3-2012

The Severity of Undetected Ambiguity in Software Engineering Requirements

Cristina Ribeiro
Sheridan College, cristina.ribeiro@sheridancollege.ca

The Severity of Undetected Ambiguity in Software Engineering Requirements

Conference Paper · March 2012

CITATIONS
0

READS
16

1 author:

Cristina Ribeiro
University of Waterloo
16 PUBLICATIONS **59** CITATIONS

SEE PROFILE

Sentence Level Fact Based Search Engine: News Fact Finder

Cristina Ribeiro	Ricardo Salmon	Swathi Amarala	Christina Hamada
Computer Science	*Computer Science*	*Computer Science*	*Computer Science*
University of Waterloo	*University of Waterloo*	*University of Waterloo*	*University of Waterloo*
200 University Avenue	*200 University Avenue*	*200 University Avenue*	*200 University Avenue*
Waterloo, Canada	*Waterloo, Canada*	*Waterloo, Canada*	*Waterloo, Canada*
cribeiro@uwaterloo.ca	*rsalmon@uwaterloo.ca*	*samarala@uwaterloo.ca*	*chamada@uwaterloo.ca*

Abstract

Users searching the Internet for news are not able to find relevant fact-based results for certain queries using the major search engines. Queries that require exact substring matching in order to obtain very relevant results are not currently possible. Furthermore, search engines do not discriminate in returning results that are opinions and not quantifiable facts. Our sentence level search engine, News Fact Finder, is designed using suffix arrays, filters out opinions, and produces very relevant results that are attractive to users. The News Fact Finder produces a 73% success rate of providing relevant fact based results.

Keywords: Heuristic Optimization and Search, Sentence Level Search Engine, Suffix Arrays, Natural Language Processing

1. Introduction

Even though a sentence level search is feasible in practice, it is believed that current document level search engines, such as Google, cannot be outperformed. However, we find a few weaknesses in the top search engines used that can be addressed by sentence level search. Sometimes these engines will return a result that contains all of the specified keywords, but the keywords are not closely related to each other, and thus the document is not relevant to the intended search. Also, these engines fail to find long queries within quotes, unless the quotes contain well known text, such as that of a Shakespeare play. Otherwise, it will return, "No results found", even though such text exists on the web. Furthermore, the results it suggests without quotes are most usually not relevant. Searching at the sentence level would provide these additional benefits to search.

However, large search engines do not implement sentence-level search because there is no fast and easy way to index and search over sentences, as there is for words with an inverted index. We believe that the best way to implement sentence-level search is with suffix arrays that excel at exact substring search. However, each sentence would have to be searched each time a query is made; this search is not scalable over a large web-sized corpus. Therefore, we focus our search engine on a smaller subsection of the web.

The idea behind sentence level search is if all the query words are in the same sentence, that result is more likely to be relevant than a result containing the query words in different parts of the text. Taking this into consideration can enhance the precision of the results set without affecting sensitivity. Generally it is hard to increase the specificity of a search without decreasing the sensitivity and thus losing a number of relevant results. Sentence level search is proposed as a way to overcome this problem.

Currently none of the major search engines in use are implemented using sentence level search. Instead they have inverted indexes, used to search for documents that match the user's query. We are of the opinion that sentence level search could be used in certain contexts to produce better results than inverted indexed search engines. The difference between the two methods is in the unit used in searching. For an inverted index, the unit is a document, and for sentence level search, the unit is a sentence.

This paper outlines some current examples of sentence level, and opinion based search engines in section 2. The algorithm of a News Fact Finder sentence level search engine is discussed in section 3. Section 4 describes the methodology. The results are shown in section 5. Conclusions are listed in section 6 and future work is discussed in section 7.

2. Background

There are many different sentence based search engines. Relemed [1] is a sentence level search engine for

978-1-4244-8098-2/10/$26.00 ©2010 IEEE

MEDLINE, which is a database of over 15 million biomedical article citations. Because of the large size of the corpus, it is common for queries to return extraneous results using traditional document level search techniques. Relemed is the first to explore sentence level searching. Like many other search engines, Relemed uses the Unified Medical Language System to automatically map terms to keywords in order to improve the sensitivity of the search. Relemed uses the relevance metric in conjunction with sentence level search to rank results.

