Complex questions: Let me Google it for you

Alexandra Vtyurina
School of Computer Science
University of Waterloo, Canada
avtyurin@uwaterloo.ca

Charles L. A. Clarke
School of Computer Science
University of Waterloo, Canada
claclark@plg.uwaterloo.ca

ABSTRACT

Many questions submitted to community question answering (CQA) sites such as Yahoo!Answers can be reasonably answered by a simple query to a standard Web search engine. However, the user may not know how to reduce their question, which may be several paragraphs in length, down to the 2-4 terms that will return an answer. The work in this short paper is directed towards the creation of a system that can do this reduction for them. More specifically, in this paper we focus on automatically generating training data for this purpose. Given existing question-answer pairs, we determine the combination of terms appearing in the question that best returns the answer. Given the large collection of question-answer pairs that already exist on CQA sites, we hope to generate large numbers of training examples. Using this training data, we hope to learn to extract short queries from long questions that have no existing answers. Although the results in this workshop paper are preliminary in nature, they demonstrate the potential of this approach.

1. INTRODUCTION

Most personal assistant systems, such as Apple's Siri and Microsoft's Cortana are equipped with a mechanism that will quickly give answers to questions like, "Where was Abraham Lincoln born?" or "How tall is Beyoncé?". Similarly, Google's knowledge graph provides a convenient interface that answers factual questions, so that a user does not have to browse through multiple web pages searching for the answer. Such factoid questions are usually quite short and have a single correct answer, making it reasonably straightforward for these systems to return this answer. However, not every question has a simple answer, and people often want to obtain an answer to more complex questions. While the advances in factoid question answering are significant, research in answering more complex questions is still at its early stages.

The exact definition of a "complex question" is somewhat vague and largely depends on the source of the questions. Nonetheless, there are some features of the complex questions that are frequently encountered. They are usually verbose, have multiple correct answers, and a complex syntactic structure. They might include opinion seeking questions, questions that can be answered based on someone else's prior experience, questions that require speculation and logical conclusions, or questions that include a variety of small details that a person sees as meaningful. They are usually open domain, i.e., they are not restricted to a particular topic. Another important feature of such questions is their free form, which entails syntactic and grammatical errors and ambiguity. Questions that contain some of the above features are not yet reliably answered by automated systems, and researchers in the field of CQA are targeting the problem of finding answers to them automatically.

Websites like Yahoo!Answers, Quora and StackExchange provide a forum-like interface for question answering. Individuals post questions on these sites, and the community of users attempts to answer them. Questions posted on these websites are often non-factoid, and may be quite long, sometimes several paragraphs in length, with separate fields for a title and a body. When the answers are posted, the best one can be chosen by the asker, and any given answer can be upvoted or downvoted, providing a natural ranking system.

While this approach may provide a convenient way to answer complex questions, many questions go unanswered. For example, according to Li and King[6] only 17.6% of questions posted on Yahoo!Answers receive a satisfactory answer within 48 hours, while 16% end up with no answers at all. Having a system that would be capable of answering complex questions automatically would mitigate this problem and provide a higher level of user experience.

Current research is putting forward ways to answer such questions automatically with little waiting time. At TREC 2015, for the first time, a LiveQA track was conducted, where participating QA systems were required to answer free-form, real-world questions from Yahoo!Answers. At the end of the track, the best performing system demonstrated that it could produce sensible answers to more than half of the questions [1].

The modern Web covers the majority of topics of interest and keeps evolving day by day. We believe that for many complex questions an answer can be found on the Web, which can be reached through a regular commercial search by a single query. In other words, the answer to many complex questions is already out there, if only the user knew the query to reach it.

Modern search engines do a good job in retrieving docu-

© 2016 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

ments based on short queries. Bailey et al.[2] report that the most popular queries for a search engines are those of lengths 2 and 3 words. Unsurprisingly, submitting a full complex question to a standard web search engine returns poor results, since an average question length is 14.5 meaningful words (based on the questions from the Yahoo!Answers dataset available on Webscope[1]). However, for many of these questions human reader can construct a 2-4 words query that returns a set of relevant documents, which are likely to contain an answer to the given question.

