MSR-TR-2015-20
Updated Version – August 28, 2015

Web-based Question Answering: Revisiting AskMSR

Chen-Tse Tsai[*]
University of Illinois
Urbana, IL 61801, USA
ctsai12@illinois.edu

Wen-tau Yih **Christopher J.C. Burges**
Microsoft Research
Redmond, WA 98052, USA
{scottyih,cburges}@microsoft.com

Abstract

Web-based QA, pioneered by Kwok et al. (2001), successfully demonstrated the power of Web redundancy. Early WebQA systems, such as AskMSR (Brill et al., 2001), rely on various kinds of rewriting and pattern-generation methods for identifying answer paragraphs and for extracting answers. In this paper, we conducted an experimental study to examine the impact of the advance of search engine technologies and the growth of the Web, to such Web-QA approaches. When applying AskMSR to a new question answering dataset created using historical TREC-QA questions, we find that the key step, query pattern generation is no longer required but instead, deeper NLP analysis on questions and snippets remains critical. Based on this observation, we propose a Web-QA system that removes the query pattern generation step and improves answer candidate generation and ranking steps, and outperforms AskMSR by 34 points of MRR. The revised dataset is made publicly available with this paper.

1 Introduction

In this paper, we consider the problem of open-domain factoid question answering, where the answers are usually short phrases or few words. For instance,

Question: *Where did Bill Gates go to college?*
Answer: *Havard University*

This form of question answering had been the focus of NIST TREC-QA tracks (Voorhees and Tice, 2000), which were held each year from 1999 to 2007. Due to the complexity of possible formulations of the same question and the variety of ways a correct answer can be presented in documents, deep natural language understanding and reasoning was generally viewed as needed for building a good QA system. Contrary to this belief, one of the top performing systems in TREC 2001, AskMSR (Brill et al., 2001), was built using very simple rules by leveraging the Web documents. More specifically, it tries to generate variations of the given question as textual patterns that can occur in the documents, along with answers. For example, one of the patterns of the above question is “Bill Gates went to college at”. Then, the system queries a search engine by issuing this pattern with quotes as the query. By assuming that the answer would appear near the pattern in the document, the system then tries to extract the answer from the top retrieved snippets (summary of documents).

Although this *question pattern generation* process was viewed as the key step in Web-QA systems like AskMSR and had later been studied extensively (Yang et al., 2003; Zhao et al., 2007), its effect has become less clear nowadays due to two reasons. First, because of the popularity of community QA sites like Yahoo! Answers, the original questions have been included with the answers. Using the original question as query could better match the relevant documents instead of using artificial question patterns. Second, given the advance of modern search engine technologies, especially on general *query reformulation* using query and click logs analysis (Huang and Efthimiadis, 2009), the existing question pattern generation process could be redundant and less optimal.

In this work, we revisit the design of Web-QA systems and examine the effectiveness of individual components of a representative system, AskMSR. We show that a Web-based QA system without question pattern generation can perform significantly better. By removing the pattern generation step, the recall of answer candidates increases without sacrificing precision. In addition,

[*] Work conducted while interning at Microsoft Research.

we improve the answer candidate generation and ranking components. In AskMSR, the answer candidates are extracted based on the n-grams in the snippets. We propose a number of simple rules to categorize questions and incorporate a Named Entity Recognizer (NER) to extract answer candidates which match the target answer type. Instead of ranking answer candidates only by the number of occurrences in the snippets, we study several features that measure the relatedness of a question to an answer candidate.

Our evaluation is done using five years of TREC factoid question answering datasets. As many answers are outdated for a Web-QA system, in order to have an accurate and fair comparison, we update the answers by crowdsourcing using Amazon Mechanical Turk. The revised dataset is released with this paper. We believe it will provide future researchers a valuable benchmark dataset for evaluating factoid question answering systems.

2 Approach

There are three steps in our system framework. Given a question, the first step is to categorize questions based on the type of answers it is looking for. We then issue the question as the query to a search engine and extract answer candidates from the top retrieved snippets. Finally, a trained ranking model which uses information from Wikipedia and WordNet is applied to rank the answer candidates. We describe each algorithmic component in the following sections

2.1 Question Classification

Classifying questions into several pre-defined target answer types has been shown as an important step for locating answer candidates accurately (Hovy et al., 2001; Li and Roth, 2002). We use a few simple rules to classify questions into 13 categories. The categories are based on the named entity types defined an NER[1], which is applied in the answer candidate generation step. The full list of 13 categories and the corresponding classification rules are listed in Table 1.

Type	Rules
continent	"what/which continent"
county	"what/which county"
city	"what/which city"
state	"what/which state"
airport	"what/which airport"
country	"what/which country"
location	starts with "where", contains "birthplace"
person	starts with "who"
date	"what date/day/year", contains "birthday" starts with "when" or "in what year"
digit	"how much/many", "how old"
company	"what/which company"
name	contains "name"
other	all the other questions

Table 1: 13 question types and the corresponding classification rules.