Two case studies were conducted to compare Relemed's and PubMed's results. These preliminary studies found that Relemed and PubMed returned the same number of results, and that Relemed displayed a high number of relevant results in the first few pages, with over 98% precision [1]. Relemed also was shown to rank false positives lower. Since PubMed posts results chronologically, the systems cannot be compared on this level. However, Relemed was successful at accomplishing its goals. These results show the promise of sentence level search to improve precision and ranking of search queries.

AnswerBus is an open-domain natural language question answering system based on sentence level web information retrieval [2]. It accepts users' natural language questions. Based on the user's question, AnswerBus selects two or three search engines from among the five and forms search engine specific queries based on the question. AnswerBus then contacts the selected search engines and retrieves documents at the top of the hit list. It then extracts the sentences that might potentially contain answers from the documents. It ranks the documents and returns the top choices with contextual URL links to the user.

TREC-8's 200 questions were used to evaluate AnswerBus' question answering performance, and also its performance was compared to four other similar question answering systems. The rate of correct answers returned to TREC-8's 200 questions is 70.5 % and the average time taken to respond to a question is seven seconds. The performance of AnswerBus in terms of accuracy and response time is better than other similar systems.

AnswerFinder is a general-purpose web-based question answering system aimed at answering simple questions that require a single fact as an answer. AnswerFinder's system consists of three main phases: determining the expected answer type, using a search engine to find relevant documents, and a final stage in which possible answers are located within the relevant documents [3]. All the retained entities are grouped to form answer groups based on the equivalence test. Two answers are said to be equivalent if all the words (excluding stop words) in one are present in the other. These answer groups are then ordered based on the number of occurrences of each answer group within the documents that were processed and on the rank of the document in which the answer was first found. Precedence is given to those answers found in documents regarded as highly relevant by the information retrieval system. This ordered list is then returned to the user. AnswerFinder's performance was evaluated using the TREC 2002 question set. AnswerFinder was capable of correctly answering approximately 26% of the questions [3].

An opinion search engine [4] uses sentences in open domain topics. Their search engine involves extracting opinion sentences from Japanese blog pages that are relevant to a user's query. As the size of the web grows, more content is user generated, in the form of blog pages, emails, and social networks. The motivation behind these types of search engines would be to infer users' opinions on a product, users' belief regarding a topic, or users' shifting trends. This knowledge can also be informative for decision-making tasks.

Opinion sentences are classified based on the particular type of opinion, i.e., sentiments, neutral opinions, requests, advice, or thoughts. Each opinion sentence is identified if an opinion clue is explicitly found, i.e., "I am glad" from the sentence "I am glad to see you" or "extremely" from "They played extremely well." Sentences containing the exclamation mark and conditionally subjective phrases are also identified as opinions. These clues are encoded as features in a Support Vector Machine to perform the classifying if a sentence is opinionated.

Opinion clues are also augmented with semantic categories to compensate relations between co-occurring words phrased in sentences. The semantic categories encode a hierarchal relationship between words. The evaluation consists of collecting pages from the web and having three judges manually labeling sentences they judge to be opinions. To illustrate the difficulty of the task: the sentences determined to be opinions if at least one judge deemed the sentence to be an opinion. All three judges only agreed that a sentence is an opinion 22% of the time [4]. In terms of performance, the system was tested against (a) the baseline method, (b) a proposed model with expression clues for opinion extraction, and, (c) the effects of adding semantic categories. The results show that the opinion sentence search engine outperformed the baseline method on the test criteria of precision, recall and accuracy.

3. Algorithm

We investigated sentence level search engines further than current research available and their benefits to improving web searches. We implemented a sentence-level search engine, called News Fact Finder, which indexes sentences using suffix arrays rather than the keywords. A Suffix array is a data structure containing all pointers to the text suffixes, which are sorted in

lexicographical order [5-9]. Each suffix is a string starting at a particular position in the original text and ending at the end of the text, as seen in Figure 3.1. Given keywords as inputs, we rank and return documents based on the exact substring match of the keywords in the same sentence.

