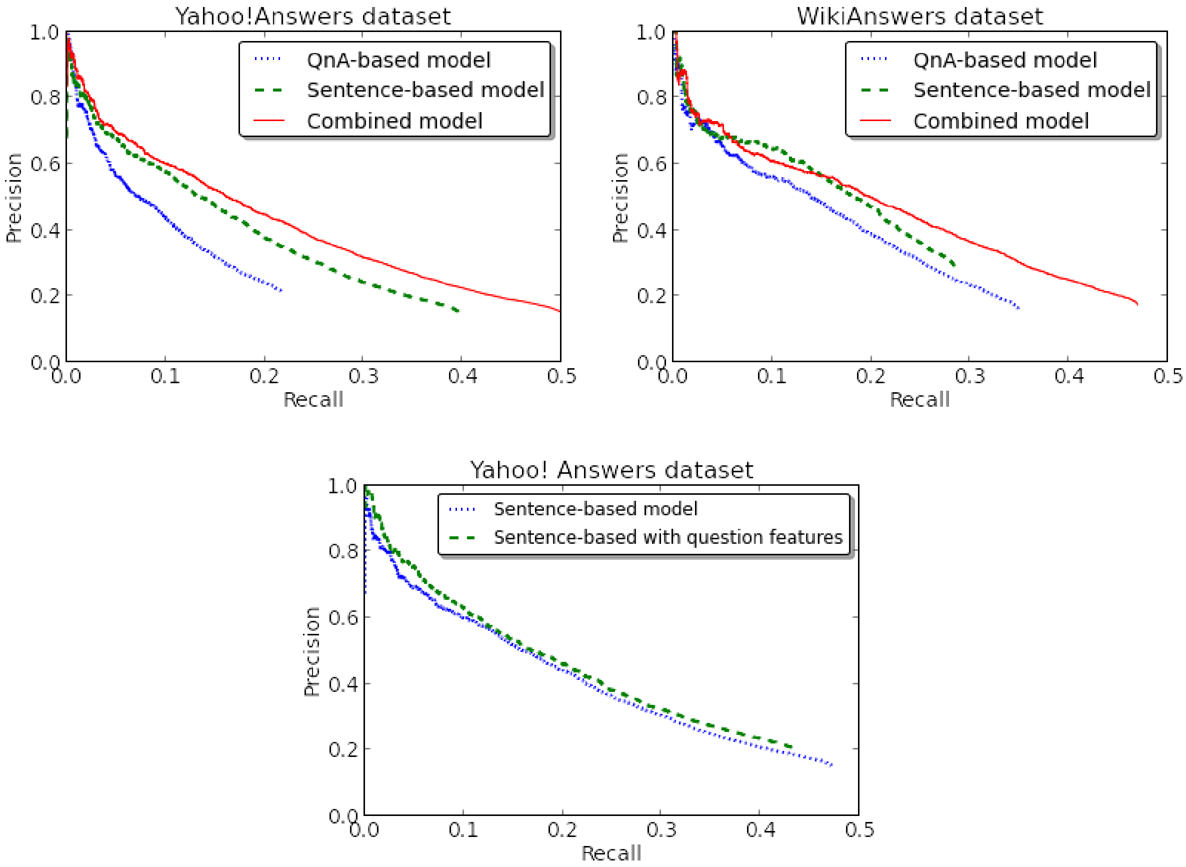


Figure 3.2: Precision-Recall curves for QnA-based vs sentence-based models and sentence-based model with and without question features

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.* [106], 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.

Conclusion

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.

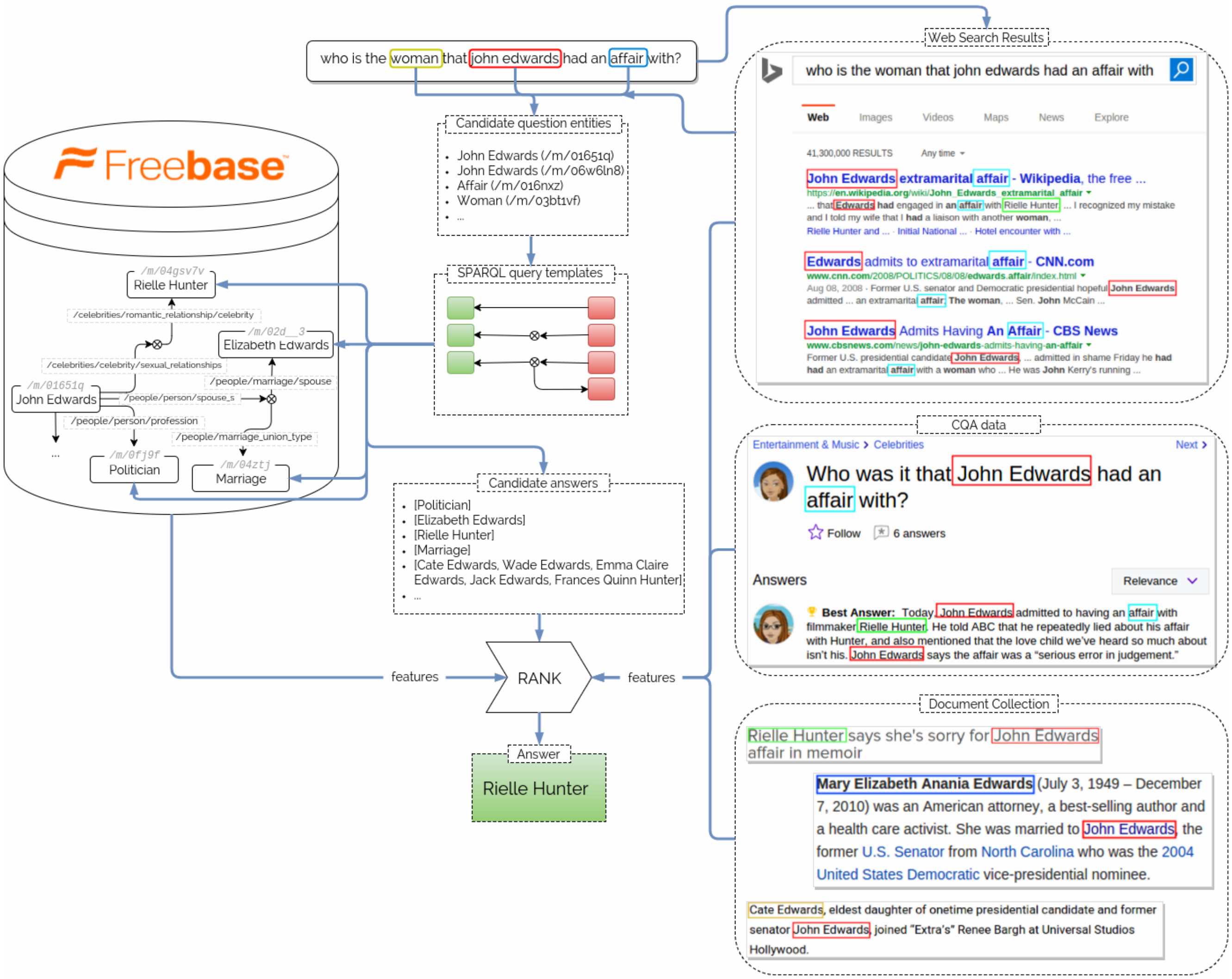
3.2 Semantic Text Annotations for Hybrid Question Answering

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.

Here I propose to utilize the synergy of structured and unstructured data, and exploit the advantages of each of them to overcome the limitations of the other. More particularly, I propose to annotate and index mentions of knowledge base entities in text documents, which essentially induce a special kind of edges to the knowledge base, and allows one to traverse these edges in both directions. These links open up many opportunities for QA reasoning, *e.g.* retrieving all the information about the entity by going from a mention to a KB entity, finding relations between entities by retrieving text passages that mention both of them, extracting candidate evidence by retrieving passages that mention question and answer entities along with some question terms, and so on. First, in Section 3.2.1 we describe how external text data can help to improve the performance of knowledge base question answering, and in Section 3.2.2 I propose a novel hybrid question answering architecture.

3.2.1 Text2KB: Knowledge Base Question Answering using External Text Data

We now introduce our system, called Text2KB, that expands upon the basic KBQA model by incorporating external textual sources throughout the QA process. The general architecture and an example use case of Text2KB is presented on Figure 3.3. The left part of the figure roughly corresponds to the architecture of the existing information extraction approaches to KBQA, described above. The right part introduces additional external text data sources, specifically we investigate the use of web search results, community question answering (CQA) data, and a large collection of documents with detected KB entity mentions. Recall that the main challenges in KBQA are linking topical entities in the question to the KB; identifying candidate answers in the neighborhood around the question entities; and ranking the candidates. In the rest of this section we present our approach to solving each of these challenges, by: using web search results, CQA data, and external corpus statistics.



who is the woman that john edwards had an affair with

John Edwards extramarital affair - Wikipedia, the free ...
https://en.wikipedia.org/wiki/John_Edwards_extramarital_affair
... that Edwards had engaged in an affair with Rielle Hunter ... I recognized my mistake and I told my wife that I had a liaison with another woman, ...
Rielle Hunter and ... · Initial National ... · Hotel encounter with ...

