San Francisco State University at LiveQA Track of TREC 2016

Maria Khvalchik and Anagha Kulkarni
maria.khvalchik@gmail.com, ak@sfsu.edu
San Francisco State University

1 Introduction

There are many situations in our everyday life where we look for answers to some questions - “Who wrote this book?”, “How to grill a fish?” or “Where is the Opera House located?”. Twenty years ago to answer these questions people were looking them up in the encyclopedias, recipe books or were asking other people. Moving the information into the electronic form and making it universally accessible helped to automate this process and save an enormous amount of time.

The developing an automatic question answering system is a complex problem which is at the intersection of multiple disciplines - Natural Language Processing (NLP), Information Retrieval (IR) and Machine Learning (ML). A typical question answering system is structured as a pipeline of three components:

- Query formulation - uses NLP techniques to analyze the question and to transform it into a query.
- Document extraction - uses IR/ML techniques to retrieve a list of documents that are likely to contain answers.
- Passage retrieval and ranking - uses ML techniques to select a unit of text (typically, sentences) from the retrieved documents, as the best answer.

Our goal is to build such a system.

2 Objective

Although the problem of answering factoid questions (e.g. "What is the capital of the US?") has been investigated extensively [5, 6, 7, 9, 11, 12], answering open-domain non-factoid questions (e.g. "What is the best way to cook fish?") remains a challenging problem. We concentrate on the latter type of questions, with the added constraint of keeping the response time per question to one minute at most.

In addition, we focus only on query formulation, and passage retrieval and ranking. The document extraction component is delegated to one of popular Web search engines. Specifically, we perform linguistic analysis of the question to transform it into a boolean query with conjunction and disjunction operators. This query is executed against the Web-as-a-corpus using a popular search engine, and the top results are used to compile the answer for the

question. The performance of the system is evaluated using Yahoo! Answers dataset[1] which consists of large set of questions along with the best answers, given and voted by users.

3 Related work

Suryanto et al [1] focus on the Collaborative Question Answering (CQA) task. This task is defined as answering questions using community question answering portals such as Yahoo! Answers, Naver, etc. These portals contain millions of questions and answers contributed by users and can be used as a database to answer questions from any user. The paper defines an algorithm to solve the CQA task which uses the quality of the answers as the key feature.

Lin et al [2] propose a method to identify different variants of the same question with the goal of finding the right answer passage. For instance, “X wrote Y” is the same as “X is the author of Y”. Their approach uses paths in the dependency trees to infer rules about question equivalence. The authors did not try to inject their inference rules into a working QA system, but they took 15 questions from TREC-8 which were manually paraphrased by humans and manually compared the output of the method with these paraphrased questions.

Lee et al [3] used lexico-semantic pattern matching and shallow NLP methods to create high-performance QA system for factoid questions. They define 18 types of answers such as QUANTITY, BODY_PART, LIFE_FORM, etc using the datasets of questions from previous TREC competitions. The type of the answer is then determined by using part-of-speech tagger and by converting the obtained terms into lexico-semantic patterns (LSP). LSP is defined as a pair of condition and conclusion where condition is a regular expression created from LSP grammar and conclusion is the type of answer. In terms of Mean Reciprocal Rank (MRR) their system performed better than the average of all other systems participating in TREC-10.

Cui et al [4] hypothesize that many question answering systems retrieve the wrong passages because of disregarding the semantics of the answer relative to the question. For instance, a question: “What percent of the nation's cheese does Wisconsin produce?” could produce a wrong answer: “the number of consumers who mention California when asked about cheese has risen by 14 percent, while the number specifying Wisconsin has dropped 16 percent.” The proposed solution for that problem is similar to Lee et al [3] that is to analyze the dependency tree and extract similar paths. The difference is that Lee’s paper proposes to generate the variations of the questions using these similar paths while Cui’s proposes to use this similarity when doing matching the question with the answer.

[1] http://webscope.sandbox.yahoo.com

4 Methods

We developed a complete end-to-end QA system which is capable of answering questions with the requirements specified in the objective. IN this section we describe the system architecture, and the various techniques and tools used in the different stages of the QA pipeline.

The architecture diagram for our QA system is depicted in Figure 1. The entry point of the system is the Query Formulation Module (QFM) which is responsible for generating a query from the given question. The resulting query is then sent to Google and Bing search engines and the top *x* results returned by the search engines are downloaded. At this point the Passage Extraction and Ranking (PER) module takes over to identify the best answer for the question. Each of these modules are described in detail next.