Suffix											Index
a	b	r	a	c	a	d	a	b	r	a	0
	b	r	a	c	a	d	a	b	r	a	1
		r	a	c	a	d	a	b	r	a	2
			a	c	a	d	a	b	r	a	3
				c	a	d	a	b	r	a	4
					a	d	a	b	r	a	5
						d	a	b	r	a	6
							a	b	r	a	7
								b	r	a	8
									r	a	9
										a	10

Figure 3.1 Suffix Array

A natural progression would be to classify sentences into different categories. Recent work by [4] has shown success in using sentence level search to classify sentences based on types of opinion from the author. The classes used were specific to a Japanese Blog portal. However, we classify sentences based on their functional and structural form to better suit the needs of users. For example, if we know the user asked a question in the query, then it would not make sense to return a sentence that we know is a question in the answer set. In order to test our search engine, we would like to run it against a large corpus, and compare it with manually ranked human queries. We believe that our system will be able to successfully return pertinent facts relating to the users' queries.

The implementation of the News Fact Finder system is comprised of five components, which includes the following: crawler, parser, index, query, and ranking. The architecture is shown in Figure 3.2.

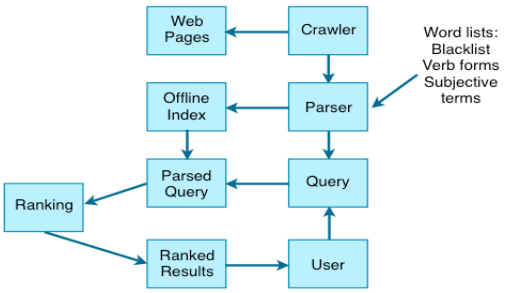

Figure 3.2 News Fact Finder architecture

Since sentence level search is performed, each html document is parsed into sentences. The parser applies the black list, verb list and subjective word list to the parsed documents. The patterns to search and eliminate the text are compiled using regular expressions.

Then the text is split into sentences on the occurrence of break. A list of prefixes correctly classifies the sentences. The last word of each sentence is checked against the prefixes in this list: if a match is found, then this sentence is combined with the next sentence. Sentences are also not broken down on the occurrence of abbreviations such as U.S., R.S.V.P, etc., or on the occurrence of middle initials such as Steve F. McCormick, see Algorithm 1.

```
function search(query)
    words = parseQuery(query)
    If empty?(words)
        return Nothing
    keywords = applyWordList(words)
    # return sentence if it contains a keyword
    matches = matchingSentences(keywords,index)
    # longer sentences with matching keywords score greater
    # documents with more results have matching sentences score
    greater
    list = rank(matches)
    for each sentence in list
        show k proceding and k subsequent sentences
    return result
```

Algorithm 1 Pseudocode

Various lists are then applied to the parsed documents from the first stage. In order to ensure that our system retrieves the most relevant information and only fact based results, we created a set of lists to accomplish this. We developed three lists: a verb list, a black list, and a subjective list. Each list is used in different components of the architecture.

The second stage of the parser reads each sentence from the parsed documents, splits the sentence into a list of words, eliminates all the words from the black list occurring in the sentence, and converts all the verbs into their root forms. It then checks to see if any word in the sentence is from the subjective list. If the sentence contains a word from the subjective list, then the parser eliminates the sentence in total. Otherwise the parser joins all the words back to form a sentence, and writes the sentence to a text file.

The reason we convert various verb forms to the root form is to have uniformity of verbs throughout the document, and hence retrieve all the possible relevant sentences to a user's query. The verb list format can be found in Table 3.1.

Table 3.1 Verb list for document uniformity

Non Base Form Verbs	Verb Base Form
Believed	Believe
Believes	Believe
Believing	Believe

The black word list is used by the parser to eliminate common words found in the English language that are not relevant to the topic, such as the word "the." This includes definite and indefinite articles, prepositions, some verbs,

and conjunctions. The black list used in the News Fact Finder consists of 131 words.

Originally, we were going to create a list of nouns to include instead of using a black list to exclude words. It is much more efficient and effective to use a black list instead. The time required creating a black list in comparison to a list of all possible nouns that might occur is considerably less. This translates into another reason, which is that the algorithm will be much more efficient in parsing out the black list words. Searching through the list is much quicker due to its size, which is considerably smaller than that of a complete noun word list.