Edwards admits to extramarital affair - CNN.com
www.cnn.com/2008/POLITICS/08/08/edwards.affair/index.html
Aug 08, 2008 · Former U.S. senator and Democratic presidential hopeful John Edwards admitted ... an extramarital affair; The woman, ... Sen. John McCain ...

John Edwards Admits Having An Affair - CBS News
www.cbsnews.com/news/john-edwards-admits-having-an-affair
Former U.S. presidential candidate John Edwards, ... admitted in shame Friday he had had an extramarital affair with a woman who ... He was John Kerry's running ...

Who was it that John Edwards had an affair with?

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."

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 3.3: The architecture of our Text2KB Question Answering system

Web search results for KBQA

To obtain related web search results, Text2KB issues the question as a query to a commercial web search engine[8], extracts top 10 search result snippets and the corresponding documents. Next, it detects KB entity mentions in both snippets and documents. This data turns out to be useful for multiple purposes, *i.e.* question entity identification and answer candidate ranking.

Question entity identification. 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 whole 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. To keep only the entities that are also mentioned in the question and avoid irrelevant entities, we use string distance. More specifically, we take names of all entities detected in the question and compute their term by term similarity with non-stopwords from the question. In this work we used Jaro-Winkler string distance and entity was added to the list of question entities if at least one of its tokens e_t has high similarity with one of the question tokens q_t excluding stopwords ($Stop$):

$$max_{e_t \in M \backslash Stop, q_t \in Q \backslash Stop} distance_{Jaro-Winkler}(e_t, q_t) \geq 0.8$$

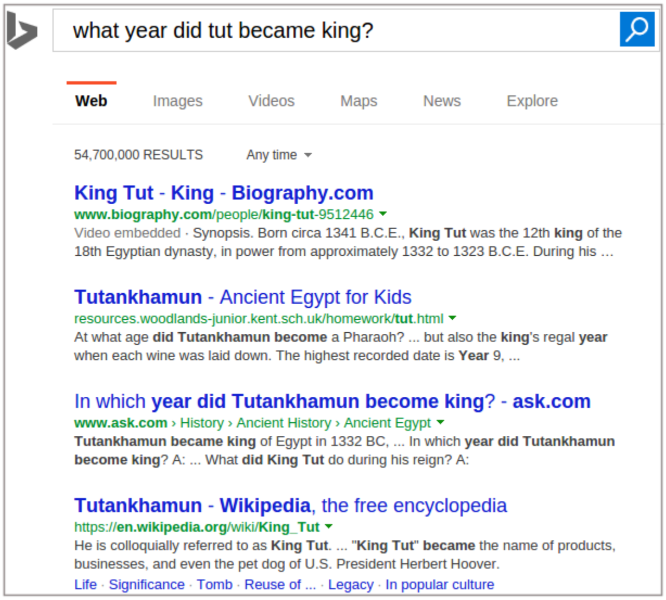
Answer candidate features. Most of the information stored in knowledge bases is also present in other formats, including natural language statements, tables, *etc.* For example, on Figure 3.4 multiple snippets mention the date when Tutankhamun became the king. Text-QA systems usually generate answer candidates from passages extracted from retrieved documents. In our case candidates are already generated from a KB and we just need to rank them to select the best one. Text2KB uses snippets and documents to compute a set of features, which are used for answer candidate ranking. More specifically we do this following:

1. Precompute term and entity IDFs[9]. We used Google n-grams corpus to approximate terms IDF by collection frequencies and available ClueWeb Freebase entity annotations[10] to compute entity IDFs
2. Each snippet and document is represented by two TF-IDF vectors of lowercased tokens and mentioned entities
3. In addition, vectors of all snippets and all documents are merged together to form combined token and entity vectors
4. Each answer candidate is also represented as TF-IDF vectors of terms (from entity names) and entities
5. We compute cosine similarities between answer and each snippet and document token and entity vectors. This gives us 10 similarity scores for every document for token vectors and 10 similarities for entity vectors, we take average and maximum scores as features.
6. We do the same for the combined document and use cosine similarities as features.

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

[9] https://en.wikipedia.org/wiki/Tf-idf

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



what year did tut became king?

54,700,000 RESULTS

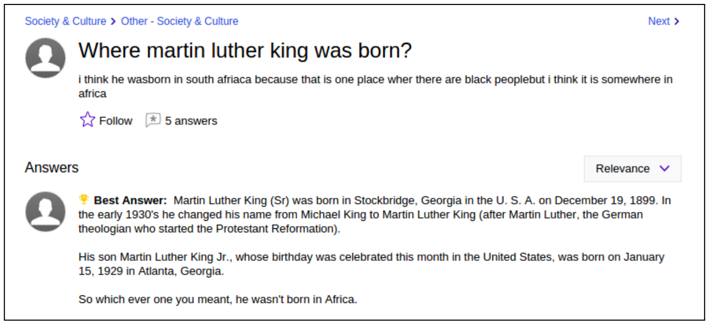
King Tut - King - Biography.com
www.biography.com/people/**king-tut**-9512446
Video embedded · Synopsis. Born circa 1341 B.C.E., **King Tut** was the 12th **king** of the 18th Egyptian dynasty, in power from approximately 1332 to 1323 B.C.E. During his …

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 3.4: 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



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

Answers

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

CQA data for Matching Questions to Predicates

Recall that a major challenge in KBQA is that natural language questions do not easily or uniquely 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 answer to learn this mapping. Unfortunately, manual labeling of questions with answers is expensive, and necessarily contains only a small fraction of the different ways the same KB predicate can be inquired about using natural language questions. Researchers have proposed to use weakly supervised methods to extend the lexicon with mappings learned from *single sentence statements* mentioning entity pairs from a large corpus [147]. However, often there is a lexical gap between how information is asked about in a question and how it is expressed in a statement. On the other hand there are huge archives of questions and answers posted by real users on various community question answering websites, *e.g.* Figure 3.5.

For our experiments we use 4,483,032 questions from Yahoo! Comprehensive Questions and Answers WebScope dataset[11]. Texts of each question and answer pair were run through an entity

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

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

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

linker, that detected mentions of Freebase entities. Next, similar to an idea of relation extraction from CQA data [105], we use distant supervision to label each question-answer pair with predicates between entities mentioned in the question and in the answer. As a result, we have a set of questions, annotated with KB predicates, which are, often incorrectly, assumed to answer the question. We learn the associations between question terms and predicates by computing pointwise mutual information scores[12] (PMI) for each term-predicate pair. Examples of scores for some terms from WebQuestions dataset questions are given in Table 3.4.

Although noisy, the statistics look reasonable to be used for candidate ranking. In Text2KB we take candidate answer predicates and look up the PMI scores between these predicates and terms in the question. Missing pairs are given a score of 0, and minimum, average and maximum of these scores are used as features. Since this kind of data is usually sparse, we also use pretrained word2vec word embeddings[13] to generate predicate embeddings by taking weighted average of term vectors from predicate's PMI table. Each term's embedding vector is weighted by its PMI value (terms with negative score are skipped). Then, we compute cosine similarities between predicate vector and question term vectors and take their minimum, average, maximum as features. Similarity between the predicate vector and average question term vector is also computed.

Estimating Entity Associations

When ranking candidate answers, we are interested in estimating whether the entities in the question and the answer are related in a way asked in the question. Existing systems usually look on how candidate predicates are expressed in questions and statements. But predicate isn't the only way we can look at this. An alternative is to consider text passages, *e.g.* sentences, that mention topical and answer entities together. For example, in the bottom right corner of Figure

[12] https://en.wikipedia.org/wiki/Pointwise_mutual_information

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

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

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

3.3 we can see some passages that mentioned a pair of people, and the context of these mentions often expresses the nature of the relationships between the entities.

We use the ClueWeb12 corpus with existing Freebase entity annotations[14] and compute counts of different terms that occur in the context to an entity pair mention. By an entity pair mention we mean a pair of mentions of different entities within 200 characters of each other. We take terms in between mentions and 100 character before and after mentions as the context. A small sample of this data is presented in Table 3.5.

First, given a set of question terms Q and an answer candidate, that starts from a question entity e_1, we compute a language model score for every answer entity e_2:

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

and use minimum, average and maximum as features. To address the sparsity problem, we again use embeddings, ie 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 tokens vector 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. In Freebase, most of the entities contain a description paragraph, which often comes from the entity Wikipedia profile. These descriptions of entities in the KB were found useful for text-based question answering [118]. For completeness, we include them in our system. The aim is to improve the matching of the question text, to the unstructured description of the candidate entities. For this, each entity description is represented as a vector of tokens, and a vector of mentioned entities. We compute cosine similarities between the question tokens and each of the candidate entity vectors, and use these scores as features for candidate ranking. In future work, we could explore incorporating any other entity profile text, such as full Wikipedia article.