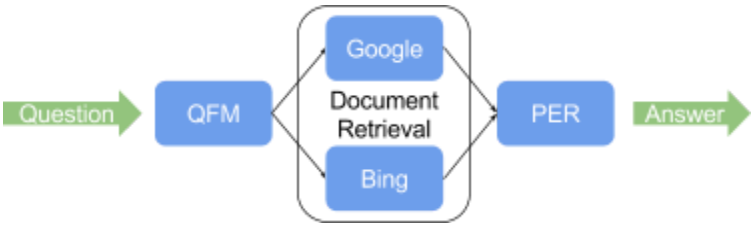

Figure 1. System Architecture

4.1 Query Formulation Module (QFM)

Our approach for QFM was driven by two observations. 1. The *law of diminishing returns* is at play for longer questions. A longer question is not proportionality more informative than a shorter question. Often, the longer question repeats or paraphrases the same information multiple times. Also, longer questions are more likely to contain terms that are only tangentially related to the core question, and thus retrieve non-relevant documents. 2. The classic problem of *vocabulary gap* manifests here as well, where the question and a relevant document use a non-overlapping set of vocabulary and thus lead to a mis-match. For instance, a question about *california* might only specify the postal code *CA* in which case a relevant document that does not use the postal code will not match the question.
Therefore, the main challenge of the query formulation module was keeping the balance between removing the unnecessary words and expanding the query with more terms. To achieve this balance the following approach was developed.

The input question is first analyzed using WordNet part-of-speech parser. The verbs and nouns are retained due to their informativeness. The original question is also analyzed with Stanford dependency parser[2] to select the root word in the question. This is the most important word in

[2] http://nlp.stanford.edu/software/lex-parser.shtml

the question and it is necessary to keep it in the query. All stop words such as “be”, “what”, “the” are removed from the query. We used the list of stopwords from NLTK[3].

For all the remaining words, only the first 6 are retained. It reduces the quality of answers for some long questions, but improves the overall quality of the results. Next each term is expanded with the top 3 synonyms from the WordNet library. The original term and the synonyms are joined with OR operators, and the synonym groups are joined with AND operators. For instance, the question: “What is the capital of California?” is transformed into the following query by the above approach: “(capital) AND (California OR CA)”.

4.2 Document Retrieval

The query created by QFM was then used to obtain matching documents from two sources: Yahoo! Answers and Bing. Specifically, the top three documents returned by Yahoo! Answers, and the top 20 documents returned by Bing for the query were downloaded. Bing and Google were both evaluated for this task, and Bing was chosen since it supports both AND and OR operators, while Google supports only AND.

4.3 Passage Extraction and Ranking

Each of the downloaded webpages is first passed through the following text processing pipeline. The first step of this pipeline extracts the ASCII text from the webpage using a html2text library[4]. We refer to the extracted text as the document.
Next, the document is split into passages, where each passage consists of five consecutive sentences. A sliding span of five consecutive sentences is used to generate the passages. Thus a document containing 6 sentences would generate two passages. This approach generates many passages, specifically, 1 + (n-5). The following rules are used to filter out passages that are not likely to contain the answer. Passage that do not contain any of the query terms, or that contain more than 2 line breaks, or more than 10 punctuation marks, or non-printable symbols are eliminated. Also, passages that are not in English are filtered out. The langdetect[5] library is employed for language identification.

Out of the remaining passages, one passage is chosen as the best candidate answer from that document, as follows. A trained classification model is first employed to categorize each passage as being relevant or non-relevant to the query. (The details about the passage classification model are described in Section 4.4.1.) If none of the candidate passages from a document are classified as relevant or if there are multiple relevant passages, then the passage with the highest BM25 score is selected as the best candidate answer for the document. Thus, one candidate answer is generated by each of the downloaded documents.

[3] http://www.nltk.org/book/ch02.html
[4] https://pypi.python.org/pypi/html2text
[5] https://pypi.python.org/pypi/langdetect?

Finally, a single answer is selected from the above pool of candidate passages using another classification model. A relevant passage, as per this second classification model, that has the highest BM25 score is chosen as the final answer. The details of second classification model are described next.

4.3.1 Passage Classifiers

The first passage classification model is designed to distinguish between relevant and non-relevant passages from the same document. For training this passage classification model a feature vector is generated for each passage. The feature set consists of the BM25 rank of the passage relative to other passages, the ratio of query terms in the passages and the total terms in the passage, the average distance between query terms in the passage, and the length of the passage. The relevance label for each passage in the training set was derived by first estimating its cosine similarity with the best answer on Yahoo! Answers. The highest scoring passages, specifically, the top quartile, were labelled as being relevant, and the bottom quartile as non-relevant. The popular classification algorithm, Support Vector Machines with polynomial kernel was used to learn a binary classification model with the above described labelled data.