The subjective word list contains words in the English language to describe a topic that suggests an opinion or a point of view. The word list was created using a subjectivity clues database, which contains subject words and classifies them by type, either weak or strong. We created a list containing only the strong subjective words, to ensure that the system is not too restrictive in the sentences it eliminates as opinions. The weak subjective terms do not necessarily apply in all cases and contexts. Even with natural language processing it is difficult to determine when the weak subjective words comprise an opinion versus a fact. Despite the fact that we only include strong subjective words, our list is quite lengthy. The exhaustive list contains 4745 words. By using this list in our system, we are testing a brute force approach to classifying sentences as facts or opinions.

At this point, each document has been parsed into sentences, with blacklisted words and sentences containing opinions removed. We then build an index of sentences. We decided to create a suffix array for each sentence, because we wanted to be able to do substring searching quickly. Building these suffix arrays can take a long time, we initially create the index and the suffix arrays, then write it into a file. Then when we search, our search engine reads the index into memory from file, without having to build the suffix arrays each time. Because suffix arrays are not a common data structure, we provide a custom implementation for our purposes.

The suffix array class stores two attributes: 1) the sentence (which is actually a list of words in the sentence); and 2) the suffix array. It contains functions to build a suffix array given a sentence, and search the suffix array for exact matches, given a string of one or more words. It also has a function that returns a string of the sentence and the suffix array, separated by the delimiter "|", which is how the suffix array is saved to disk.

The index builder reads in one document at a time and turns each sentence into a suffix array, and also keeps track of its document ID and sentence ID. Sentences that are excluded from the final parsed documents because they contain opinions are denoted by blank lines, and although suffix arrays are not created for them, they are still given sentence IDs, in order to trace them back to the parsed documents and exclude them from being returned in the results as neighboring facts. The suffix arrays are stored in one list that is iterated through when searching. Each unit of a document ID, sentence ID, and suffix array is stored as a tuple in the list.

When matches are found, we scan the original corpus to find the matching sentence before filtering so it would be still meaningful and comprehensible to the user. Even by ranking by sentence length we get similar results with little diversity in topics from the query. In addition, we return sentences we considered as facts that are in close proximity to the original matching sentence, i.e., $k = 2$ in the preceding and the succeeding two sentences. This is based on the locality principle; we assume related information is closer in proximity on a page.

Based on studies by [11] 56% of news report articles were judged to be objective. We consider k small since we only wish to return relevant results that the user will have time to read. Finally, documents are ranked by the number of sentences that match the query. We assume matched sentences are a good indicator of document relevance to the user. If more sentences chance the answer will be in those sentences or in close proximity.

4. Methodology

We performed crawling on the CNN website extracting news article web pages for use in our corpus. The corpus is 120 MB in size, consisting of 4756 documents. The articles span from November 6th to December 3rd, 2009. This is a reasonable corpus size for experimentation of our system, as it pertains only to reputable news sites and not the entire Internet.

The experiments conducted consisted of testing the News Fact Finder with a set of 26 queries, based on 5 different articles from different topic areas found in the corpus. The topics included politics, health, entertainment, sports, and opinions. Four judges were assigned to read the selected articles and classify the sentences as either fact or opinion. We then combined the classifications from all of the judges. There were many ambiguous cases, which were difficult to classify. It was agreed that syntactic analysis of the sentences produces far more valuable results to the user. For example: the user may consider certain expert opinions as facts as some experts may be more experienced, established in the field, or more credible. In such cases, the News Fact Finder leaves the decision up to the user's discretion by including those sentences in the results set.

The News Fact Finder is optimized for sentence level search queries. It returns exact substring matches from the users parsed query. The exact substring matches are then classified as facts or opinions, and the opinions are removed. For the experiment, we consider a sentence to be an opinion if one or more judges classified the sentence as an opinion. In cases where all of the judges classify the sentence as a fact, the sentence is considered a

fact. This method of classification was used in experiments conducted in [4].

We expect that our system will return a relevant results set to the user based on our system's heuristics. It is preferred that more information be provided to the user. As a result, in the cases where doubt exists, we leave it up to the users to interpret whether the sentence is a fact or an opinion.