[14] http://lemurproject.org/clueweb12/FACC1/

Table 3.6: Performance of the Text2KB system on WebQuestions dataset compared to the existing approaches. The difference from the baseline Aqqu system is significant with p-value ¡ 0.01

System	avg Recall	avg Precision	F1 of avg Prec and Recall	avg F1
SemPre [15]	0.413	0.480	0.444	0.357
Subgraph Embeddings [26]	-	-	0.432	0.392
ParaSemPre [16]	0.466	0.405	0.433	0.399
Jacana [147]	0.458	0.517	0.486	0.330
Kitt AI [145]	0.545	0.526	0.535	0.443
AgendaIL [17]	0.557	0.505	0.530	0.497
STAGG [122]	0.607	**0.528**	**0.565**	**0.525**
Aqqu (baseline) [13]	0.604	0.498	0.546	0.494
Our system: Text2KB	**0.6354**	0.5059	0.5633	0.5223

Evaluation

We followed the standard evaluation procedure for the WebQuestions dataset and used the original 70-30% train-test split, which results in 3,778 training and 2,032 test questions. Since each answer is potentially a list of entities a^*, the quality of an answer a is represented by F1-score:

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

where $precision(a^*, a) = \frac{|a^* \cap a|}{|a|}$ and $recall(a^*, a) = \frac{|a^* \cap a|}{|a^*|}$.

We also report average precision and recall, as well as an F1 score of average precision and recall. The results of existing approaches, our baseline and Text2KB systems is presented in Table 3.6.

As we can see, Text2KB significantly improves over the baseline system and reaches the current best published result - STAGG [122], and we believe that this system will also benefit from the ideas of our work, and we will explore this question in Section **??**.

Ablation Study. To study effects of different components in isolation we made a series of ablation studies. For convenience, we introduce the following notations for different components of our system:

- T - notable type score model as a ranking feature
- DF - date range filter-based query template
- E - using web search result snippets for question entity identification
- W - using web 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 `+<component>` to for a system with a certain component added, and `-<component>` when the component is removed. For example, the baseline

Table 3.7: Average Recall (R), Precision (Pr), and F1 of Aqqu (baseline), Text2KB (our system), and variations of TextKB with respective components removed. * indicates significant differences at p¡0.05.

System	R	Pr	F1
Aqqu (baseline)	0.604	0.498	0.494
Text2KB -E-W-CQA-CL=Aqqu +DF+T	0.6169	0.4807	0.4987
Text2KB -W-CQA-CL	0.6272*	0.4920*	0.5083*
Text2KB -E	0.6344*	0.4966*	0.5140*

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

System	R	Pr	F1
Text2KB -W	0.6327	0.4960	0.5126
Text2KB -CQA	0.6420	0.4987	0.5185
Text2KB -CL	0.6444	0.5047	0.5228
Text2KB (Web search results only)	0.6423	0.5028	0.5216
Text2KB (ClueWeb only)	0.6307	0.4978	0.5138
Text2KB (CQA only)	0.6224	0.4928	0.5077

system will be denoted as "Aqqu" according the authors notation. 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 -E-W-CQA-CL". Our full system "Text2KB" can be also denoted as "Aqqu +DF+T+E+W+CQA+CL".

The first question that we are asking is what are the improvements, introduced by adding date range filter templates, notable type model, entity linking from web search results and text-based features generated from all the different sources. Results of this ablation experiment are presented in Table 3.7. As we can see, additional date range filters and notable types model (Text2KB -E-W-CQA-CL) are responsible for an increased recall and a drop in precision compared to the baseline model. Detecting question entities (Text2KB -W-CQA-CL) help improve both precision and recall, and therefore average F1 score by 0.096 points. An even bigger improvement is achieved by introducing all our external text-based features, and since these improvements are independent, their combination boosts the performance even more.

Now, let's look into the relative importance of each of the data sources, we will remove or use a group of web search, cqa or clueweb-based features and see how the performance of the whole system changes. Table 3.8 summarizes the results of these experiments.

Features that we generate from web search results are the most effective, because even without other data sources the QA performance is almost as high as the full system. In addition, if we remove web search results based features the performance drops more than for other text data sources. Features based on ClueWeb entity pair statistics perform better than CQA-based features.

Since we used each data source to generate multiple different features for candidate ranking, it is interesting to see which particular features are more useful than others by the ranking machine learning algorithm (we used random forest). Figure 3.6 plots a subset of features ranked by their Gini index-based importance scores in the final answer candidate ranking model.

The figure supports the observation that web search results features are the most useful, however, other text data sources also contribute to the improvement.

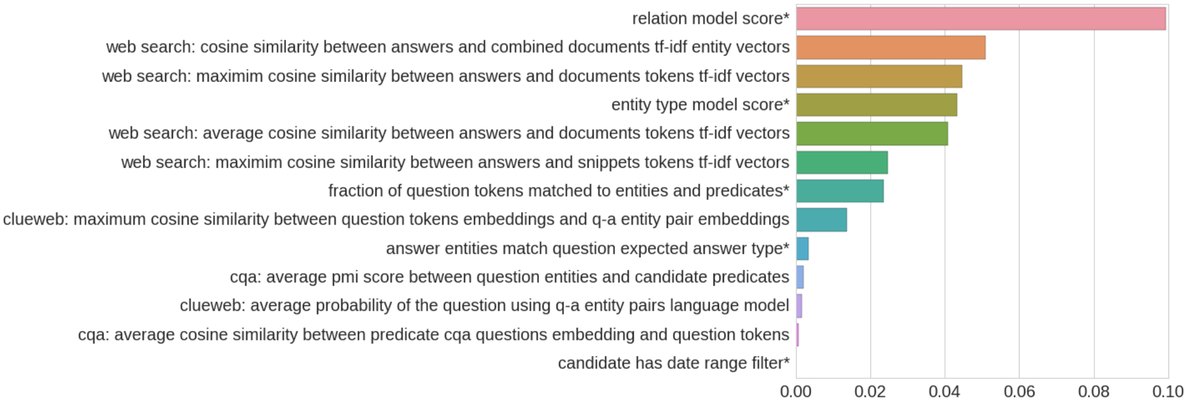


Figure 3.6: Importances of different text-based features for KBQA (features with * are not text-based and are provided for comparison)

Table 3.9: Average Recall (R), Precision (Pr), and F1 for Text2KB (our system), STAGG and their combinations

System	R	P	F1
Our system: Text2KB	0.6354	0.5059	0.5223
STAGG [122]	0.607	0.528	0.525
Text2KB + STAGG	0.5976	0.5343	0.5320
Text2KB + STAGG (oracle)	0.7144	0.5904	0.6056

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 have shown that Text2KB outperforms the baseline. 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 [122]. The differences between our baseline system Aqqu and STAGG lie in the components, *i.e.* entity linking algorithm, a set of query templates and ranking methods, therefore our approach is complementary and should be helpful. 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

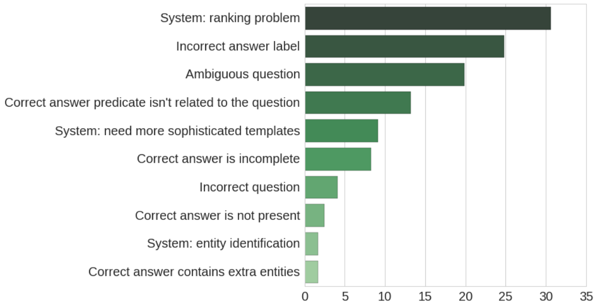


Figure 3.7: Distribution of problems with questions, where Text2KB returns an answer with F1<1

achieves slightly better average F1 score. Alternatively, we can look at the oracle combination of the systems, which always chooses an answer with higher F1. This experiment shows that that systems don't make exactly the same mistakes and therefore can be combined. As we can see such a combination results in a performance of 0.6056, which is much higher than either of the systems.

WebQuestions dataset is rather small as a result, answers to 112 of the test questions involve a predicate 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[15].

To get a better insights of the problems that remain, we collected 1219 questions for which Text2KB didn't return completely correct answer, *i.e.* F1 score of the answer is less than 1. We manually looked through a couple of hundreds of these examples and grouped the problems into several clusters. The results are summarized on Figure 3.7.

As we can see candidate ranking is still the major problem, and it accounts for 31% of the cases. The second most popular problem is incorrect ground truth labels (almost a quarter of errors). For example: for the question *when tupac was shot?"* the label says `Tupac 1994 assault` instead of `Las Vegas`. Another set of questions have 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. The profile page shows only a sample of 10 entities from large lists and the others are hidden behind the "NNN 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 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 predicate related to the olympic sport competitions country participated in[16], which obviously isn't related to the question.

As for the system errors, there are wins and loses introduced by each of our components. 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

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

[16] `olympics.olympic_participating_country.athletes`

evidence that romo refers to `Tomo 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.* [40].

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.

