MANYASPECTS: A System for Highlighting Diverse Concepts in Documents

Kun Liu
IBM Almaden Research
San Jose, CA
kun@us.ibm.com

Evimaria Terzi
IBM Almaden Research
San Jose, CA
eterzi@us.ibm.com

Tyrone Grandison
IBM Almaden Research
San Jose, CA
tyroneg@us.ibm.com

ABSTRACT

We demonstrate MANYASPECTS – a document-summarization system that ingests a document and automatically highlights a small set of sentences that are expected to cover the different aspects of the document. The sentences are picked using simple coverage and orthogonality criteria. With MANYASPECTS, you get a concise yet comprehensive overview of the document without having to spend lots of time drilling down into the details. The system can handle both plain text and syndication feeds (RSS and Atom). It can run either as a stand-alone application or be integrated with Web 2.0 forums to pinpoint different opinions on online discussions for blogs, products, movies, etc. For comparative analysis and exploratory flexibility, the system includes other off-the-shelf text-summarization methods, e.g. k-median clustering and singular value decomposition. Thus, the system allows the user to explore the content of the input document in many different ways.

1. INTRODUCTION

The problem of identifying the gist of a document is conventionally referred to as the *text summarization* or *document summarization* problem. Traditional document-summarization techniques [5] mostly focus on the central idea of the text. With the rapid explosion of content on the world wide web, particularly in online text collections, it has become increasingly useful to provide improved mechanisms for identifying the important information themes associated with a text document or a collection of documents.

Consider a RSS feed that contains users' comments about a specific movie. Summarization is necessary since there are movies for which more than tens of thousands of people have written comments. However, the challenge is that there usually does not exist a single central idea in these comments. Each one of these comments often has a different focus because every user looks at the movie from a different angle. Some users comment about the scenery, some about the actors or the director, others about the plot itself, etc. From the reader's perspective, going through all the reviews is a tedious and often annoying task. At the same time, it is useful (and often necessary) to get a quick idea of what other moviegoers think about the movie. This would require a generation of a summary that covers *different aspects* of the comments written by the different users. This scenario brings out a very interesting point - summarization is becoming very important in helping us deal with the information explosion that is currently underway. The significance of this phenomenon is amplified by the fact that the above use case applies to many other domains and applications, e.g. when someone is reading online comments and discussions following blogs, videos and news articles.

In this demonstration, we showcase MANYASPECTS – a novel document-summarization system that can identify different aspects of a document. Our system generates summaries by reusing sentences that already exist in the input document. Therefore, the problem becomes that of picking the right sentences from the original document so that these sentences can capture different viewpoints. This requirement is further refined by the following two criteria:

a) *Coverage* – The summary should consist of sentences that span a large portion of the spectrum of aspects discussed in the document.

b) *Orthogonality* – Each sentence in the summary should capture different aspects of the document. That is, the sentences in the summary should be as orthogonal to each other as possible.

MANYASPECTS provides summaries that satisfy both criteria. In our demonstration, we will illustrate this by showing summaries for 1) standard documents: e.g., news articles, emails, scientific papers; and 2) syndication feeds (RSS and Atom): e.g., reviews for movies, cars; discussions following blogs and news articles, etc. In the latter case, comments of users are merged to construct a large document, which is then summarized to extract the general *feeling* of all the comments.

Our system runs either as a stand-alone application or as an add-on for web browsers. It also serves as a extendable platform that can host many other existing summarization algorithms such as clustering and singular value decomposition. In particular, we show that a new method for summarizing documents, based on a combinatorial formulation, performs significantly better and gives intuitive and naturally interpretable summaries.

The rest of the paper is organized as follows. Section 2 describes the architecture of MANYASPECTS. Section 3 presents a list of summarization techniques. Section 4 illustrates how we are going to demonstrate the system.

2. SYSTEM ARCHITECTURE

Figure 1 depicts the architecture of MANYASPECTS. The entire system consists of three modules: 1) the user interface module is responsible for user interactive operations and visualization, including the *GUI* and *highlighter* components; 2) the parser module

VLDB '08, August 24-30, 2008, Auckland, New Zealand

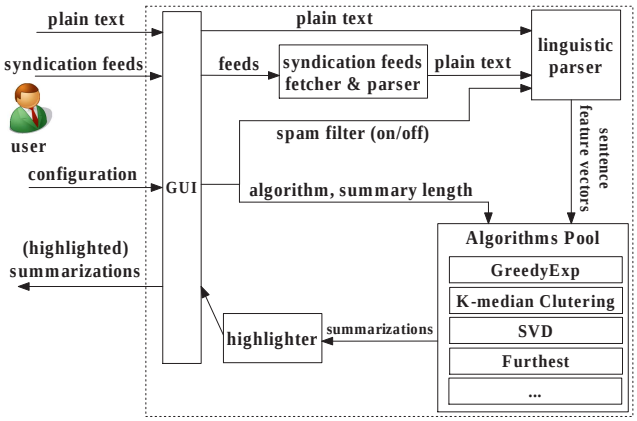

Figure 1: Architecture of MANYASPECTS.

provides support for parsing and filtering syndication feeds (RSS and Atom) and plain text, including the *syndication feeds fetcher and parser* and *linguistic parser* components; and 3) the *algorithms pool* contains different document-summarization algorithms.