5. Experimental results

Each query's results were compared to the judges' classification of each sentence in the selected articles. If all of the sentences classified as facts were displayed in the results set, and none of the sentences classified as opinions were displayed to the user, this would be considered a successful result. Any results returned pertaining to other articles that were not classified by the judges were analyzed for subjective keywords. If one or more such words exist in the sentence, this would indicate that the sentence is an opinion and therefore should not be displayed. This would result in a false positive result. The overall results obtained are very promising. The News Fact Finder returned 19 successful results and achieved a 73.08% success rate in the results returned to the user. This can be seen in Figure 5.1.

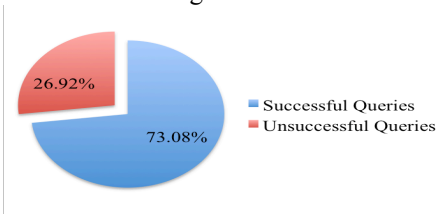

Figure 5.1 Results of queries tested

The total number of queries tested was 26. These 19 successful queries fall into different categories: opinions and facts. Opinion queries tested consisted of a sentence that contained strong subjective words, such as "I think". It is expected that the News Fact Finder would not return any results matching this query, as it should eliminate these sentences and not index them. Fact queries tested resulted in all fact-based sentences being displayed to the user, with none of the opinion sentences displayed from the related article. In Figure 5.2 the number of queries tested from the different categories can be seen.

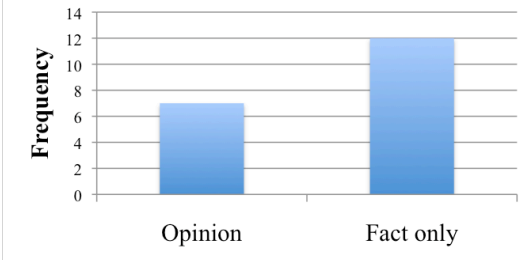

Figure 5.2 Frequency of query types

Of the 7 queries that the News Fact Finder did not return correct results for, there are two types of errors encountered. These errors are false positive and false negative results. The false positive results case includes opinion sentences appearing in the results set, as though they were facts. These sentences were classified as opinions by the judges. The false negative results case is comprised of situations where the subjective word list was too strict and classified facts as opinions. In these cases, some of the expected fact based results were not returned to the user. The frequency of each result type is shown in Figure 5.3.

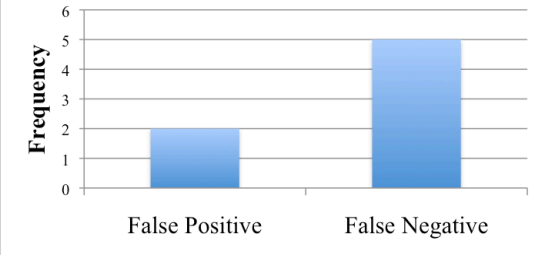

Figure 5.3 Frequency of result types

Our success rate is quite high in comparison to the results achieved by the similar sentence level search engines discussed in section 2. Answer Finder achieved a 26% success rate and Answer Bus achieved a 70% success rate. While the opinion based blog had only achieved judges' consensus on opinions versus facts 22% of the time. Each query was processed in roughly 0.2 seconds on our system. News Fact Finder performed considerably better than AnswerBus, which performs better than other similar systems, has an average query-processing rate of 7 seconds. We feel the response time can be further improved by using more optimal data structures.

One of each of the various types of results obtained from the News Fact Finder is listed in this section. We selected tests that show different cases, and listed them here.

The first query we would like to discuss is "Fantastic Mr. Fox", where no results were returned. This was a successful test as one of the judges classified the sentence containing this substring as an opinion. Therefore News Fact Finder should not display this sentence in the results.

The second query "a Christmas carol", was also a successful test, the results were returned relating to the movie "A Christmas Carol", including previous success, current success, ranking among movies, and a comparison to other debuts in the past. These are excellent results that a user could use to determine whether or not to watch the movie.

The third query run was "French President Nicolas Sarkozy". This was a success, nine facts are returned from three different articles. Each of the facts is relevant to the user's query. Each of the sentences displayed in the

results were all classified as facts by each of the judges.