In summary, we show that ideas behind Text2KB could be integrated into other systems and improve their performance. The error analysis suggested that even though a significant number of questions in the WebQuestions dataset have incorrect or ambiguous ground truth labels, there is still a room for improvement. In particular, the future work for Text2KB will include a better strategy for entity linking using external data sources and a better context model for entity mentions in text documents, which can put more weight on entities mentioned in the context related to the question.

Conclusion

Our work showed that unstructured text resources can be effectively utilized for knowledge base question answering to improve query understanding, candidate answer generation and ranking. We focused on three particular techniques and associated text information sources: web search results for query understanding and candidate ranking, community question answering data for candidate generation, and text fragments around entity pair mentions for ranking. Certainly, there are more resources that could be potential adapted, *e.g.* entity profile pages like Wikipedia, news sources, textbooks, and many others. However, we believe that the proposed approach is general enough that it could be extended and successfully incorporate these other diverse text sources.

In the future, we plan to extend our work to the more open setup, similar to the QALD hybrid task, where questions no longer have to be answered exclusively from the KB. This would require extending the described techniques, and creating new QA benchmarks.

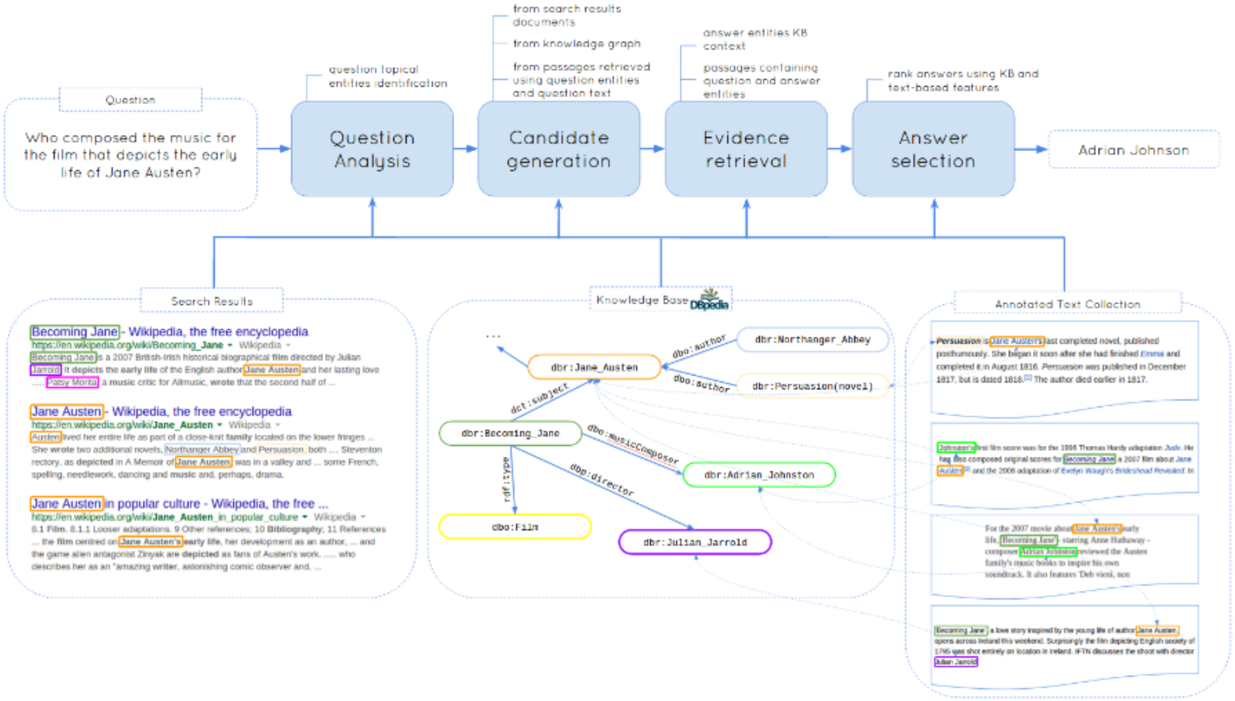

from search results documents
from knowledge graph
from passages retrieved using question entities and question text
question topical entities identification
answer entities KB context
passages containing question and answer entities
rank answers using KB and text-based features
Question
Who composed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

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

3.2.2 Hybrid Question Answering over Knowledge Bases and Semantically Annotated Text Collections

Experiments of the previous chapter demonstrated that external text data can be useful for knowledge base question answering systems. However, such an approach can only help with the problem of lexical gap, *i.e.* mapping from natural language phrases to knowledge base concepts. It doesn't deal with the problem of KB incompleteness, *i.e.* in Text2KB system text resources cannot compliment the knowledge from the database, it only extends its representation.

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 QA pipeline are the following:

- **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 [3]) as a query and use an approach similar to [41] to detect question topical entities

- **Candidate generation from text**: extract candidate answer (or intermediate answer) entities with evidence from the retrieved text documents using existing techniques, e.g. [123].
- **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
- **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
- **Rank candidate**: rank candidate answers using evidence extracted from the KB as well as from text

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.

Evaluation

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 a semantically enriched text-based system QuASE [118], a hybrid YodaQA system, designed to be an open source analogue to IBM Watson [14] and open question answering approach of [51].

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 [118, 123]. Unlike most of the previous approaches, however, the proposed system aims at retrieving KB entities as answers of the questions. Therefore, I'm planning to annotate TREC QA datasets answers with KB entity identifiers. The proposed system can use the web as the corpus and query it using Bing Search API[17]. dbPedia, Freebase and Reverb extractions [50] are examples of schema-based and open knowledge bases that can be used for the experiments. The metrics used for evaluation typically include accuracy and mean reciprocal rank (MRR).

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

Most of the recent work on knowledge base question answering and semantic parsing have been evaluated on the WebQuestions dataset [15]. I will compare the proposed approach with previous results [18] on this dataset. Again, web can be used as a text collection which can be queried using Bing Search API.

New factoid question answering dataset. WebQuestions dataset has certain limitations. Most of the questions mined using Google Suggest API have very similar structure and lexicon, which makes it easier for systems to learn a mapping from natural language phrases to KB predicates. 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. To overcome this limitations I'm designing a new factoid QA dataset, derived from questions posted to Yahoo! Answers CQA website. A set of heuristics can be used to filter factoid questions from more general opinion, recommendation, *etc.* questions. 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.

3.3 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.

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 next Chapter.

[18] http://goo.gl/sePBja

4 Non-factoid Question Answering

In this chapter I summarize the proposed work in developing a non-factoid question answering system. In particular, Section 4.1 described the general architecture of the system, and the following chapters describe the proposed improvements to different stages of the pipeline.

4.1 The architecture of the system

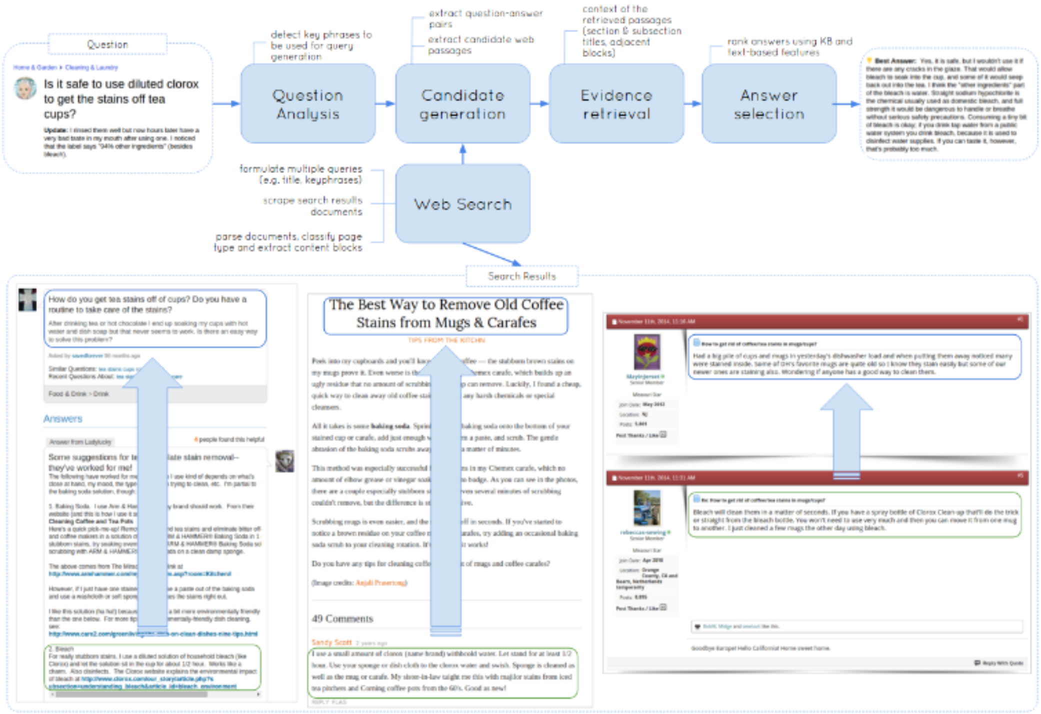

Question
Home & Garden > Cleaning & Laundry
Is it safe to use diluted clorox to get the stains off tea cups?
detect key phrases to be used for query generation
Question Analysis
extract question-answer pairs
extract candidate web passages
Candidate generation
context of the retrieved passages (section & subsection titles, adjacent blocks)
Evidence retrieval
rank answers using KB and text-based features
Answer selection
Best Answer:
formulate multiple queries (e.g. title, keyphrases)
scrape search results documents
parse documents, classify page type and extract content blocks
Web Search
Search Results
How do you get tea stains off of cups? Do you have a routine to take care of the stains?
Answers
The Best Way to Remove Old Coffee Stains from Mugs & Carafes
TIPS FROM THE KITCHEN
49 Comments

Figure 4.1: Using web page structure information for non-factoid question answering

The architecture of the system I develop is somewhat standard and contains 4 main stages: question analysis, candidate generation, evidence retrieval and answer selection. Figure 4.1 depicts the question-answering pipeline.

