WaterlooClarke: TREC 2016 LiveQA Track

Alexandra Vtyurina
University of Waterloo
avtyurin@uwaterloo.ca

Charles L. A. Clarke
University of Waterloo
claclark@plg.uwaterloo.ca

ABSTRACT

The problem of question answering is becoming increasingly popular. Previous question answering tracks included factoid questions, list questions and complex questions[4]. An effort had been put in modelling users behaviour, but the important constraint that remained was that the questions did not come from real users. The purpose of LiveQA track is to overcome this limitation and to provide participants with real world questions posted by users of a community question answering (CQA) website. Running for the first time in 2015, LiveQA track has attracted people from 14 institutions submitting 22 systems[1].

The source of questions for this track is Yahoo!Answers (Y!A)[1] – a CQA website, where users are invited to post their questions on a variety of topics ranging from gardening to relationship. During the challenge, the newly posted questions from select topics were sent out to every participating system, that in turn would return a comprehensive answer to the question within 60 seconds. The challenge was running for 24 hours starting May 31st. All of the returned answers were afterwards manually judged by NIST assessors on a 5-level Likert scale.

Our approach in this competition was based largely on the one our team submitted last year. We were selecting keywords from submitted questions, and using them as a query to a web search engine. The answer was selected based on keywords frequency and proximity from the top 10 documents returned by the search engine. A difference from the last year's approach is that the final answer was constructed out of top 5 ranked paragraphs, as opposed to a single paragraph last year.

1. INTRODUCTION

The task of automatic question answering has appeared a multiple times in TREC. There can be noted a general tendency of moving from simple questions, like factoid questions, towards more complex questions, like cQA[4]. Although effort had been put in modelling real-users behaviour, the questions were not coming from real users and the corpora for answer selection was often limited.

LiveQA track brought the task to the new level by providing real world questions, unlimited corpora usage and restricting the time in which an answer had to be provided. The questions for this task were harvested from Y!A – a community question answering website. Questions there consist of two fields: title and body. Title is usually used as a short description of the question, and the body allows to provide more details if necessary. After posting a question, the asker waits for another Y!A user to provide an answer. The waiting time depends on the number of users willing to answer the question at this time.

Y!A questions vary greatly across many topics and question types. Y!A users are often seeking other people's opinion, advice about a problem they're having. Some of them just want to share their insights or emotions about newly acquired knowledge or experience. In certain cases people do not have a well defined information need, but they are looking to start a conversation (see Table 1).

The questions were collected from the list of newly posted (and not yet answered by human users) questions on Yahoo! Answers. Each question was sent to every participating system and an answer was expected to arrive within a 60-seconds window. The answer was supposed to contain a text snippet of length less than 1000 characters and the list of resources, from which it was obtained. If the answer was received after 1 minute, it did not count towards the total score of the system. Participants could also choose not to answer any question.

The evaluation of the answers given by participating systems was done by human NIST assessors on a 5-level Likert scale.

2. EXPERIMENTAL SETUP

The experiment was running for the duration of 24 hours starting May 31st. During this period of time the participating systems were supposed to be online. Yahoo! server collected a newly posted question from Yahoo! Answers and broadcasted it to all the registered systems at a rate of approximately 1 question per 60 seconds.

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

`Title:` Is my nose too big? `Body:` Is my nose bad or horrible. I know it's bigger but just how bad is it
`Title:` Whats the meaning of life? `Body:` i wanna know YOUR meaning of life!
`Title:` Emma Stone, Mila Kunis or Penelope Cruz? Who's most beautiful in your opinion? `Body:` I can't decide they are all gorgeousss <3 :)

Table 1: Various types of questions asked on Yahoo! Answers

Every question consisted of 4 fields: *qid* - question identifier, *title* - a question, formulated by a person, *body* - optional detailed description of the question, and finally, *category* - the category that the person chose for their question (if the user skips the step of picking a category, it is defined automatically).

A response to each question was expected upon 60 seconds after sending. It was supposed to contain the following fields: *pid* - participant id (`uwaterlooclarke` was used for this submission), *qid* - question identifier, *answer* - a text of length of max 1000 characters, *sources* - a list of sources where the answer was fetched from, *local time* - locally measured time in ms it took to produce the answer, *explanation* - an optional string containing additional information about the answer. Responses that were received after the 60 seconds were not judged.

3. GENERAL APPROACH

The two main assumptions that we made for this task were first, an answer to virtually any question already exists on the Internet, and second, modern day search engines perform well in retrieving relevant documents given a query – the information sought will likely appear in the top 10 documents. In other words, we reformulate the initial task of finding an answer to a question as follows:

1. given a question, form a descriptive query;
2. submit the query to a search engine;
3. select the best passage from the top 10 documents returned.

One of the main challenges in the described approach is the task of forming a descriptive query. As shown in Figure 1, there are questions that contain irrelevant details, making it hard to understand the primary intent of the asker. The method we used for selecting query words relies on pointwise KullbackâĂŞLeibler divergence score. After selecting query words, we concatenated them and submitted the resulting query to Bing! Search and examined the top 10 documents returned. From them we extracted a set of candidate answers and ranked them based on their length, keyword frequency and proximity as described in[3].