User Interface: The *GUI* takes as inputs the file name, syndication feeds and user configuration (summarization algorithm, summary length, spam filter on/off), then passes them to the *parser* and *algorithm pools* to generate the summary. It delivers the final results to the user in two ways: 1) uses the *highlighter* component to mark up the background of the extracted sentences in the original document with colored areas so that one can easily locate them and identify the context; and 2) displays the extracted sentences in a separate panel for a quick review.

Parser: Given a URL pointing to a syndication feed, the *syndication feeds fetcher and parser* component downloads the feed, extracts the title, author, published date, description and other related information and combines them to a single document in plain text. The *linguistic parser* processes the plain text to determine the vocabulary of terms the summarization system uses; it also applies spam filtering techniques to remove spam inputs. More specifically, this parser performs the following operations: sentence segmentation, words tokenization, removal of stop words, token normalization, stemming, and spam filter [2]. The output of the *linguistic parser* is a set of informative sentences, each represented as a feature vector.

Algorithms Pool: Having constructed the sentence feature vectors, the system computes the summary using the algorithm selected by the user in the configuration. The outputs are the indices of the sentences to be extracted. This module is fully extendable. It allows new algorithms to be easily integrated without having to re-compile other modules.

Running Mode: The system runs either as a stand-alone application or as an add-on of Firefox browser. It could also be integrated with Web 2.0 forums as a background process to identify different aspects of online discussions for blogs, products, etc. When running as a stand-alone application, the system can summarize documents stored in a local file system; it can also handle syndication feeds pointed by URLs which are typed in the GUI by the user. When it is running as an add-on, a user can simply click a a specially added button on the Firefox toolbar, the syndication feeds in the page will be passed to our system and the summaries are generated and delivered to the user.

Next, we describe the summarization techniques we have implemented in the system.

3. SUMMARIZATION TECHNIQUES

As previously mentioned, the main idea behind our summarization technique is that we construct summaries by picking sentences that already exist in the input text document. More specifically, our summary of a document consists of at most k sentences coming from the input document. The number of sentences k is an input parameter to our system and it captures the user requirement of small (small values of k) versus larger (large values of k) summaries.

In this section we provide a formal definition of the summarization problem that exactly captures the two requirements described in Section 1. We also develop a new algorithm for solving this problem and demonstrate its usefulness with a simple but illustrative example document. Finally, we briefly describe alternative algorithmic solutions to the document-summarization problem. We have implemented all these alternative solutions and in our demonstration we will illustrate their capabilities (and their weaknesses when compared to our proposed approach).

3.1 Problem Definition

Conceptually, every document is represented by a matrix D, such that the rows of D correspond to words and the columns of D correspond to sentences. That is, $D(i, j)$ is the number of appearances of word i in sentence j. (Since we remove stop words and spam words, the entries of D should be expected to take values 0 or 1). Therefore, if a document consists of n sentences and m unique words (after the removal of stop words and spam words) then matrix D is of size $m \times n$. Let W denote this set of unique words. The problem of summarizing document D using at most k sentences can be formally defined as follows:

PROBLEM 1. *Given matrix D and an integer k, find S, subset of the columns of D with $|S| \leq k$ such that the weighted coverage of the projection of D on S, $C_f(D[S])$ is maximized.*

Note that the *weighted coverage* $C_f(D[S])$ is computed as follows:

1. Let the S partition the elements in W (rows of matrix D) into $\mathcal{P}(W, S) = \{W_0, \ldots, W_{|S|}\}$, where every word $w \in W_x$ appears in exactly x sentences in S.

2. Then, the *weighted coverage* of summary S is

$$C_f(D[S]) = \sum_{x=0}^{|S|} \sum_{w \in W_x} f(x) = \sum_{x=1}^{|S|} \sum_{w \in W_x} f(x).$$

In this paper we use two different forms of the function f.

Uniform f_u: In the case of uniform f we set that $f(0) = 0$ and $f(x) = 1$ for every $x > 0$. The **uniform** function implicitly gives higher weight to coverage than orthogonality.

