An Open-Domain Question Answering System Using Annotated Web Feeds

Bukhary Ikhwan Ismail[1*], Yu-N Cheah[1]

[1] *School of Computer Sciences, Universiti Sains Malaysia, 11800 USM Penang, Malaysia*

Open Domain Question Answering Systems (ODQA) aim to answer all possible questions, regardless of topic and time. For this to be possible, most current ODQA systems depend on the World Wide Web through search engines, e.g. Google, which provide an abundence of information. Problem arises when search engines require a few days to crawl, archive and index the latest documents depending on the popularity of the website and the search engine's indexing efficiency. Thus, recent information may not be available as soon as it is published. Our work focuses on capturing and populating current news articles from various trusted resources into a single unified repository. We have implemented a prototype which uses Web 2.0 RSS feed technology to capture the information. This includes an interface which allows other question answering systems to access our repository using various formats, e.g. XML and SQL query. We have also implemented a question answering engine which employs keyword detection, query expansion and rule matching.

1. Introduction

Information has become increasingly accessible online. The over-abundance of information has resulted in a society that is information-rich but yet attention-poor (1). When attempting to search for information online, a user is often inundated with numerous search results, many of which are not relevant. Question answering (QA) systems are seen as a way to deliver information, answers in particular, in a more concise manner and in a way that captures the attention of the user. It also has the advantage of allowing users to ask for information or answers in natural language. Some experts argue that QA systems will be the next wave of information seeking tool, eventually replacing the current search engine techniques in the future.

In this paper, we present an Open Domain Question Answering System (ODQA) called OpenQA that incorporates the use of annotated web feeds from news agencies. By this, we provide the latest information to be searched by the QA engine which enables the OpenQA system to provide answers with this latest information. Current QA systems do not focus on this issue of information currency. The Text Retrieval Conference (TREC) community argues that even with the best QA engine available, it could not provide relevant answers without a good repository (2). Thus, by capitalising on web feeds, the OpenQA system is able to address this problem.

2. Related Work

ODQA is a system which can answer questions regardless of topic, domain or time. It relies heavily on a large amount of information to cover all possible questions and domains. Thus, the internet is the best source of information. Here, we present several related works which uses the internet as its main source.

ARANEA (3) is developed by the MIT Artificial Intelligence Laboratory. It tests the extraction of information using the WWW and specialised databases. One strategy used is via knowledge annotation which effectively answers common occurring questions using annotated structured and semi-structured resources such as the CIA World Fact Book and IMDB Online. Another strategy is through knowledge mining which utilises statistics and takes advantage of the massive amount and redundance of information on the web via search engine.

GuruQA (4) uses a customised version of a search engine on the WWW to answer factoid question. The system tries to answer open question and shows promising results when evaluated using the TREC-8 question set. It uses natural language processing (NLP) and information extraction techniques with a customised text search called predictive annotation which produces effective QA.

There is also work to detect possible surface text pattern for ODQA using search engine (5). For example, a BIRTHDATE question: When was X born? (X denotes a person). The answer pattern can then be extracted such as "<NAME> was born <BIRTHDATE>" or "<BIRTHDATE> <NAME> ". An attempt was made to extract these patterns automatically by querying on the AltaVista search engine. These patterns are then applied to find answer to new questions using the TREC-10 question set.

In general, ODQA relies heavily on the internet as its main information source. Even when such vast info is available, the information retrieval (search engine) process could not acquire the latest information. The limitation lies within the search engine itself. The TREC community has addressed this problem and have observed that "new data sources must be incorporated in the QA systems as soon as they become available, offering the user an answer even when the question refers to the most recent events or facts" (2).

3. Our OpenQA Methodology

Our OpenQA system consists of two engines (see Fig. 1): 1. annotated text repository population engine, and 2. question answering engine.

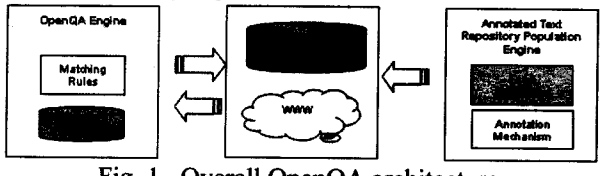

Fig. 1. Overall OpenQA architecture

3.1 Annotated Text Repository Population Engine

The annotated text repository is populated using RSS web feeds. The annotated text repository population engine consists of four components (see Fig. 2).