The second passage classification model is expected to learn finer distinctions between the passages and thus provide the best passage as the final answer. This is evident in the how the training data for this model is compiled. Only the single best candidate passage from each document is included in the training set for this model. Out of these only the top quartile that exhibit the highest similarity with best human generated answer from Yahoo! Answers are labelled as relevant. The bottom quartile are labeled as non-relevant. The feature set for this model is a superset of the one used by the previous model. Whether the document that sourced the passage was downloaded from Wikipedia is an additional feature. Another feature captures if the source document was downloaded from Yahoo! Answers. Both these features model the source quality. Whether the source document appears in the search results from both, Bing and Google, is another feature. The rank of the document in Bing search results is also a feature. We also tried using Google rank, but it resulted in model of a lower quality. SVM is used to learn a binary classification model using the above described training data.

5 Experimental Setup

A subset of 500 question-answers from the L6 dataset from the Webscope datasets of Yahoo! Labs[6] is used in this work. The first 400 questions in this dataset are used to train the two classification models, and the remaining 100 questions for testing. For the baseline approach we treat the question as a query and download the top ranked webpage by Bing. The passage from this document with the highest BM25 score with respect to the query, is considered the final answer by the baseline approach.

[6] http://webscope.sandbox.yahoo.com

The quality of the generated answers is compared with the human generated best answer from Yahoo! Answers dataset, and is quantified using three text similarity metrics: Jaccard Coefficient, Cosine Similarity, a variant of Kullback–Leibler divergence that is symmetric and incorporates smoothing [13]. Question length can influence the effectiveness of our technique. Extremely short questions, as well as long questions are harder to answer. The former suffers from data sparsity, and the latter from noise. Thus we categorize the questions based on their length and report the results for each category separately.

Question Length (in tokens)	[0;6]	[7;12]	>=13
Number of questions	32	43	25
Baseline			
Jaccard similarity	0.05∓.01	0.06∓.01	0.06∓.01
Cosine similarity	0.08∓.02	0.09∓.02	0.09∓.02
KL divergence	0.50∓.06	0.53∓.05	0.42∓.05
Our System			
Jaccard similarity	0.09∓.03	0.10∓.02	0.10∓.03
Cosine similarity	0.21∓.05	0.20∓.05	0.20∓.06
KL divergence	0.65∓.08	0.75∓.08	0.68∓.10

Table 1. Answer Quality Results

6 Results and Analysis

The results are reported in Table 1. The length in tokens used to categorize the questions is specified at the top of the table, along with the number of questions that get sorted into each category. Next, the average similarity between the answers generated by the baseline approach and the best answers are reported. Finally, the average similarity between answers generated by our approach and the best answers is reported.

The overall trend is that our approach performs consistently better than the baseline across the board. Jaccard coefficient measures fraction of term overlap between the system generated answer and the best answer. The Jaccard low values indicate small term overlap. Often time the reason for this is that the answers generated by the system are not succinct. Our approach extracts a sequence of 5 sentences as the answer. Typically only one of these sentences

contains the actual answer. This observation inspires one of the future directions: identifying the answer boundary for each question. This will also model that for some questions the answer maybe longer than 5 sentences, and for others much shorter than 5 sentences.

The cosine similarity improves over the previous metric by modeling term frequency and by incorporating length normalizing. With this metric, the improvement over the baseline is much larger. The next metric, KLD incorporates language modeling and smoothing, thus providing an even better metric for quantifying similarity between two texts. Again, the average similarity values of our system are substantially higher than those with the baseline. Of the three length categories, the KLD value is highest for mid-range questions, as we expected. We also see that the shorter questions, that is data sparsity, are slightly harder for our system than longer questions.

The main bottleneck in the system is downloading the document from the internet - it is the most time consuming operation, which takes 90% of the time to answer the question. Our system limits the impact of this bottleneck by setting a time cutoff of 2.5 seconds for downloading an individual URL, and a cutoff of 51 seconds for downloading all the URLs necessary for a question. With these constraints the average time to answer a question is 25 seconds. With 99% of the questions being answered within 1 minute.

Table 2 provides a few questions, and the corresponding answers generated by the baseline system, and our system. The selected questions are both factoid and non-factoid. It can be observed, that answers given by our system are more relevant to the question. On the other hand, the answers given by baseline rely only on the frequency of the search terms and generally do not answer the question and could very irrelevant. The answers given by our method are few sentences long even for factoid questions - this could be a potential way of improvement for our system.