Let us consider an example. Figure 1 shows a question that was posted on Yahoo!Answers, and the answer that was chosen by the asker as the best. While supplying the entire question text to a search engine does not yield the desired results, a short query "shy timid gathering", constructed exclusively from the words appearing in the question, retrieves a set of relevant documents, from which an answer to the question could be further selected. Alternatively, these web results could be presented to the asker directly, and she could discover the answer herself. If we could automatically shrink a long question to a 2-4 words query describing the issue, we would be half-way towards answering complex questions. However this task is not trivial.

In order to learn to extract descriptive words from arbitrary questions, we would need a large amount of training data – a set of question-query pairs, that have already been reliably labeled. Labeling questions by hand is a costly and time consuming problem, and ideally we would like to come up with a way to do so automatically with little or no human involvement. Although it is expensive to manually create queries for complex questions, the big advantage is that there is an abundance of questions that have already been answered by human users, which have been evaluated for quality by the ranking/voting process.

Therefore, we possess a large dataset of questions and their answers that we would like to use to automatically produce a training set of question-query pairs. Our plan is: first, to learn to identify question keywords based on the question and its answers, second, to automatically create a large training set of question-query pairs, and finally, to use this training set to learn to extract keywords from new questions, that have not yet been answered.

In this paper, we describe work in progress on a method for automatically creating a short 2-4 words query for a given question and its existing answers. In our method, we first rank question words by their pointwise K-L divergence score, then we use top words from the list to create a pool of possible queries. For each of the constructed queries we retrieve top 10 snippets from a search engine, that we afterwards use to evaluate the "goodness" of the query for the given question. In our work we assume that the answers given by humans are exemplary.

2. PRELIMINARY EXPERIMENTS

Our goal is to devise a method of automatically creating a query, describing question's intent, given the question and its answers. To evaluate the proposed approach, one of the authors manually labeled a small set of questions with short queries, describing each question's topic perfectly.

The proposed approach is as follows:

[1]L6 dataset
http://webscope.sandbox.yahoo.com/catalog.php?datatype=l

Title i think i have a problem. i feel shy when i arrive at a gathering and say hello to people or guests.?
Body i also feel the same when i leave. i feel like just leaving without saying goodbye cause i feel timid.? what should i do to overcome this?
Best answer HI friend! Dont feel like that. U can overcome this simple problem. Do one thing that go and stand infront of the mirror and practice before talking with others. Do this u can overcome this problem. bye

---

Figure 1: Example question from Yahoo!Answers

1. create a pool of probe queries;
2. retrieve a set of 10 snippets for each of the probe queries, and for the manually created query;
3. calculate a similarity score between retrieved snippets and the question and its answers;
4. rank queries based on the similarity score of the corresponding snippets. It is our goal to have high quality queries ranked in the top.

Since we know that the manually created queries are near perfect, we expect them to be ranked high. We evaluate the described approach by examining the rank of the manually created queries, and show that it is beneficial compared to the baseline.

2.1 Dataset

We ran our experiment on a small subset of Yahoo!Answers dataset, that contains a number of questions with human generated answers, as well as their attributes, such as language, category, etc. In our work we will only be using questions and their answers. Each question consists of two fields – title and body. The contents of both fields have no restrictions. We extracted 82 questions from the WebScope dataset to run our experiment. The questions we chose had to be in English, with the body length not less than 150 characters. We applied no restrictions to the answers.

2.2 Labeling and preprocessing

We first labeled each question, assigning it a query, manually created by one of the authors. The requirement for a query was that it should return relevant results (as judged by the query creator), submitted to one of the search engines. Each query could only contain words that appeared in the initial question with no limitation to the length or order of the words. We will further deem such queries as the perfect ones for their corresponding questions.

Due to a noisy nature of human created content, we had to put questions and their answers through a preprocessing step first. To begin with, we shrank all characters repeating more than 3 times in a row with a single instance to eliminate possible emotional occurrences like "heeeeelp!!!!". We also removed stop words and words that were less than 3 characters long. Finally, we applied s-stemmer[5] to discard any plurals from the text, and substituted all URL links with their main domain names. After this procedure, questions and answers turned into sets of tokens.

2.3 Constructing probe queries

Our goal is to learn to identify question words that will create a good (producing relevant results given to a search engine) query. We would like to construct a pool of all possible queries created of question words, and rank them according to the results they retrieve. But due to a big length of the questions, the number of all possible combinations would grow exponentially and the experiment would become very expensive. Hence we decided to only consider 10 keywords for every question and use them for query construction.