Exponential f_e: In the case of exponential f we set that $f(0) = 0$ and $f(x) = \frac{1}{2^{x-1}}$ for $x > 0$. The **exponential** function imposes a relatively stronger requirement for orthogonal summaries.[1]

[1]Constant 2 in the denominator can in principle be replaced by any other constant. In practice we have observed that for constants in the range $[2, 4]$ the results appear to be identical.

k	Concept
1	i) Blackberry Cellphones stopped working
2	ii) RIM Network problems
3	v) Google owns Performics
4	iii) People's reliance on Blackberry
5	iv) Google acquired DoubleClick

Table 1: New concept identified by Greedy algorithm with respect to k.

Due to space constraints, an in-depth analysis of this definition is beyond the scope of this document. However, we note that the requirement of maximizing $C_f(D[S'])$ as defined above encourages the selection of sentences that are as orthogonal as possible, and at the same time they cover as many of the words (representatives of different concepts) as possible. Function f controls the strictness of the requirements for coverage and orthogonality.

3.2 The Greedy Algorithm

We solve Problem 1 using the **Greedy** algorithm. The algorithm operates in rounds and it greedily picks columns of D.

That is, it starts by picking the column (sentence) s_1 such that $C_f(D[s_1])$ is maximized. Then, among all the remaining columns of D it proceeds by picking sentence s_2 such that the marginal gain in the weighted coverage by adding this sentence to the existing summary is maximized. This process continues until k sentences are picked or if no sentence with a positive marginal gain remains.

3.3 Illustration of Greedy

As an example, we choose two news articles to demonstrate the performance of the Greedy algorithm for f being the exponential function f_e. The first article titled "BlackBerry hit by major disruption in US, Canada" is chosen from Yahoo! News, [2] and the second one titled "Google Now Selling SEO Services Via Performics" is from TechCrunch.com. [3]

We deliberately extract the first paragraph (which has only two sentences) of the Google article and insert it to a random place in the BlackBerry article (which has thirteen sentences). A judicious analysis shows that the new document contains the following five key points: i) BlackBerry cellphones in North America stopped working; ii) the problem was due to the RIM network that handles wireless data transmission, but not the telephone network; iii) people's reliance on BlackBerry is fierce; iv) Google has successfully acquired DoubleClick; and v) Google now owns SEO service Performics. The first three concepts are from the BlackBerry article, while the last two are from the first paragraph of Google article.

For the experiment, we change the value of k from 1 to 5; each time a new sentence is extracted and added into the summary. Table 1 shows how a new concept is identified sequentially with respect to k. When $k = 1, 2$, the summary covers concepts i) and ii), and this is potentially due to the fact that the majority of the document is about BlackBerry, and therefore it is more likely to select a long sentence about BlackBerry in the summary since such a sentence is expected to give a higher gain. When $k = 3$, the system identifies a new concept v), which is from the Google article. A snapshot of the system in that state is shown in Figure 2. When $k = 4, 5$, concept iii) is first located followed by concept iv).

3.4 Other summarization techniques

In this section we describe some alternative solutions to the document summarization problem. In our system, we have implemented each one of those solutions.

Clustering: Each sentence s_i corresponds to an m-dimensional vector over the space of words. Picking k sentences that are representative of the document and diverse of each other, can also be considered as a *clustering* problem: the sentences can be clustered in k clusters and the centroid of each cluster can be considered as the representative of the sentences in the cluster. In our implementation we use standard k-median algorithm combined with the cosine similarity for measuring the similarity between sentences.

SVD: The requirement of orthogonality between the sentences in the summary suggests that maybe the column basis found by doing a Singular Value Decomposition (SVD) on the words-sentence matrix D could be used for solving the document-summarization problem. We have implemented this approach in our system as follows: First we find an orthonormal basis for the columns of the matrix using classical SVD algorithm. Then we use the k basis vectors that correspond to the k largest singular values as indicative representative sentences. The problem with these k vectors is that they do not correspond to any of the original sentences in the document. However, using a simple greedy algorithm we match each one of them to the closest column vector from the original matrix D.

Furthest: This is a simple heuristic that is motivated by the *furthest* algorithm for solving the k-center problem. In this case, a summary of k sentences is constructed as follows: we first start with a random sentence of the document and add it to the summary. We then pick the sentence that is the most dissimilar to the ones that have been already picked. We use one minus the cosine similarity as the measure of dissimilarity between sentences. The distance between a single sentence and a set of sentences is defined as the sum of the distances between the former and every element in the set.

Related Work: For completeness, we describe some related work in the literature.