4.1.1 Question Analysis

Questions that users post to CQA websites often contain title and body and can be relatively long and contain multiple important and not so important context information and details. The success of the retrieval-based question answering system depends on the information it finds in a collection. Extra long search queries are not very efficient and can return few or zero results. Therefore, there is a problem of summarizing the user question. Some strategies often used include considering first part of the question, *e.g.* title of the question only. However, often the most important part

of the question isn't the title, or the title doesn't contain all the crucial information, *e.g.*
Title: *Diet please please help asap?*
Body: *I want to lose weight, at least 4 stone but I don't know what I should eat :(what should I have for breakfast lunch and dinner? Should I exercise a little? Please help!!*

To summarize the question I propose to use recent advances in the field of deep learning, in particular a model similar to [101]. In more detail, I propose to train neural network based summarization model to generate the summary of the question, which will be used as a query to retrieve similar questions. Such a model can be specifically trained to maximize retrieval performance on a collection of question-answer pairs, retrieved by systems in LiveQA TREC 2015 and labeled by NIST assessors.

4.1.2 Structure of the Web Page for Candidate Generation and Scoring

The diversity and complexity of non-factoid questions pose additional challenges for automatic question answering systems. To answer such questions a system often needs to provide a whole paragraph of text, *e.g.* TREC LiveQA'15 limits the answers to a maximum of 1000 characters. Therefore, a candidate answer becomes much longer, which requires additional attention on candidate generation and ranking stages.

Existing techniques usually extract one or more consecutive sentences not exceeding the maximum answer length, pool them together and rank using certain feature representation, by large ignoring the context information from the page where the answer was extracted from. In the system I develop I propose to utilize the structure of the web page for both candidate generation and scoring.

To generate better candidates I'm going to utilize the structure of retrieved web pages. The previous analysis showed [104], that many search results retrieved for the question are FAQs, forums or other community question answering websites. From such resources it's beneficial to extract question answer pairs, which can be done either be designing wrapper for popular websites, utilizing semantic annotations, such as https://schema.org or by applying some of the automatic question-answer extraction methods [39]. For other resources, page segmentation techniques, similar to [34], can split a page into semantically information blocks, which will help to make sure that candidate don't contain disjoint and unreadable information.

After a candidate passage is extracted, we need to score it as a potential answer. Often, a passage taken out of context is hard to understand even for human. Therefore, I propose to include the context information, *i.e.* some features, representing the web page where the passage was taken from and its location there (*e.g.* adjacent passages, which are often related [143]).

4.1.3 Answer Summarization

Unlike factoid questions, where evidence from all retrieved passages is usually aggregates, a traditional non-factoid question-answering system simply returns the top scoring passage as the answer. However, such an answer can contain a lot of redundant or irrelevant information, whereas other good candidate passages may contain supplemental information or different opinion. Therefore, an idea to summarize passages and generate the final answer sounds natural in this scenario. The winning approach from TREC LiveQA'15 included a combination strategy, that simply put together multiple top scoring passages while the answer doesn't exceed the maximum length [127]. In my thesis I'm planning to explore the answer summarization problem in more detail. More specifically, I propose to explore deep learning techniques, that lie in the core of recent successes in text generation, *e.g.* image caption generation and machine translation models [12, 138]. Answer summarization problem can be posed as answer generation problem using recurrent neural network, that has information about the question and retrieved passages, and it can be trained using existing CQA question-answer pairs and retrieved passages, that can be assumed as the source of the answers. Alternative and simpler approach is to solve this problem as sequential answer selection problem, where a model is trained to predict the next sentence in the answer. Such a model can be trained on a collection of questions and answer sentences, which will provide the model information on both answer discourse coherence and relevance.

4.2 Crowdsourcing for Real-time Question Answering

The first iteration of TREC LiveQA shared task in 2015 was pretty successful, with the winning system able to automatically return a reasonable answer to more than half of the submitted questions, as assessed for TREC by the trained judges from NIST. Nevertheless, many questions were unable to be answered well by any of the participating systems. As much as we would like an automated system to handle all user questions equally good, there will always be cases, when it is unable to come up with a good response. Instead of returning a non-sense or unusable answer, an automated system can ask for an external help, *e.g.* using crowdsourcing. Additionally, an automated system can consult with a human crowd, which can provide some information useful for the system to rank and select the answer. In this section, we first explore if crowdsourcing can be used in these two ways to help an automated system answer complex user questions in near real-time scenario, *e.g.* within a minute, and then I propose some future directions on incorporating crowdsourcing in a real QA system.

4.2.1 Experiments

We conducted a series of crowdsourcing experiments designed to answer the following questions:

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

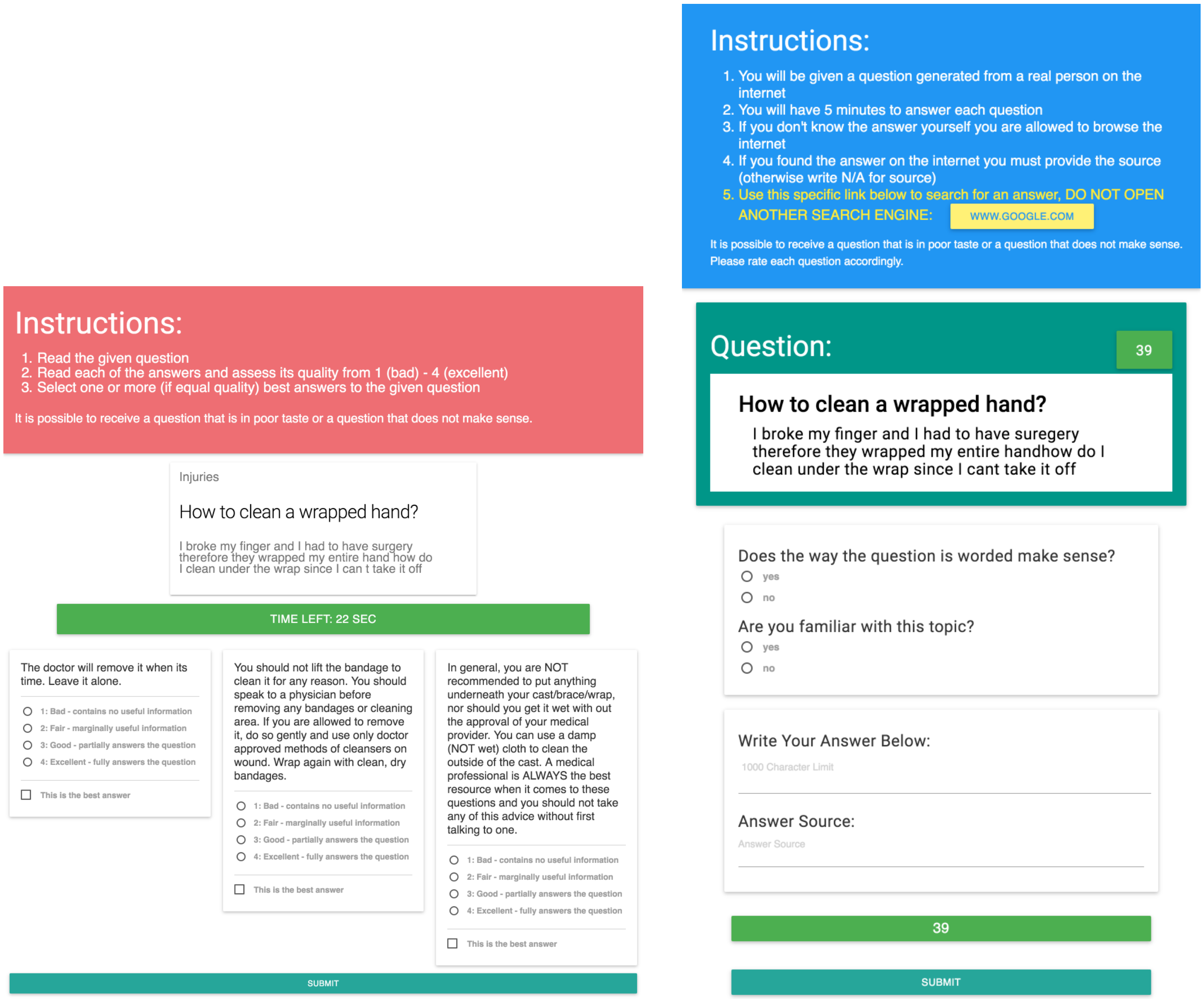
2. Is it possible to use crowdsourcing to collect answers to real user questions under a time limit?