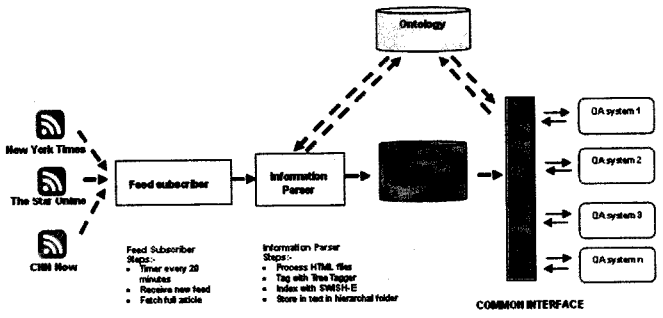

Fig. 2. Overview of the Annotated Text Repository Population Engine

3.1.1 Feed Subscriber

The feed subscriber has two main functionalities. Firstly, it allows subscription to web feeds so that it can grab the latest information (news articles) available online. Subscription to more than one feed of the same category (e.g. sports, news, etc.) will promote redundancy of information. This redundancy leads to two advantages: 1. the burden of the QA system having to perform complex NLP techniques on the text will be lessened in view that possible correct information will appear in different forms, and 2. correct information can be filtered from incorrect ones (correct information tend to appear more frequently than incorrect ones) (6).

The second functionality of the feed subscriber is to automatically fetch the subscribed feeds. These feeds are usually in RSS or atom format. It will automatically fetch new information on the regular basis (e.g. every few minutes depending on the user specification). For our purpose, we have set the frequency to once every half-an-hour. When new information is available, the fetching mechanism proceeds to retrieve the full article using the link found in the XML tagged feed. The full article is then passed to the information parser component.

3.1.2 Information Parser

In this component, the full article is parsed to locate the beginning and the end of the actual content. These unique identifiers (where relevant) are manually made known to the information parser by the user. For example, news feeds from The New Straits Times (a Malaysian newspaper), the HTML formatted article has <!--start full story--> and <!--end full story--> at the beginning and the end of the actual content respectively. These details are stored in a database together with information about the feed's title and link.

After the actual content has been identified, all other tags are removed, leaving only the plain original text. Finally, this cleaned content is tagged using a part-of-speech (POS) tagger. For our purpose, we used TreeTagger and Penn-Tag set which can categories up to 36 POS.

3.1.3 Annotated Text Repository

The core of any information retrieval or QA system is the repository. Without it, the system could not retrieve any information or reason with it. Here, the tagged content of the full article is stored in a repository. The repository essentially stores the following fields: id, title, link, description, update, full article, annotated article, status and file location.

3.1.4 Common Interface

The common interface is targeted to be generic enough so that other QA systems will be able to use the repository. It can be plugged-in to other existing QA systems which can enrich or add value to the current system. The common interface component indexes the content of the annotated text repository. The index is used to facilitate the search for relevant content in the QA subsystem. It is through this common interface that the QA engine will submit the question-related inputs.

3.2 Question Answering Engine

The question answering engine consists of five components (see Fig. 3).

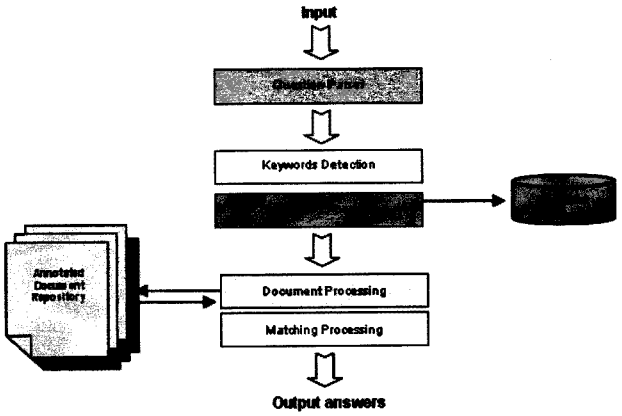

Fig. 3. Overview of the Question Answering Engine

3.2.1 Question Parser

A rather simple method is applied here. Special characters such as commas, full stops, etc. are removed. Then, the question will be POS-tagged. Words such as verbs, adverbs, nouns and adjectives will be taken into account as potential keywords for the next step.

3.2.2 Keyword Detection

Our approach for question answering relies on keywords. However, for our OpenQA system we aim to make intuitive choices of keywords from the question. The remaining words from the question parser will be matched against a collection of stop words and a collection of 830 empty English words[1] which are deemed to be less meaningful as a search keyword (e.g. words like be, hello, say, etc.). If any of these words are found, they are also removed from the list of potential

[1] Empty English - words consisting of prepositions, stop words, etc. which are not suitable to be keyword candidates. The list can be obtained from Professor Lluís Padró's website: http://www.lsi.upc.es/~padro/lists.html.

keywords. Note that as a result, keywords together with their part-of-speech are obtained. Fig. 4 illustrates how keywords are detected from an input question.