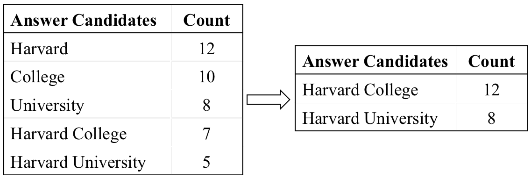
2.2 Answer Candidate Extraction

Given a question, instead of rewriting it by several hand-crafted rules as AskMSR does, we directly submit the whole question as a query to a search engine, and collect the top 40 returned snippets. The next step is to extract answer candidates from these snippets. If the given question is not the "other" type, we apply an NER on the snippets to extract entities with the target answer type of the question as answer candidates. If the question is not classified as any named entity type, we consider all unigrams, bigrams, and trigrams in the snippets as answer candidates.

Filtering One problem of using n-grams is that many of them are not meaningful phrases. To improve the quality of the answer candidate set, we remove n-grams that satisfy any of the following conditions: (1) containing a verb, (2) is a punctuation as a single token, (3) starting or ending with a stop word, and (4) consisting of words in the question. In addition, for questions asking for a specific entity, we require that an answer candidate needs to be a title in Wikipedia if the candidate does not contain any digit. Applying this filtering rule largely increases the precision of answer candidate extraction.

Tiling Even when each answer candidate is a meaningful phrase, there might be duplicate information among all candidates. For instance, "Harvard" and "Harvard University" are both candidates and "Harvard" should have counts no less than "Harvard University" does due to the nature of n-gram. Although both of them could be the correct answer, we want to reduce the redundant information and provide the most complete answers. To resolve this issue, we apply the n-gram tiling algorithm used in AskMSR. The algorithm constructs longer n-grams from a sequence of overlapping shorter n-grams. For example, the

[1] We use an in-house named entity recognizer, which consists of 16 named entity types.



Answer Candidates	Count
Harvard	12
College	10
University	8
Harvard College	7
Harvard University	5

Answer Candidates	Count
Harvard College	12
Harvard University	8

Figure 1: An example of answer candidates tiling. The right list is the result of performing answer candidates tiling on the left list.

list in Figure 1 (left) shows answer candidates before performing tiling. After tiling, only the longer answer candidates are kept (the right list).

2.3 Answer Candidate Ranking

Unlike the original AskMSR approach, where answer candidates are ranked by the frequency counts, we learn a binary classifier instead. The classifier relies on two vector-space models to capture semantic relatedness between the question and answers, as well as other simple features.

WordNet Noun Beginners Matching In this model, we use the 25 unique noun *beginners* in WordNet to represent questions and answers. These beginners are the top words of the WordNet noun taxonomy, each of which corresponds to a distinct semantic domain. We create a 25-dimensional vector for each question and answer candidate by extracting nouns and looking up their beginners. For a question vector, the i-th component of the vector is the total number of i-th beginner appearing in the question. We also increase the counts by one for the first and last beginner in the question since they may carry more information. For an answer candidate, we create the vector using the count of beginners of the words in it, but also take the named entity types into account. In particular, we use the named entity type provided to generate the corresponding beginner. For example, if the named entity type is “Person”, we add the beginner which represents the “Person” sense into the vector.

Wikipedia Introduction Matching This model measures the textual similarity between a question and an answer candidate. Since an answer candidate only consists of few words, we use the introduction section in the corresponding Wikipedia page to enrich textual information. Note that we simply use the Wikipedia page where the title exactly matches the answer candidate. In other words, no entity disambiguation is performed. If an answer string is ambiguous, there will be no match since titles in Wikipedia have some additional description to distinguish an ambiguous concept (e.g,. *Insurgent (novel)* vs. *Insurgent (film)*). Each question and answer candidate is represented by a bag-of-word vector. The i-th component of an vector indicates the term frequency of the i-th word in the vocabulary.

For both vector space models, we take the inner product between the vectors of a question and an answer as the score, indicating how relevant the answer candidate is to the question. We further combine these two scores with other features via training a binary classifier. For a pair of question and answer candidate, we add the following sparse features: words in the question and the answer candidate, named entity types in the question and the answer candidate, total number of occurrences of the answer candidate in the retrieved snippets, and the initial ranking (by the number of occurrences in snippets). Note that words are lowercased and lemmatized in all the features.

3 Evaluation

We first describe how we construct the new dataset by revising answers from five years of TREC shared tasks, and then we compare each proposed algorithmic component with the AskMSR system.

3.1 Dataset

The Text REtrieval Conference (TREC) held a question answering track (Voorhees and Tice, 2000) each year from 1999 to 2007. The data used in the competitions are publicly available and have been a popular benchmark for evaluating various QA systems. We take the factoid QA collections from 1999 to 2003 (TREC 8-12) to develop our system. Excluding the questions which were left out from the evaluation in the shared tasks, there are 2,131 questions in total. We use the 1,751 questions from 1999 to 2001 for training and the 380 questions from 2002 for testing.

