FOCUSED QUESTION ANSWER EXTRACTION FROM Q/A RICH WEBSITES

David Urbansky
University of Technology Dresden, Germany
david.urbansky@tu-dresden.de

Daniel Schuster
University of Technology Dresden, Germany
daniel.schuster@tu-dresden.de

Maximilian Walther
University of Technology Dresden, Germany
maximilian.walther@tu-dresden.de

Alexander Schill
University of Technology Dresden, Germany
alexander.schill@tu-dresden.de

ABSTRACT

This paper presents a focused crawling approach to question answer extraction from Q/A rich websites. Recently, websites with user generated questions and answers have become very popular and form a source of valuable information. We present a Q/A extraction system that can collect question answer tuples from these websites and aggregate them to a knowledge base that can be used for question answering. We show that using a focused crawling approach works well in the Q/A domain and outperforms the trivial crawling approach regarding time and memory efficiency.

KEYWORDS

question answer extraction, machine learning, focused crawling.

1. INTRODUCTION

Recently, question/answer websites such as Wiki.Answers.com, Answers.Yahoo.com, ChaCha.com or StackOverflow.com have become increasingly popular. On these Q/A rich sites, users can ask almost any kind of question which is then answered by other users. The sum of all question/answer tuples available on these sites, build an enormous real-world corpus of knowledge that can be used in Q/A research and question answer applications.

Our contributions in this paper are as follows. (1) We describe WebKnox Q/A, a system for question answer extraction from several Q/A rich websites, (2) we demonstrate that only little configuration is necessary to make the system effective and (3) we show that a focused crawling approach increases the efficiency (time and memory usage) of finding the question/answer tuples without losing recall.

WebKnox Q/A is part of the **Web Kno**wledge E**x**traction system (Urbansky et al., 2009). Research results and demonstrations will also be published at http://www.webknox.com.

Basically, there are two approaches in question answering. One is to compute the answer to a question with a given structured knowledge base (e.g., TrueKnowledge.com) or to match a new question to answers of similar questions (Jeon et al., 2005) that are stored in the knowledge base (e.g., Wiki Answers). The former approach works best with factoid questions such as “What is the capital of Italy?” whereas it has more difficulty answering the question “Is downshifting a good way to slow down my car?” since it is harder to compute an answer for that. In this paper we will focus on the latter approach.

Extracting and using question/answer pairs has long been recognized as being useful for serving answers to users with new questions. FAQFinder (Burke et al., 1997), retrieves Frequently Asked Question (FAQ) files and performs question answering on them by comparing a new question to questions from the indexed set. The problem of finding and extracting FAQs using heuristics and machine learning has been studied in (Jijkoun and Rijke, 2005) and (Lai et al., 2002) and can be considered highly reliable with about 0.94 precision and recall (Jijkoun and Rijke, 2005). MULDER (Kwok et al., 2001), Agichtein et al. (2004) and Omnibase (Katz et al., 2002) try to find the answer to a given question directly on certain web pages instead of crawling them offline. Recently, Cong et al. (2008) studied the detection and extraction of question/answer pairs in online forums using labeled sequential patterns and machine learning. They were able to extract over two million questions with a precision of up to 0.88. Our work differs from the aforementioned, in that we focus on the extraction and not the question answering part and that we use Q/A rich websites instead of FAQs, blogs or online forums.

2. Q/A EXTRACTION

In this section we describe the basic design of Q/A rich websites and how we can use it to configure the Q/A extractor. Furthermore, we explain our focused crawling algorithm that increase the extraction efficiency.

2.1 The Q/A Extractor

2.1.1 Q/A website design and WebKnox Q/A configuration

Q/A rich sites all have a similar layout. There are three possible states for a question page on a Q/A website:

1. There is only one question and nobody has answered yet.
2. There is one question and one or more answers.
3. There is one question and one “best answer” and none or more other answers.

We are only interested in the pages that have a question and at least one answer must have been given. To extract question and answers, we assume that every question and answer is structured using (X)HTML elements, that is, we assume that we can create an XPath (Clark and DeRose, 1999) that targets the question or answer without too much noise around it. So instead of detecting questions and their answers by lexical or semantic analysis we will take user given XPaths to the question and answer part(s) of the page. This is a much more effective (since the path is generated by a user) and efficient (since parsing and classifying is time and resource consuming) way than using heuristics and machine learned classifiers. The downside is, that we need the user to find those XPath in advance, but since the number of Q/A rich websites is quite small, we think the benefit of having very accurate extractions is worth the little effort. Figure 1 shows an excerpt of the file that is used to configure WebKnox Q/A. We have chosen yaml[1] as a serialization format since it can be read and edited very easily by a human user.

```
sites:
    - name:            "Yahoo! Answers"
      entryURL:        http://answers.yahoo.com
      questionXPath:   "//div[1]/div[1]/div[2]/h1"
      bestAnswerXPath: "//div[1]/div[1]/div[2]/div[2]/div[1]"
      allAnswersXPath: "//div[3]/div[2]/ul/li/div/div[2]/div[1]"
      answerPrefix:    "ratings ) "
      answerSuffix:    " Comments Sign in"
```

Figure 1. Simple yaml configuration file for WebKnox Q/A.