For keywords selection we calculate pointwise K-L divergence[8] for each question word, using the ClueWeb90b corpus[2] for our background language model:

$$score_t = p_t \cdot log(p_t/q_t), \quad (1)$$

where p_t is a relative frequency of term t in the question text, and q_t is a relative frequency of term t in the background collection – ClueWeb09b. We then selected 10 words with the highest score. We have noted that oftentimes question title would contain many of the words descriptive to the question topic. We wanted to put an emphasis on these words and therefore doubled the number of title words in the question text. This heuristic was partially inspired by a work submitted on LiveQA 2015 track, where only title words were used as a query[3]. After the selection process, we arranged the resulting keywords in the same order they appeared in the initial question.

Out of the 10 keywords we constructed combinations of 3 words without repetitions, that constituted our probe pool for this question. There could be up to 120 queries for each question (less, if the question length after preprocessing was less than 10 words). To this pool we also added previously created ground truth queries.

2.4 Probe query ranking

To evaluate the goodness of each of the probes, we have submitted each of them to Bing and, using Bing search API[3], collected 10 snippets and their corresponding URLs that were returned in response. Sometimes a page where the question was initially posted, or pages with the question duplicate would be returned. We ran the snippets through a filter to remove such occurrences. We discarded a snippet if its URL contained a string "answers.yahoo" and the question's yahoo qid – a question's unique identifier. We also would remove a snippet, if it looked too similar to the question – more than half of the snippet's words appear in the question. We will further use the collected snippets to evaluate queries. The reason we chose not to use full documents for our task is additional noise that comes in from processing full web pages. Also, we will be comparing the snippets with answers and questions, that are more alike in their nature with the snippets rather than with full web pages.

Our goal is to find a query that would return a list of relevant snippets for a given question. Therefore, we would like to use a few question words that retrieve snippets similar to the answers. However, oftentimes, similarity with the answer alone may not be enough. For example, in figure 2 similarity of snippets with the answers will not at all guarantee that the snippets will be relevant to the question topic. Therefore, we also included similarity with the initial question in our metric. Overall, we would like our top ranked query to return snippets that look both like question and their answers.

[2]http://lemurproject.org/clueweb09/

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

Title: How can you ?
Body: who will approve the names of the newest elements? Wil it be A) the scientist that discover each element B) a committee scientist C) the chemist's from a research insitution
Answer 1: A
Answer 2: I agree, A

Figure 2: Question and answers do not intersect

At this point we have a set of questions, their answers, manually created ground truth queries, a list of probes constructed of question words and snippets retrieved with the probes as will as with the ground truth query. Now we will rank the probes based on their corresponding snippets and their similarity with the question and the answers.

To eliminate possible noise we represent question as 20 top terms ranked by pointwise K-L divergence same way as shown in (1). We will denote this set as Q. We would also like to shorten answers text to a set of 20 terms. We decided to take advantage of having multiple answers for every question and make use of possible redundancy. Following Clarke et al.[4] we substituted p_t in formula (1) with a relative term frequency between answers, i.e. $p_t = f_t/num_{ans}$, where f_t is the number of distinct answers the term appears in, and num_{ans} is the total number of answers given to this question. We will denote a set of terms extracted from the answers as A.

The similarity metric that we chose was proposed by Tan and Clarke [7] and is based on the number of terms overlapping between snippets and keywords extracted from question/answer. For every probe query Pr we calculate the following:

- average fraction of snippet words, probe query terms excluded, overlapping with question keywords:

$$AvgQ = \frac{1}{|S|}\Sigma|(S_i \setminus Pr) \cap Q|; \quad (2)$$

- average fraction of snippet words, probe query terms excluded, overlapping with answers keywords:

$$AvgA = \frac{1}{|S|}\Sigma|(S_i \setminus Pr) \cap A|; \quad (3)$$

- fraction of all words of all snippets, probe query terms excluded, overlapping with question keywords:

$$TotalQ = \frac{1}{|\cup S_i|}|(\cup S_i \setminus Pr) \cap Q|; \quad (4)$$

- fraction of all words of all snippets, probe query terms excluded, overlapping with question keywords:

$$TotalA = \frac{1}{|\cup S_i|}|(\cup S_i \setminus Pr) \cap A|, \quad (5)$$

where Q is a set of keywords extracted from the question, A is a set of keywords extracted from answers, S_i is a set of words from a single snippet, $|S|$ is a total number of snippets for this probe query, Pr is a set of probe query terms.