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?

In our experiments we used the Amazon Mechanical Turk crowdsourcing platform[1], and focused on the 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

[1] http://mturk.com

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



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

SUBMIT

(a) Answer validation form

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

39

SUBMIT

(b) Answer crowdsourcing form

Figure 4.2: User interfaces for crowdsourcing answers and their judgements for TREC LiveQA questions

(August 31, 2015). For these questions we also crawled their community answers, that were eventually posted on Yahoo! Answers[3].

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

[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.

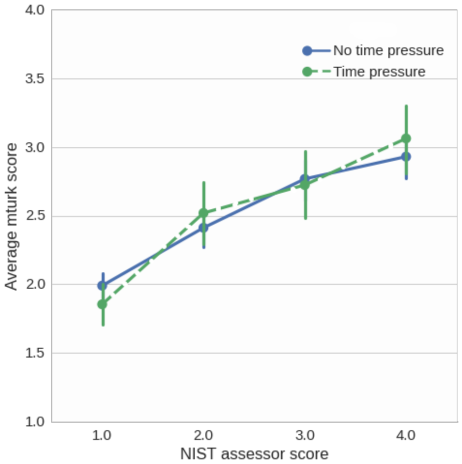


Figure 4.3: Correlation between NIST assessor scores and crowdsourced ratings with and without time limit on the work time

with a 1 minute countdown timer shown to them, while the other could complete the task without worrying about a time limit. Within each group, we assigned three different workers per question, and the workers were compensated at a rate of $0.05 per question for this task.

The interface for collecting answer ratings is illustrated in Figure 4.2a[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 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.

In this experiment we collected 6 ratings (3 with and 3 without time pressure) for each of three answers for a sample of 100 questions, which makes it a total of 1800 judgments. Each answer also has an official NIST assessor rating on the same scale. Figure 4.3 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 4.4 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.

Next, we studied if 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

[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

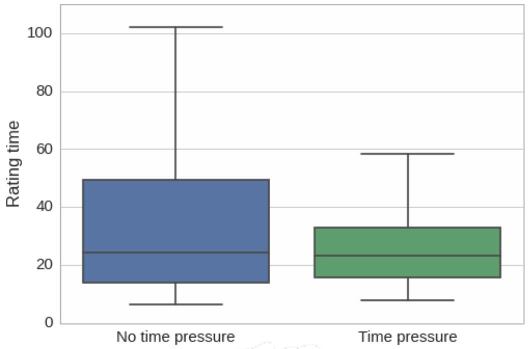


Figure 4.4: Box plot of answer rating time with and without time pressure

workers received a $0.10 compensation for each answer. The form for answer crowdsourcing is shown in Figure 4.2b, 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

We were able to collect 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 4.1 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

[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

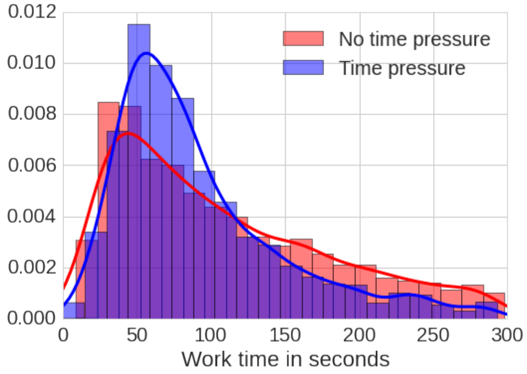


Figure 4.5: Distribution of answering times for experiments with and without time pressure

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.

Table 4.1: Statistics of different types of answers for Yahoo! Answers questions

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

From Figure 4.5 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.

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 for a sample of 100 questions. Each answer was judged by 3 different workers (without time pressure), and their scores were averaged. Figure 4.6 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 4.7) sheds some light on the nature of the problems with automatic and human answers. The automated 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 initial results show that crowd workers are capable of validating a small set of answer candidates quickly, which could be potentially incorporated into an automatic QA system for

[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

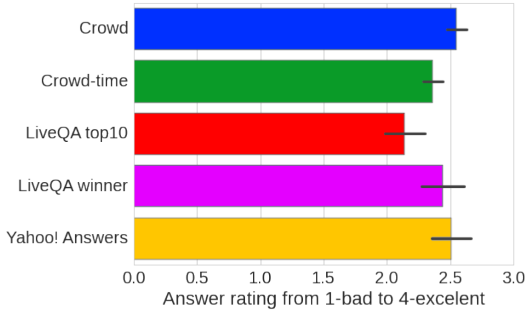


Figure 4.6: Average scores of different types of answers to Yahoo! Answers questions

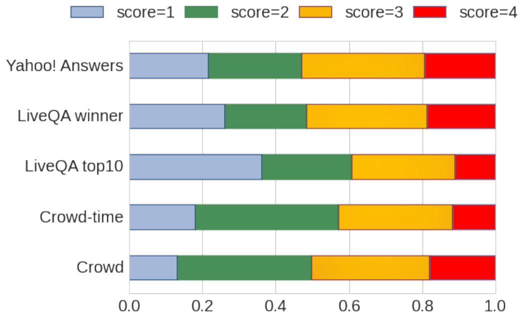


Figure 4.7: Distribution of scores for different types of answers to Yahoo! Answers questions

answer validation and reranking. In addition, even one minute appears enough for a crowd to generate a fair or good response to most real questions drawn from a CQA site, which can be useful in case a QA system didn't have good candidates in the first place. The quality of crowdsourced answers was comparable to the original Yahoo! Answers responses, and even with time pressure, crowdsourcing was shown promising to complement or augment automated QA systems. In the next section, I describe the proposed work on incorporating crowdsourcing into a real QA system.

4.2.2 Integrating Crowdsourcing into a Real QA System

There are a number of questions, that need to be addressed in order to build an efficient real-time human-computer question answering system. I'm planning to add crowdsourcing as a component to my TREC LiveQA 2016 system, which would require solving the problems of obtaining and retaining a set of workers for the duration of the challenge, designing a good user interface, developing a proper interaction model between an automated system and the workers, *etc.*

The crowdsoucing module for our LiveQA system will represent a website, where workers from Amazon Mechanical Turk will enter and stay until we give them a task to do. Here, we are building on the previous research on obtaining a crowd for real-time tasks, *e.g.* [19]. The workers will be payed a small fixed rate for accepting the HIT and entering our QA interface and then by time they spent there responding and validating questions. Inactive workers will be logged out of the system if they don't respond to an incoming question. Figure 4.8 shows the user interface of the crowdsourcing module of our TREC LiveQA system. It displays the question and some potentially useful information for answering it (*e.g.* search results). A worker can type his own answer if he happen to know it, or she can judge the provided list of candidate answers.

The interactions between the automated system and the crowdsourcing module will look the