Another Query run was "I think", no results were returned. All of the judges classified the sentence containing this substring as an opinion; therefore News Fact Finder should not display this result.

6. Conclusions

We have implemented a News Fact Finder System, which conducts sentence level searching through reputable online news media sites. The system also removes opinions and extracts fact based sentences. This is a difficult problem in the context of our system. As we are dealing with only reputable news media web sites, we must also take the style of writing into account. One of the challenges we found was that articles are not written in a fashion where opinions are clearly marked as such, and as such, it can be difficult to classify sentences into opinions or facts.

We have used a subjective word list to classify opinions. Perhaps the subjective list is too strict in some cases, and returns false negatives. This could be avoided altogether with the use of natural language processing in addition to the subjective word list. This would allow for some cases to include the sentence as a fact and exclude it in others, providing more useful results to the user.

7. Future work

There are a few areas of possible future work. We would like to offer user suggestions when few or no results are shown, show the results for corrections to the query such as play off versus playoff. Other areas include changing the ranking, including a list of concepts, and paging similar facts.

A user always expects to read news about the latest happenings of an event. Hence it is best to rank by date. Higher ranking should be given to more recent articles and relatively lower ranking should be given to older articles. In this way it can be made sure that up-to-date information is provided to users, and higher user satisfaction can be obtained.

A list of concepts of verbs can be built and incorporated at the parsing stage. The basic idea behind this is to replace all occurrences of similar verb concepts/senses by a single base verb form. For example, all occurrences of the verb "buy" and its various forms can be replaced by the verb "purchase" both in the documents and in the query. This ensures that all the relevant results to a user's query are retrieved resulting in an increase in the accuracy of the system.

Similar facts from various pages can be grouped together. This aids in providing information from different aspects to the user. For example if a user is querying about "H1N1 vaccination", there are news articles mentioning that the vaccine has had a good effect, and other news articles about the severe negative side effects. Hence if we group similar facts together, we will be able to cover the news from a wider perspective.

8. References

[1] M.S. Siadaty, J. Shu, and W.A. Knaus, "Relemed: sentence-level search engine with relevance score for the MEDLINE database of biomedical articles," *BMC medical informatics and decision making*, BioMed Central Ltd, London, United Kingdom. 2007, vol. 7, no. 1, 2007, pp. 1.

[2] Z. Zheng, "AnswerBus question answering system," *Proceedings of HLT 2002, Second International Conference on Human Language Technology Research*, 2002, San Francisco, CA, USA, pp. 399-404.

[3] M.A. Greenwood, "AnswerFinder: Question Answering from your Desktop," *Proceedings of the 7th Annual* Research *Colloquium* of the *UK, Special Interest Group for Computational Linguistics.* January 6-7, 2004, Birmingham, UK, pp. 75-80.

[4] O. Furuse, N. Hiroshima, S. Yamada, and R. Kataoka, "Opinion sentence search engine on open domain blog," *In Proc. of the 20th Int'l Joint Conf. on Artificial Intelligence (IJCAI-07)*, January 2007, Hyderabad, India, pp. 2760-2765.

[5] V. Makinen, and G. Navarro, "Succinct suffix arrays based on runlength encoding," *Lecture Notes in Computer Science*, 2005, Finland, vol. 3537, 2005, pp. 45-56.

[6] E. M. McCreight, "A space economical suffix tree construction algorithm," *Journal of the ACM (JACM)*, 1976, vol. 23, no. 2, pp. 262-272.

[7] E. Ukkonen, "On-line construction of suffix trees," *Algorithmica*, 1995, vol. 14, no. 3, pp. 249-260.

[8] U. Manber, and G. Myers, "Suffix arrays: A new method for on-line string searches," *Society for Industrial and Applied Mathematics*, Philadelphia, 1989, PA, USA, pp. 319-327.

[9] J. Wiebe, T. Wilson, and M. Bell, "Identifying collocations for recognizing opinions," *Proceedings of the ACL 2001 Workshop on Computational Natural Language Learning (ConLL)*, July 6-7, 2001, Toulouse, France, pp. 24-31.

8. Acknowledgements

We would like to acknowledge Natural Sciences and Engineering Research Council (NSERC) of Canada, for funding this work.