1 http://www.yaml.org

We need 7 parameters per page that can be obtained in less than two minutes for an experienced user. The "name" parameter is optional and specifies the name of the website, "entryURL" specifies the start page for the crawler, "questionXPath" is the XPath that points the question section of the page, "bestAnswerXPath" points to the section with the chosen answer (described as state 3 above) and "allAnswersXPath" points to all (other) answers for a given question. Not every site makes a distinction between "best answer" and other answers but if they do, often different XPaths must be used to extract the answers. The "answerPrefix" and "answerSuffix" parameters are used to extract the answer string without the noise around it by specifying what must appear before and after the answer string.
The extraction is then performed by retrieving the contents of the Xpath for every candidate page. To clean up the answer string, the answerPrefix and answerSuffix are deleted from the string. Which pages are candidates for Q/A extraction is explained in the following section.

2.1.1 Focused Crawling

The trivial way to find all question/answer pages in a certain domain is to crawl the complete domain from one start page. On every page, all hyperlinks are added to the URL stack and we try to find the question and answers by using the specified XPaths. This approach is a breadth-first search without any focus, and therefore, is very inefficient. We use focused crawling to faster retrieve the pages where the Q/As can be found. For that purpose we assume that the URLs that point to Q/A pages all have a similar prefix. For example, the two Q/A pages "http://wiki.answers.com/Q/What_is_the_speed_of_light" and "http://wiki.answers.com/Q/Why_is_the_sky_blue" have the *common prefix* "http://wiki.answers.com/Q/". This assumption is likely to be true for many Q/A rich websites, since the pages are generated with database contents and templates.

In order to focus the crawler, we classify every page in a given domain into three categories:

1. 🟩 "Green", that is, a question (and an answer) was found on the page
2. 🟨 "Yellow", there are no Q/As on the page but it directly links to "green" pages with Q/As
3. 🟥 "Red", there are no Q/As on the page and it is unknown whether the page links to "green" pages

For the green class we learn one *common prefix* and the number of "/" separations (*address depth*) in the URL. For the yellow and red class we each learn a set of *common prefixes* and also store the a*ddress depth* for *common prefixes* from the red class.

The classification is done using prefix filtering which works as shown in Figure 2. A page is fetched from the URL stack (first page is the "entryURL" specified in the configuration file). All links to other pages within that domain are added to the URL stack. Then, a question and an answer are sought and extracted using the specified XPaths. If no question was found, the page is added to the red class. If a question was found, the *common prefix* for green pages is updated with the current URL and its parent URL (the URL that linked to the current one) is classified as yellow. In the next iteration, a green class URL is retrieved from the stack, that is, the prefix of the URL from the stack must match the learned *common prefix* for the green class. If no green page was found, it is attempted to find a page from the yellow class since it is known that these pages point to more green pages. If there are neither green nor yellow URLs on the stack, a URL which is not from the red class is taken from the stack, that is, we try to get an unclassified URL since we know that URLs from the red class definitely do not contain questions and we should explore another URL path.

After only a few analyzed URLs, the prefixes for all three classes are learned and pages containing Q/As (green class) are preferred over yellow and red pages. This leads to more extracted questions after the same amount of time while the recall stays the same compared to non-focused crawling.

In the next section we evaluate our focused crawling algorithm by comparing it to a non-focused, trivial crawler.

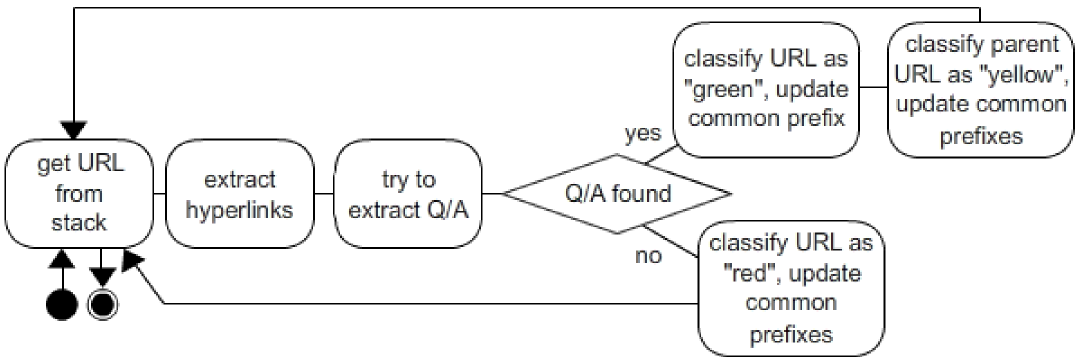
get URL from stack
extract hyperlinks
try to extract Q/A
Q/A found
yes
no
classify URL as "green", update common prefix
classify parent URL as "yellow", update common prefixes
classify URL as "red", update common prefixes

Figure 2. Process of learning the classification of URLs into three classes.