Question #1	What is the breed of the dog in doge Internet meme?
Baseline	Reacting to the meme, she explained, "To be honest, some pictures are strange for me, but it's still funny. I'm very impressed with their skills and taste. Around me, nobody knows about the doge meme. Maybe I don't understand memes very well, because I'm living a such an analog life." Satō has also expressed that she had learned that "the risk of the internet is that anyone in the world can see my life on my blog".
Our system	Until doge, the Shiba Inu wasn't even on my radar. I'm only mildly embarrassed to admit that it's become my new favorite meme, eliciting real life excitement whenever I see the breed, which, only a few months prior, was invisible to me.
Question #2	What is the best way to cook fish?

Baseline	How to test fish for doneness. Always check the fish at the minimum baking time. Insert a fork into the fish and gently twist. The fish is done as soon as it begins to flake.
Our system	It's not only delicious, it's also versatile there are a number of healthy ways to prepare it. Here are five cooking techniques that will inspire you to add more fish to your meals . You can grill fish: Grilling, indoors or out, cooks fish fast and keeps it tender, juicy, and flavorful. Season the fish with your choice of fresh or dried herbs or a mix of herbs and spices. With a basting brush, lightly oil the fish with extra- virgin olive or canola oil so it won't stick to the grill or grill pan.
Question #3	When Mozart was born?
Baseline	When Mozart lived, there were no music downloads. No records of any kind, just notes written on paper. People heard music only when it was 'live'. People sang while they worked. They danced at balls and parties.
Our system	He was a composer, and one of the greatest musicians of all time. He was famous as a child because he could play and write music from the age of 4. He grew up to write some of the most beautiful music ever heard. When did he live. Mozart was born in 1756.
Question #4	How to learn skateboarding?
Baseline	You will learn the basic Teaching Model which serves as your class outline from introduction to review, you will learn how to break down each skill into three easy to understand steps and how to clearly communicate those steps to your students. You will learn how to manage your class effectively and safely. You will learn about what makes for a great teacher so you can become one as well. Live sessions will give you a chance to practice teaching each of the steps.
Our system	In a nutshell, no, there is no such thing as skateboarding classes. just pick up a board. get on it, and skate. just learn from your falls, and whatever you do.

Table 2. Sample questions and answers

7 Conclusions

In this paper we presented our first attempt at tackling the challenging problem of answering open-domain non-factoid questions. Many of our system design choices were governed by the track requirement of answering every question under a minute. Our approach starts by translating the question into a boolean query which is then run using the Bing search engine to identify documents that may contain the answer. The last phase extracts candidate answers for each of the downloaded documents, and selects the final answer from this pool.

References

[1] Suryanto, Maggy Anastasia, et al. "Quality-aware collaborative question answering: methods and evaluation." Proceedings of the second ACM international conference on web search and data mining. ACM, 2009.
[2] Lin, Dekang, and Patrick Pantel. "Discovery of inference rules for question-answering." Natural Language Engineering 7.04 (2001): 343-360.
[3] Lee, Gary Geunbae, et al. "SiteQ: Engineering High Performance QA System Using Lexico-Semantic Pattern Matching and Shallow NLP." TREC. 2001.
[4] Cui, Hang, et al. "Question answering passage retrieval using dependency relations." Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2005.
[5] Brill, Eric, et al. "Data-Intensive Question Answering." TREC. Vol. 56. 2001.
[6] Ravichandran, Deepak, and Eduard Hovy. "Learning surface text patterns for a question answering system." Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, 2002.
[7] Ittycheriah, Abraham, et al. "IBM's Statistical Question Answering System." TREC. 2000.
[8] Chinchor, Nancy, and Patricia Robinson. "MUC-7 named entity task definition." Proceedings of the 7th Conference on Message Understanding. 1997.
[9] Bian, Jiang, et al. "Finding the right facts in the crowd: factoid question answering over social media." Proceedings of the 17th international conference on World Wide Web. ACM, 2008.
[10] Zheng, Zhaohui, et al. "A regression framework for learning ranking functions using relative relevance judgments." Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2007.
[11] Moldovan, Dan, et al. "The structure and performance of an open-domain question answering system." Proceedings of the 38th Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2000.
[12] Prager, John, et al. "The use of predictive annotation for question answering in TREC8." Information Retrieval 1.3 (1999): 4.
[13] Kulkarni, Anagha, and Jamie Callan. "Selective search: Efficient and effective search of large textual collections." ACM Transactions on Information Systems (TOIS) 33.4 (2015): 17.