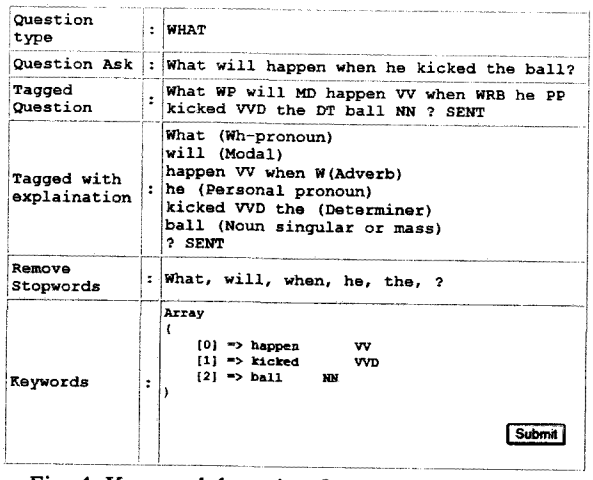

Fig. 4. Keyword detection from an input question

3.2.3 Keyword Expansion

After obtaining the list of keywords, we then derive their synonyms and different word senses based on the WordNet general ontology. WordNet recognize four types of POS which are nouns, verbs, adverbs and adjectives. After the keywords have been filtered, we fetch every sense of the word for each keyword. For example, if the keyword is "bush", it will be expanded to include "bush", "shrub", and the proper noun "Bush" (see Fig. 5).

```
Synonyms/Hypernyms (Ordered by Estimated Frequency) of noun bush
   7 senses of bush

   Sense 1
   shrub, bush

   Sense 2
   Bush, George Bush, George W. Bush, George Walker Bush, President Bush, President George W. Bush, Dubyuh, Dubya
```

Fig. 5. Sample synonyms and senses

The user is then allowed to select the synonyms. By expanding the keywords this way, we ensure that other related words are taken into consideration when trying to obtain the correct answer. Automating the detection of the correct sense can be done, but presently we allow the user to select it manually.

3.2.4 Document Processing

The user selected keywords and their respective synonyms/different senses are then used in the query to obtain all matching annotated articles from the repository. The matching is not only done based on the keywords but also based on the parts-of-speech (recall that the annotated articles have been POS-tagged).

3.2.5 Rule Matching

Finally, from the annotated articles, we first split the articles into sentences (detected using the full-stop symbol). Then, we employ question answering rules adapted from the QUARC (Question Answering for Reading Comprehension) (7) system (see Table 1) and attempt to look for relevant content that may answer the question. Only "who", "where" and "when" questions are considered in our OpenQA system. Each rule will be applied to each sentence and the sentence that matches the rules would accumulate weights that are assigned by the rules. The sentence with the highest weight is deemed to answer the question.

In Table 1, the semantic classes, i.e. LOCATION, HUMAN and NAME, are derived from WordNet.

Table 1. QUARC rules

TYPE	DESCRIPTION	RULES
When	• Evaluated using a special set of **dateline** rules • Several keywords identify the time or when it happens items e.g. {Yesterday, tomorrow, first, last, since, ago, start begin} and more. • Use WordMatch function[2]	1. Score(S) += WordMatch(Q,S) **THEN** Score(S) += **good_clue** Score (S) += WordMatch(Q,S) 2. **IF** Contains (Q, *the last*) and contains(S, *{First, last, since, ago}*) **THEN** Score (S) += **slam_dunk** 3. **IF** contains (Q,{start, begin}) and contains (S, *{Start,begin, since,year}*) **THEN** Score(S) += **slam_dunk**
Who	• Use NAME and PERSON class. PERSON contains person jobs, such as writer, poet and Dr. Mrs. Mr. and more. NAME contains famous person names. • **Rule #1** if the question (Q) does not contain any names, then • **Rule #2** rewards sentences that contain a recognized NAME, • **Rule #3** reward sentence the word n"name"	1. Score (S) += WordMatch (Q,S) 2. **IF** contains (Q,**NAME**) and contains (S, **NAME**) **THEN** Score(S) +=**confident** 3. **IF** contains (Q,**NAME**) and Contains (S,name) **THEN** Score(S) += **good_clue** 4. **IF** contains (S, {**NAME,HUMAN**}) ***THEN*** Score(S) += **good_clue**
Where	• Here we will use several location preposition (LocationPrep) such as "at, behind, under, etc") • **LOCATION** class is a list derived from WordNet ontology of common location places and its synonyms.	1. Score(S) += WordMatch(Q,S) 2. **IF** Contains(S,*LocationPrep*) **THEN** Score(S)+= **good_clue** 3. **IF** contains(S,**LOCATION**) **THEN** Score(S) +=**confident**

4. Results & Discussion

The implementation of our OpenQA system shows that the system is able to produce answers to input questions. Here, we would like to highlight that the acquisition of documents via web feeds allows questions to be answered based on the latest available information (as long as the article is available on the web).

[2] WordMatch function strips a sentence from its stopwords.