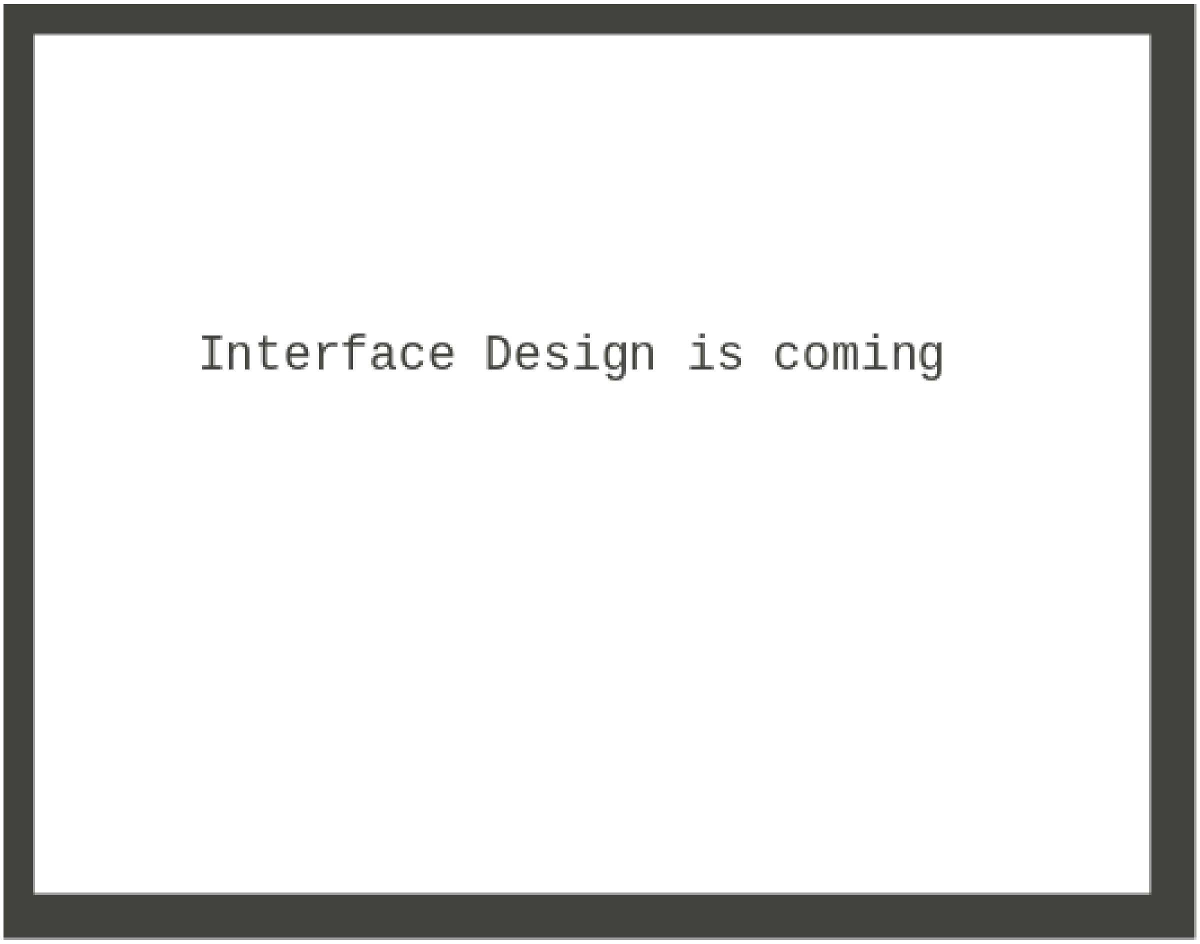


Figure 4.8: User interface for the crowdsourcing module of TREC LiveQA system

following way:

1. When a system receives a question, it is sent to the crowdsourcing UI, so workers could read it and possibly start responding if they happen to know the answer
2. Then a system generates a set of candidate answers, score them and send top candidates to the crowdsourcing module for validation
3. At the end of the alloted 1 minute period the crowdsourcing module responds back with the answer a worker possibly wrote as well as scores for validated candidate answers
4. The automated system aggregates this information and chooses whether to return the worker answer or to go with one of the existing candidates.

4.3 Evaluation

Evaluation of complete non-factoid question answering systems is complicated due to the variability of answer language, the quality of which is impossible to estimate using manually created answer patterns (as is the case for factoid TREC QA dataset). A manual judgment of answers is needed, and luckily TREC LiveQA 2016 is offering such an opportunity. The model I'm developing will participate in the shared task, which will allow us to evaluate it against other competing approaches.

The analysis of individual components can be performed using labeled data from TREC LiveQA 2015, which includes passages extracted from Yahoo! Answers as well as regular web documents. In more detail, the performance of the answer summarization module will be estimated by similar questions retrieval performance, *e.g.* Precision @ N since we are interested in retrieving more relevant answers in TopN rather than good ranking within the retrieved group. To make results reproducible a collection of Yahoo! Answers QnA pairs from the WebScope[8] and Lucene IR library will be used for question retrieval.

The dataset to evaluate the effectiveness of using web page structure for answer scoring will be derived from TREC LiveQA 2015 labels, which include a big number of passages, that were generated from regular web pages. Therefore, the problem of evaluation of answer scoring methods can be posed as passage ranking problem and metrics, such as Precision@1, can be used as a quality measure.

Finally, the answer summarization module is the most difficult to evaluate, because it's result is a free form text, the quality of which needs to be manually labeled. Therefore, for this task I will refer to the wisdom of a crowd and use Amazon Mechanical Turk to label the quality of answers and compare to the top scoring passages for the same question.

4.4 Summary

The proposed research directions target various aspects of non-factoid question answering pipeline and can help improve both precision and recall of existing systems. The results of this work can help to move in the direction of systems, that produce a natural language response by analyzing the information available on the web. In the next chapter, I will describe the proposed work and results on interactions between a question answering system and users.

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

5 Human Interaction with Question Answering Systems

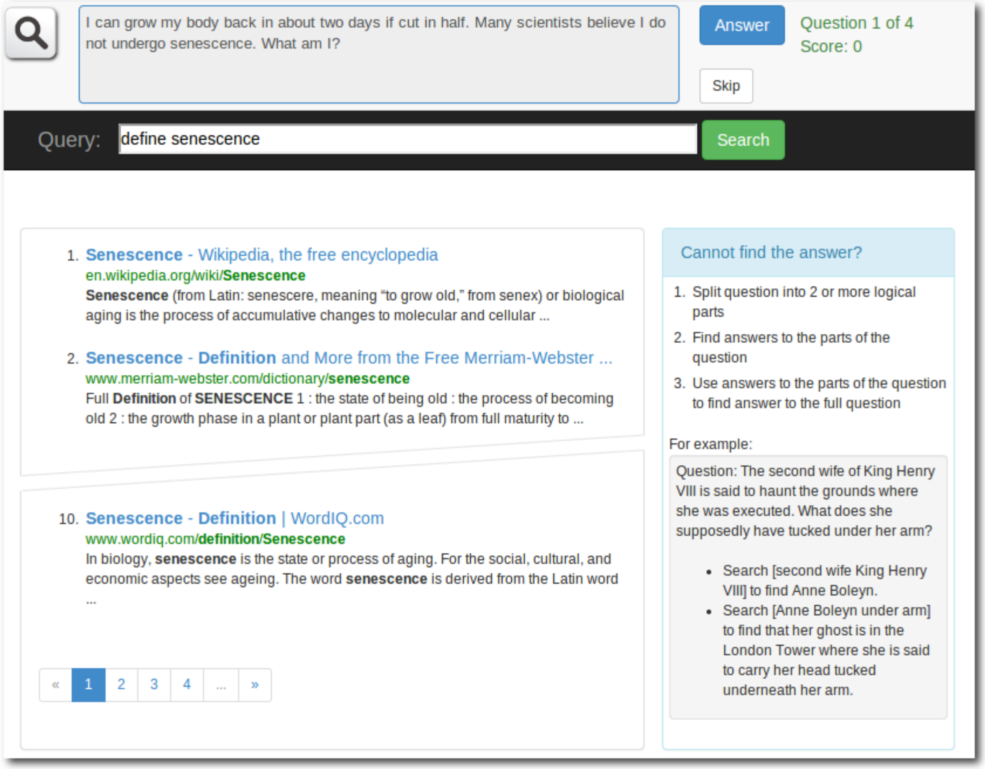
Modern automatic question answering systems are still far from AI machines, that we often imagine or see in the movies. Many user information needs are still left unanswered by existing automatic question answering systems. For example, only 36% of answers of a winning approach from TREC LiveQA 2015 shared task were judged good or excellent. Therefore, in this chapter I will discuss the research on interactions between a human and a question answering system.

More specifically, Section 5.1 describes results on studying the effect of hints on user behavior for solving complex informational tasks. In Section **??** I propose research on using crowdsourcing for improving the performance of a question answering system. And Section 5.2 shifts the focus towards a dialogue between a user and a QA system, in particular towards clarification questions, that are often needed to understand user's question.

5.1 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 [23] reported that about half of the participants of their user study felt frustration when searching. Xie and Cool [137] 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. [94] demonstrated that hints, suggesting certain search engine functionality, help people find answers more quickly, and the effect is retained after a week without hints.

In my thesis I propose to 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.



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?

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 5.1: The interface of the search game used in the study

5.1.1 User Study

To estimate the effect of strategic search hints on user behavior we conducted a study in a form of a web search game similar to "a Google a Day"[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 5.1). 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.

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,

[1] http://www.powersearchingwithgoogle.com
[2] http://www.agoogleaday.com/
[3] http://www.mturk.com/
[4] http://www.bing.com/toolbox/bingsearchapi

Table 5.1: Search tasks used for the study, and specific search hints shown to one of the user groups

	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

which makes them interesting for user assistance research. We filtered search results to exclude all pages that discuss solutions to "A Google a Day" puzzles. To do this we removed pages that mention a major part of the search question or "a google a day" phrase. To keep users focused throughout the whole game we limited the number of questions to 4. The tasks are described in Table 5.1 and were presented to all participants in the same order to ensure comparable learning effects.

The questions have multiple parts and to solve them it is helpful to search 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 5.1). 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 [94], 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

5.1.2 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).