2.2 Evaluation

In this section we evaluate our focused crawling approach by comparing the number of extracted questions and the size of the URL stack to the trivial crawling approach. We use seven Q/A rich websites for this experiment (Wiki.Answers.com, Answers.Yahoo.com, StackOverflow.com, ChaCha.com, Mahalo.com, Answers.Nobosh.com and HubPages.com/Answers/latest) and sum the number of extracted Q/As over all websites. As a baseline, we use the trivial approach which does not perform any classification and simply takes the next available URL from the stack. In Figure 3, we can see that after 5,000 URLs have been visited, WebKnox Q/A was able to extract 99.5% more questions using focused crawling compared to the baseline crawling. Since WebKnox Q/A prefers URLs with Q/As, the stack size is increasing about 30% slower and thus saving resources. Also, URLs with red prefixes are deleted from the stack which explains the slight drops of the URL stack size in Figure 3 (red line) at about 300 and 2,000 visited URLs.

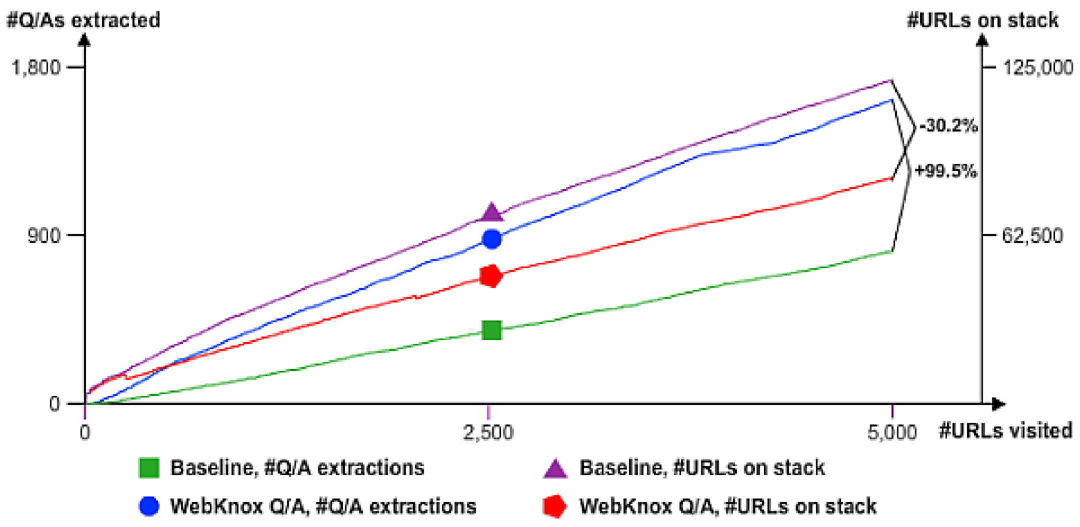

Figure 3. Comparison of efficiency between a non-focused baseline crawler and WebKnox Q/A.

3. CONCLUSION

In this paper we have presented WebKnox Q/A, a question/answer extraction system that uses focused crawling with simple configuration to extract question/answer tuples from user centric question answering sites. Our system can be used to quickly create a huge corpus of Q/As across several Q/A rich sites without requiring the user to create special “wrappers” and crawling policies for each new site.

The described focused crawling approach is more efficient than trivial crawling, leading to almost double as many extracted question/answer pairs for the same number of visited URLs. The preference of the algorithm for Q/A rich wesites also saves memory resources as the URL stack increases more slowly, having about 30% less entries.

In future work we will investigate how we can extend the system to also reliably find and extract FAQs from specified domains. We would also like to add features for finding and extracting Q/As from blogs and online forums using a very similar approach.

REFERENCES

Agichtein et al., 2004, Learning to find answers to questions on the web. *ACM Transactions on Internet Technology, Vol. 4, No. 2, pp 129-162.*

Burke et al., 1997, Question Answering from Frequently Asked Question Files: Experiences with the FAQFinder system. *AI magazine, Vol. 18, No. 2, pp 57-66.*

Clark and DeRose, 1999, XML Path Language (XPath) Version 1.0. *World Wide Web Consortium,* http://www.w3.org/TR/xpath

Cong et al., 2008, Finding question-answer pairs from online forums. *Proceedings of the 31st annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* pp. 467-474.

Jeon et al., 2005. Finding similar questions in large question and answer archives. *Proceedings of the 14th ACM International Conference on Information and Knowledge Management.* pp. 84-90.

Jijkoun and Rijke, 2005, Retrieving answers from frequently asked questions pages on the web. *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management.* Bremen, Germany, pp. 76-83.

Lai et al., 2002, FAQ mining via list detection. *Proceedings of the International Conference On Computational Linguistics*. pp. 1-7.

Katz et al., 2002, Omnibase: Uniform access to heterogeneous data for question answering. *Lecture notes in computer science, pp 230-234.*

Kwok et al., 2001. Scaling question answering to the Web. *Proceedings of the 10th International Conference on World Wide Web.* pp. 150-161.

Urbansky et al., 2009. Entity Extraction from the Web with WebKnox. *Proceedings of the 6th Atlantic Web Intelligence Conference.*