	KLD	EqualWeights	BestWeights
R@3	**0.407**	0.356	**0.407**
R@5	0.535	0.544	**0.614**
R@7	0.633	0.689	**0.761**

Table 1: Results of the experiment show slight improvements compared to the baseline.

We calculated the final score of the probe query using a linear combination of the above terms:

$$Score_{Pr} = \alpha AvgQ + \beta AvgA + \gamma TotalQ + \delta TotalA, \quad (6)$$

where we varied parameters $\alpha, \beta, \gamma, \delta$ for our experiments. We ran the ranking with all parameters being equal to 0.25, and denoted this run as $EqualWeights$.

We have also noted that due to a variety of questions and answers and their quality, choosing different weights may be beneficial. We ran parameter sweep for $\alpha, \beta, \gamma, \delta$, varying each of them in the range [0..1] with a step of 0.1, so that they add up to 1.0. The best weights for each question were the ones that gave the ground truth query the highest score.

We found that for different questions the best weights turned out to be very different and we couldn't find a pattern, according to which they changed. Learning a dependency between question-answers features and the weights that suit them the most is one of the future directions for this research. We also performed ranking using best weights for every question-answers tuple and called it $BestWeights$.

We evaluated the results of each run using $Recall@M$, where we looked at the number of ground truth query words returned using different ranking methods. We used question terms ranked with pointwise K-L divergence as our baseline. We denoted the baseline run as KLD.

Table 1 shows that although the baseline performs well, reranking the terms using snippets is superior.

3. DISCUSSION AND FUTURE WORK

We have conducted an experiment on identifying question keywords, that well describe the question topic. We have encountered a significant variety in the questions and their answers. The differences included question types – looking for specific information, or seeking advice, as well as text quality – usage of slang words, typos, abbreviations. All of it made it difficult to find a general solution that would work well, however, the results of parameter sweep showed that there could be found a solution suitable for a particular question. As a future work for this topic we would like to add more manual labeling to the questions, that would include text quality and asker's intent. We are hoping to find clusters of questions that require particular weights distribution.

We would also like to improve ground truth query composing, by asking several people to compose queries for the same questions, thus, eliminating bias that a single person's judgment may bring. Another direction of the future work may include filtering out answers with low text quality. As using all answers in this work may have introduces additional difficulty in ranking the probes. Finally we plan to use crowdsourcing to collect a large dataset of question-query pairs and compare them with the results we will get from automatic labeling.

4. CONCLUSION

Many questions posted on community question answering websites, such as Yahoo!Answers and StackExcahnge, require a long wait time to receive answers, or remain unanswered altogether. Such questions are usually verbose and have a complex intent and existing methods are not always applicable to them. We would like to tackle this problem and design an approach that would allow to answer such questions automatically with no human involvement. We believe that to nearly any question asked there is an answer somewhere on the Web, and we only need to retrieve it. This paper describes work in progress on identifying question words that describe the question intent and can be used as a search query to retrieve relevant documents. We use a small manually labeled dataset of questions and their existing answers proposed by human users to find out which question words make up the best query for a given question. We then plan to extend this work and to automatically create a large set of questions and their queries that can be used in further research.

5. REFERENCES

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

[2] P. Bailey, R. W. White, H. Liu, and G. Kumaran. Mining historic query trails to label long and rare search engine queries. *ACM Transactions on the Web (TWEB)*, 4(4):15, 2010.

[3] D. Bogdanova, D. Ganguly, J. Foster, and A. H. Vahid. Adapt. dcu at trec liveqa: A sentence retrieval based approach to live question answering.

[4] C. L. Clarke, G. V. Cormack, and T. R. Lynam. Exploiting redundancy in question answering. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 358–365. ACM, 2001.

[5] D. Harman. How effective is suffixing? *Journal of the American Society for Information Science*, 42(1):7, 1991.

[6] B. Li and I. King. Routing questions to appropriate answerers in community question answering services. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1585–1588. ACM, 2010.

[7] L. Tan and C. L. Clarke. Succinct queries for linking and tracking news in social media. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1883–1886. ACM, 2014.

[8] T. Tomokiyo and M. Hurst. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment-Volume 18*, pages 33–40. Association for Computational Linguistics, 2003.