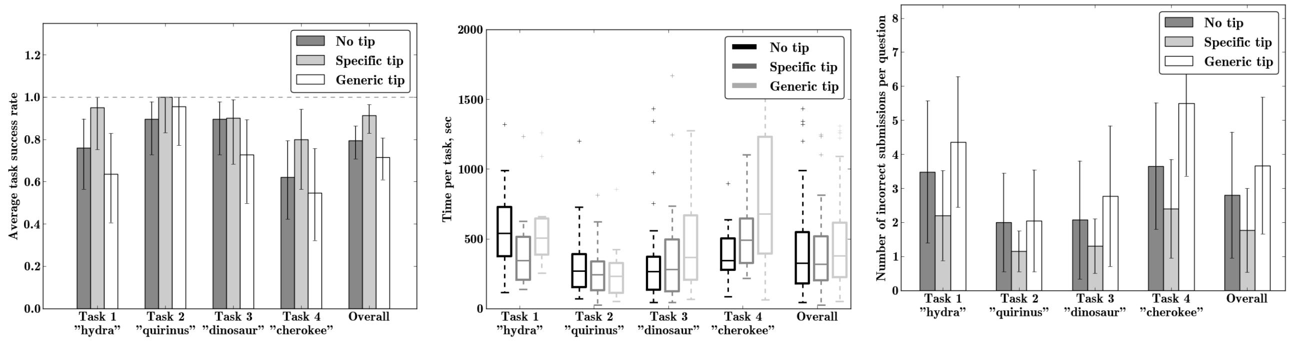
Effects of Search Tips on Performance

In order to measure search success rate we looked at the number of questions answered correctly by different groups of users[5]. Figure 5.2a 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.

The plot of average time to answer a question on Figure 5.2b 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 5.2c 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.

[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

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.

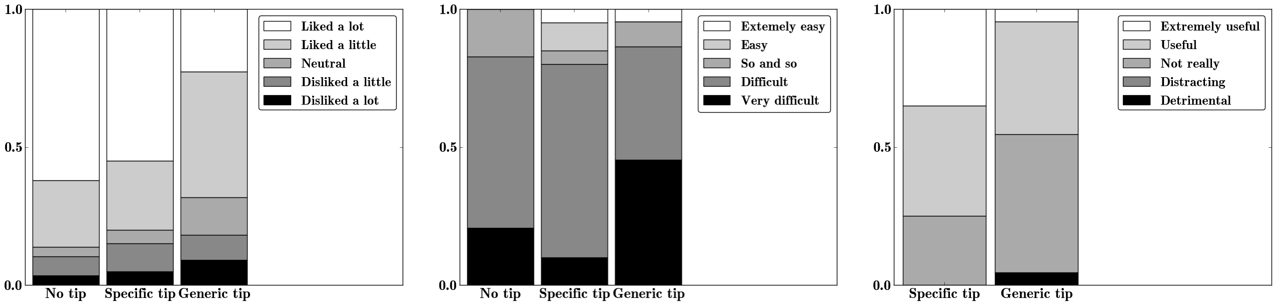


(a) Success rate per task for each group of participants

(b) Task completion time for each group of players

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

Figure 5.2: Results of the user study on the effectiveness of strategic search tips on search task success rate



(a) How did you like the game?

(b) How difficult was the game?

(c) Were search hints useful to you?

Figure 5.3: Proportions of replies to some of the survey question for each group of users

We also looked at other search behavior characteristics: number of queries submitted, 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 5.3 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 5.3c).

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.

5.1.3 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.

We should note, that specific search hints used in this work were manually generated and an interesting question of future work is how to generate such useful hints automatically. It should be possible to learn strategies applied by the experienced search users and suggest them to the rest.

5.2 Clarification Questions

Today, we are observing a possible shift towards natural language interfaces, which will enable richer interaction between a human and computer, in particular question answering. Most of the existing systems are one-sided, *i.e.* they operate by returning an answer in a response to a user question. A richer model will inevitably lead to a dialogue rather than request-response kind of communication. There are many practical aspects of maintaining a dialogue for question-answering. For example, many questions that user ask allow multiple interpretations, *e.g.* “FIND GOOD EXAMPLE”. In such cases a machine could ask a clarification questions to disambiguate certain aspects of the question and provide her with a reasonable response, instead of returning something non-relevant in the first place and force the user to think why did it happen.

Of course, this task is very challenging, and as a first step I propose to study what kind of clarifications do real people typically ask.

5.3 Summary

SUMMARIZE THE CHAPTER.

6 Summary and Discussion

In my thesis I propose several pieces of work towards improving user satisfaction with question answering systems. I plan to consider factoid and non-factoid questions, as usually classified in the community, separately, because certain techniques and data sources are most useful for one type of questions and not another.

The research I'm propose to conduct for improving factoid question answering targets a problem of combining information available in different data sources, *i.e.* structured knowledge bases and unstructured text documents. Semantic annotations of entity mentions in documents create additional connections between knowledge base entities, which should improve KB coverage and allow a system to answer more questions and with better precision. However, such an approach have certain potential limitations.

- A set of additional links for some entities is likely to be big, which makes it impossible to explore them all. Information extraction approaches on the other hand aggregate information over the whole collection, although the extraction process itself brings extra noise.
- ....

For non-factoid question answering I propose several improvements, targeting different stages of the question answering process. Question to query generation neural network model, trained to improve the document retrieval performance, should increase the recall of the question answering system by identifying the key phrases that needed to be search for. The answer passage scoring module should achieve a better precision by analyzing the structure of the answer origin web page, and detecting question-answer pairs and other key structural elements. Finally, the proposed direction in automatic answer summarization is a way to increase the quality of answers by combining evidence from multiple different data sources, possibly providing additional information and alternative opinions. Possible limitations of the proposed directions and approaches are:

- question to query generation model can be retrieval engine specific, which may force the model to be retrained after certain changes in the retrieval algorithm. An alternative strategy is to integrate a similar question summarization module into the retrieval engine itself.
- web page structure?...
- summarization?... The problem is that it might not work well...

Finally, I touch a user aspect of question answering, in particular user assistance with hints in case a system failed to respond to user information needs, clarification questions, which is one of the first steps in dialog-based question answering and finally using the wisdom of a crowd to improve the performance of question answering systems.

- Hints
- Clarifications
- Crowdsourcing

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, 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.

[4] 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.

[5] 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.

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

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

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

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

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

[11] 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.

[12] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.

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

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

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

[28] L. Braunstain, O. Kurland, D. Carmel, I. Szpektor, and A. Shtok. Supporting human answers for advice-seeking questions in cqa sites. In *Advances in Information Retrieval*, pages 129–141. Springer, 2016.

[29] 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.

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

[31] 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.

[32] 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.

[33] M. J. Cafarella, J. Madhavan, and A. Halevy. Web-scale extraction of structured data. *SIGMOD Rec.*, 37(4):55–61, Mar. 2009.

[34] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. Vips: a visionbased page segmentation algorithm. Technical report, Microsoft technical report, MSR-TR-2003-79, 2003.

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

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] M. De Boni and S. Manandhar. An analysis of clarification dialogue for question answering. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–55. Association for Computational Linguistics, 2003.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] O. Etzioni. Search needs a shake-up. *Nature*, 476(7358):25–26, 2011.

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

[50] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011*, pages 1535–1545, 2011.

[51] 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.

[52] A. Fader, L. S. Zettlemoyer, and O. Etzioni. Paraphrase-driven learning for open question answer-

ing. In *ACL*. Citeseer, 2013.

[53] 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.

[54] 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.

[55] 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.

[56] 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*, SIGMOD '11, pages 61–72, New York, NY, USA, 2011. ACM.

[57] 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.

[58] 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.

[59] 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.

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

[61] 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.

[62] 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.

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

[64] B. Hixon, P. Clark, and H. Hajishirzi. Learning knowledge graphs for question answering through conversational dialog. In *Proceedings of the the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA*, 2015.

[65] 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.

[66] 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.

[67] 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.

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

[69] 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.

[70] 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.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

[75] 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.

[76] 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.

[77] 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.

[78] 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.

[79] N. Kushmerick. *Wrapper induction for information extraction.* PhD thesis, University of Washington, 1997.

[80] 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.

[81] 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.

[82] 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.

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

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

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

[86] 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.

[87] 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.

[88] 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.

[89] 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.

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

[91] 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.

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

[93] 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.

[94] 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.

[95] 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.

[96] 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.

[97] 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.

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

[99] 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.

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

[101] 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.

[102] 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.

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

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

[105] 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.

[106] 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.

[107] 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.

[108] 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.

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

[110] 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.

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

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

[113] 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.

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

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

[116] V. I. Spitkovsky and A. X. Chang. A cross-lingual dictionary for english wikipedia concepts. In *LREC*, pages 3168–3175, 2012.

[117] 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.

[118] 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.

[119] 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.

[120] 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.

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

[122] 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.

[123] 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.

[124] 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.

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

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

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

[128] 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.

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

[130] 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.

[131] 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.

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

[133] 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.

[134] 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.

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

[136] 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.

[137] 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.

[138] 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.

[139] K. Xu, Y. Feng, S. Reddy, S. Huang, and D. Zhao. Enhancing freebase question answering using textual evidence. *arXiv preprint arXiv:1603.00957*, 2016.

[140] 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.

[141] 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.

[142] 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.

[143] 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.

[144] 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.

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

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

[147] 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.

[148] 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.

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

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

[151] 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.

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

[153] 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.

[154] 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.