Table 2. The Star published article vs. Google's Cached time

Article Title	Date Published	Cached by Google	Differences
Electricity supply to Kota Tinggi restored	2006-12-25 17:23:19	None	**None**
Floods update: Situation in Johor improves	2006-12-25 15:36:02	None	**None**
Floods this Christmas an alarm bell from....	2006-12-25 15:40:99	None	**None**
Time running out for kidney patient	2006-12-23 02:39:96	2006-12-23 11:06:22	**09:33:74**
Couples cry foul over fake certs	2006-12-23 01:40:64	2006-12-23 08:58:00	**07:18:64**
Flood Update : Two found dead in Segamat	2006-12-22 16:39:43	2006-12-23 10:26:29	**18:13:46**
Firemen's leave frozen	2006-12-22 15:39:26	2006-12-23 10:18:06	**19: 39:20**
Floods update: Bus falls into collapsed	2006-12-21 03:04:11	2006-12-21 13:22:22	**11:18:11**
Teacher told colleague she was stranded	2006-12-21 12:04:11	2006-12-22 08:59:06	**20:04:11**

Table 2 shows the comparison between the date and time an article is available online and the date and time the article is cached by Google. We chart the results of the last six articles from Table 2 (see Fig. 6) and found that Google needed a minimum of 9 to 20 hours before it can cache the website's contents. After the content has been cached, the content will only be available to the user approximately one more day later. This is probably due to the working mechanism of the search engine which only indexes the article after certain number of cached items has been obtained. This is to decrease the workload of the Google server every time a new article is available on the internet. But for our purposes, this reduces the availability of information to a QA system which relies on search engines. To our advantage, since we rely on web feeds, our OpenQA system can be instructed to access frequently updated websites to obtain new content more efficiently.

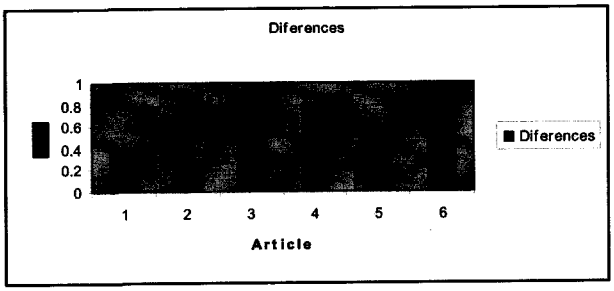

Fig. 6. Time comparison between date published and cached by Google

From our implementation, we observed that keyword expansion using WordNet does enhance the recall of documents (in our case annotated articles). By listing all available synonyms and senses, users can interactively clarify their input question, thus reducing the ambiguity of words with similar/different meanings. This is just one way of utilising an ontology to enhance information retrieval and QA system results.

We also observed that matching POS-tagged keywords to POS-tagged documents does increase the precision of question answering. In our experiment, the question "What will the lion do with its kill?" ("kill" here being a noun) would not return sentences with the word "kill" as a verb.

5. Conclusion

Using web feeds such as RSS and atom is effective in capturing the latest information directly from the content provider as it is published. Due to the fact that most QA systems rely heavily on the WWW via search engines, the OpenQA system does save time by using web feeds. The amount of time saved varies from one feeds to another. It also depends on the popularity of the website. From our survey, search engines do not index web feed content. Since most ODQA systems use search engines, we can say that our method can complement existing systems.

We hope to extend our work further to focus on more intuitive answer generation as opposed to returning the entire sentence that contains the answer. Further NLP techniques need to be applied to realise this goal.

6. References

(1) G. Fischer and J. Ostwald: Knowledge Management: Problems, Promises, Realities, and Challenges. *IEEE Intelligent Systems*, Vol. 16, No. 1, pp. 60-72 (2001)

(2) J. Burger, C. Cardie and V. Chaudhri: Issues, Tasks and Program Structures to Roadmap Research in Question & Answering (Q&A). Available at http://trec.nist.gov (2006)

(3) J. Lin, A. Fernandes and B. Katz: Extracting Answer from the Web Using Knowledge Annotation and Knowledge Mining Technique. *Proceedings of the Eleventh Text REtrieval Conference (TREC 2002)*, Gaithersburg, Maryland (2002)

(4) J. Prager, E. Brown and A. Coden: Question-Answering by Predictive Annotation. *SIGIR 2000*, Athens, Greece (2000)

(5) D. Ravichandran and E. Hovy: Learning Surface Text Patterns for a Question Answering System. *Proceedings of the ACL Conference*, 2002

(6) A. Lampart. (2004). *A Quick Introduction to Question Answering.* Available at: http://ict.csiro.au/staff/Andrew.Lampert/writing/IntroductionToQuestionAnswering.pdf

(7) E. Riloff and M. Thelen: A Rule-based Question Answering System for Reading Comprehension Tests. *ANLP/NAACL-2000 Workshop on Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems*, Seattle, Washington, 2000