The systems participated in the shared tasks were automatically evaluated against answer patterns which consist of regular expressions to capture possible answer strings. Although these patterns are also publicly available, they became inaccurate several years after the data was released. This could be due to two reasons. First, in the shared tasks, participants are asked to extract an-

#	Approach	MRR
1	AskMSR[2]	27.22
2	Direct Query	45.24
3	Direct Query + Question Type	57.27
4	WordNet Beginners Matching	57.33
5	Wikipeida Introduction Matching	61.06
6	AskMSR+	62.11

Table 2: An ablation study of how each proposed component improves over the original AskMSR.

swers from a collection of documents. Some answers may not occur in the documents but exist in the snippets retrieved by a search engine. In addition, the patterns may only be able to capture the wording used in the documents, and thus fail to cover other possible expressions of the answers. Second, answers could change over time, such as those to the questions asking about the population of a city and the current president of a country.

We use Amazon Mechanical Turk to collect the answers of these 2,131 questions. Along with the question, we provide the top 40 snippets retrieved by Bing to the Turkers and ask them to extract an answer from each snippet. If a snippet does not contain any answer, "None" should be provided as the answer. Each snippet is examined by three Turkers and the majority of the three answers is considered as an answer pattern. We manually resolve the cases where three answers are all different.

3.2 Experimental Results

In our experiments, we use Bing as the search engine to query relevant snippets and an SVM model with polynomial kernel of degree 3 is used as the binary classifier. The decision value obtained from the model is used as the relevance score to rank answer candidates. Following the evaluation method used in TREC, we take the Mean Reciprocal Rank (MRR) of the top 5 answer candidates as our main evaluation metric: $\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{r_i}$, where N is the total number of questions, and r_i is the highest ranking position of a correct answer to the i-th question. If no correct answer found in the list of answers, $\frac{1}{r_i}$ is set to 0.

Table 2 shows a comparison between the original AskMSR system and each proposed algorithmic component. The main difference between AskMSR and Direct Query is that Direct Query issues the question string to a search engine directly and considers all n-grams from the top retrieved snippets, whereas AskMSR queries a search engine by several reformulations of the original question in order to mine the n-grams around the query strings. Note that the answer candidates are ranked by the number of occurrences for both approaches. We can see that Direct Query already outperforms AskMSR significantly without applying other proposed methods.

In the third approach, we add the proposed simple question classification method, as well as the candidate generation approach that extracts the named entities with the corresponding target answer type from the snippets. The 12-point improvement over Direct Query implies that question typing is a very important step which poses semantic constraints on possible answer candidates, and thus prunes many irrelevant answer candidates.

The last three approaches re-rank the answer candidates of the third approach by the proposed WordNet Noun Beginners Matching, Wikipedia Introduction Matching, and the final combination by training a binary classifier respectively. We can see that Wikipedia Introduction Matching performs very well when comparing to ranking by counts, which indicates that it is useful to enrich contextual information of answer candidates from other knowledge sources. Finally, AskMSR+ is the full proposed system, which combines the two vector-space models with other semantic and syntactic features extracted from the snippets and the question. By learning from the training data, a binary classifier can achieve the best ranking result on the test set. It would be interesting to investigate other learning to rank algorithms.

4 Conclusions

In this paper, we revisit the design of Web-QA systems and propose an enhanced framework. By removing the question pattern generation process and by adding semantic matching features between questions and answers our system achieves a substantial gain on TREC-QA questions. We hope this study can provide an updated view of Web-based question answering and with the new QA dataset, enable more future research on this important problem.

[2] We follow the implementation from Cucerzan and Agichtein (2005) which added more query reformulation patterns to Brill et al. (2001), and change the underlying search engine to Bing.

References

Eric Brill, Jimmy J Lin, Michele Banko, Susan T Dumais, and Andrew Y Ng. 2001. Data-intensive question answering. In *TREC*.

Silviu Cucerzan and Eugene Agichtein. 2005. Factoid question answering over unstructured and structured web content. In *TREC*, volume 72, page 90.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the first international conference on Human language technology research*, pages 1–7. Association for Computational Linguistics.

Jeff Huang and Efthimis N Efthimiadis. 2009. Analyzing and evaluating query reformulation strategies in web search logs. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 77–86. ACM.

Cody Kwok, Oren Etzioni, and Daniel S Weld. 2001. Scaling question answering to the web. *ACM Transactions on Information Systems (TOIS)*, 19(3):242–262.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207. ACM.

Hui Yang, Tat-Seng Chua, Shuguang Wang, and Chun-Keat Koh. 2003. Structured use of external knowledge for event-based open domain question answering. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 33–40. ACM.

Shiqi Zhao, Ming Zhou, and Ting Liu. 2007. Learning question paraphrases for QA from Encarta logs. In *IJCAI*, pages 1795–1801.