3.1 Query selection

Question posted on Y!A vary not only in their topicality and intent, but also in their length and details provided. Some

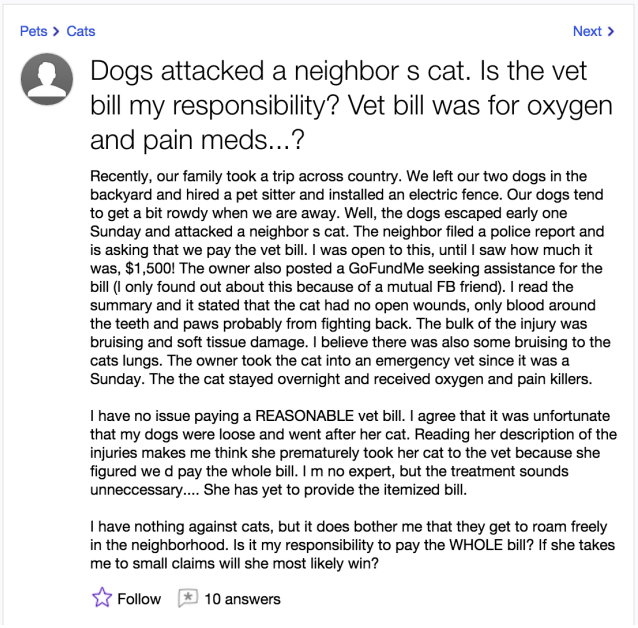

Figure 1: An example of a long question, containing a lot of detailed information

of the questions, for example Figure 1 contain much details that are unnecessary for answering the question. But the biggest challenge in trying to answer such questions is that the real intent of the asker is hard to uncover due to the noise created by the addition of extra words to the question. In this work we implement a baseline for keyword selection: we rank all question words by the pointwise KL-divergence score and select the ones that scored the highest.

In order to use KL-divergence for query term extraction we needed to have a background language model. Given that the language used in online user-generated content differs significantly from formal English [2], we needed to have an example of the language used on Y!A. We crawled Yahoo! Answers to collect a dataset of questions and answers from all categories in order to see what type of language is used[2][3]. For each question thread we collected question title, question body, and answers (if any), posted by other people. We removed web links from the obtained text and used the rest of the text to build a language model for background probability computation.

3.2 Answer extraction

For every question received we combined its *title* and *body* together and removed the links from the resulting text. We compared the words distributions in the question text and the previously constructed language model and picked the words with the greater divergence value, which means that these words distinct the given question from the common language. For every word in the text a corresponding KL-divergence[3] value was computed, using the Yahoo! Answers language model constructed earlier. Afterwards, the

[2]All code used for this task is available at:
https://github.com/sashavtyurina/LiveQATrack
[3]https://github.com/yuvalpinter/LiveQAServerDemo

words were sorted based on their corresponding KLD score. We also used NLTK[4] to extract named entities from the question text. These named entities as well as the 4 words with the highest KLD score were put together in the order of their occurrence in the initial question to form a resulting query.

The query was submitted to the Bing Search API[5] and the top 10 returned documents were retrieved. We ignored pages from Yahoo! Answers, as well as all non-html pages (for example, pdf, doc, etc). For every web-page we allowed a 5 seconds time limit to load, otherwise it was ignored. We used this set of web documents as a corpus to extract an answer to the given question from.

After the web pages were retrieved, they underwent a preprocessing step, during which only useful text was extracted from each of them. First, we removed the contents of a predefined list of tags (that are highly unlikely to contain the useful text that we are after): style, script, table, label, title, etc. From the remaining portion of the page the tags, with contents of less than 10 words are removed. By doing this we excluded ads, "follow us" links, and other irrelevant information.

After the preprocessing every web page becomes was transformed into a clean text document. At this step we inserted a pair of special symbols used to denote the beginning and the end of each sentence. This was done in order to produce more readable results in the future. For every document we found a set of m-covers (passages containing keywords), using the terms from the query we previously submitted to Bing. If the length of a passage was greater than the given limit (1000 characters), it was discarded. The remaining passages were ranked according to the number of query terms they contained and their proximity to each other within the passage[3]. After the passages were scored, at most the top 5 passages were concatenated by a newline character and submitted as the final answer. The URLs, corresponding to the documents from which the top 5 paragraphs were retrieved were passed along as the resources of the answer. If no passages passed the filtering, the answer returned was "Try these links:" followed by the top 5 links returned by Bing! Search.

4. CODE BASE

The code for this project was written in Python 3.5. The server for communication with Y!A server was built with Bottle framework[6] for Python. Answer extraction module was implemented in Python with the use of NumPy library[7]. We also used NLTK for named entity extraction.

5. FUTURE WORK

We would like to improve the procedure of finding an answer to a given question by analysing existing human-generated question-answer pairs. We are hopeful that finding the ways in which an answer is related to the question will help extract more precise answers in the future.

It is not uncommon for community question answering services to have an exceedingly long question descriptions. People often want to see an advice that is unique for their situation (see figure 1). Redundant details often obstruct question focus, making it hard even for a human to understand. We have started making first steps towards extracting keywords from long questions.

6. CONCLUSIONS

The LiveQA track provides an opportunity for the participants to try their QA systems on real-world questions, collected from Y!A. The approach we chose is mostly similar to that of last year's. It is based on selecting key terms from a question, submitting them to a search engine and extracting an answer from the top 10 retrieved documents.

7. REFERENCES

[1] E. Agichtein, D. Carmel, D. Harman, D. Pelleg, and Y. Pinter. Overview of the trec 2015 liveqa track. In *The Twenty-Fourth Text REtrieval Conference (TREC 2015) Proceedings. National Institute of Standards and Technology (NIST)*, 2015.

[2] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 183–194. ACM, 2008.

[3] S. Büttcher, C. L. Clarke, and G. V. Cormack. *Information retrieval: Implementing and evaluating search engines*. Mit Press, 2010.

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

[4]http://nltk.org/
[5]https://datamarket.azure.com/dataset/bing/search
[6]https://bottlepy.org/docs/dev/
[7]http://www.numpy.org/