The idea of picking actual sentences to summarize a document is not new. There is a body of work in the Information Retrieval (IR) literature in defining score functions for sentences and picking the sentences with the highest score as summaries [3]. The emergence of the Web has also inspired the text mining community to create methods for automatically summarizing Web sites [1], for snippet generation [7] and for identifying important blocks of web pages [6]. Finally, opinion extraction [4] has been used as a form of summarization for product reviews. However, this technique often pre-selects a set of product features, and classifies reviews as either positive or negative, or as either subjective/opinion or objective/fact.

4. DEMONSTRATION OVERVIEW

In this demonstration we will illustrate MANYASPECTS working either as a *stand-alone* application or as an *add-on* for a web browser. Depending on the running mode, different types of documents will be summarized and analyzed.

Stand-alone Application: In this case (shown in Figure 2), the user is asked to pick either a text file from the local file system, or a syndication feed by typing the URL in the URL/URI field. After the document is rendered, the user is prompted by the interface to pick the degree of summarization, i.e., how many sentences he/she wants to appear in the summary, i.e., k. The value of k is controlled by a slider in the GUI, where the tick label 100% means all the sentences in the original documents should be in the summary and 0% means none. The user is also required to select the algorithm to

[2] http://ca.news.yahoo.com/s/afp/080212/canada/technology_us_canada_it_telecom_company_rim

[3] http://www.techcrunch.com/2008/03/12/google-now-selling-seo-services-via-performics/

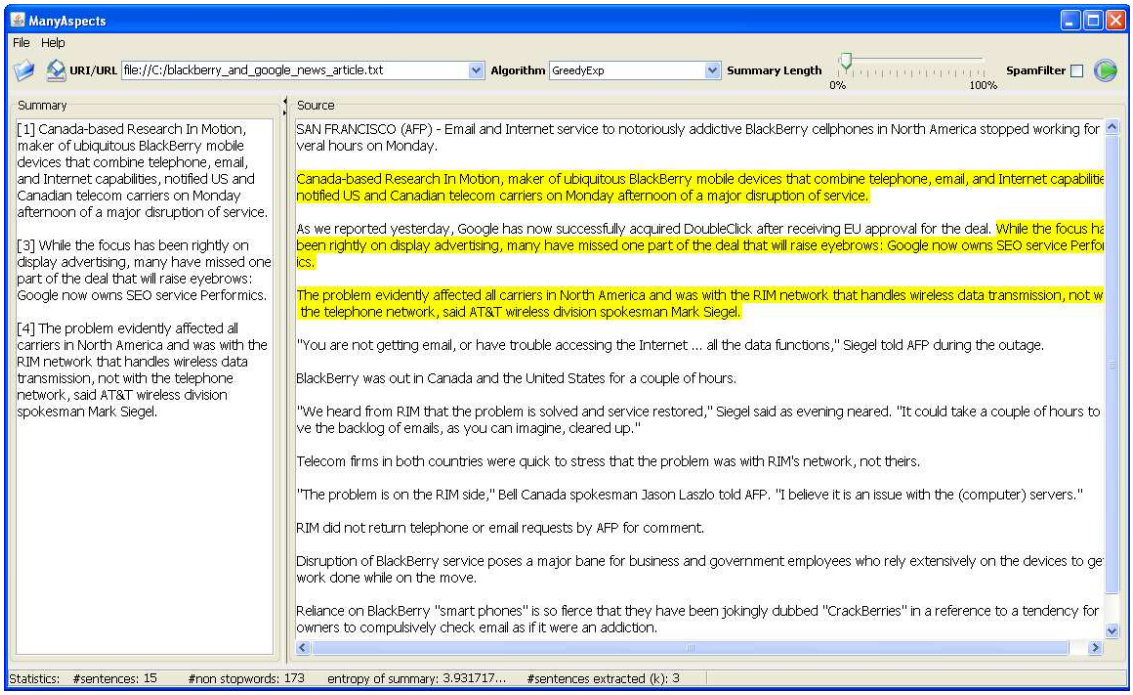

Figure 2: Snapshot of MANYASPECTS.

be used for summarization. The algorithm options are **Greedy** (our new algorithm), **Clustering**, **SVD** and **Furthest**. By pressing the execution button the summarization is executed and the sentences picked in the document are highlighted in the original text and also displayed separately.

We will demonstrate this version of our system using *scientific papers*, *news articles* and *personal emails*.

Add-on for Web Browser: In this case (shown in Figure 3), we will show how our system can be integrated into Firefox as an add-on. We will demonstrate how a specific web page (in our case a syndication feed such as RSS and Atom) on a browser can be summarized. Pressing a specially added button on the Firefox toolbar, the URL and/or content of the page is passed to our system, and the summary of that page is highlighted and printed in the system GUI. The user options here are the same as in the previous setting. The user can further enable (or disable) the spam-controlling option since spams are common in online reviews, discussions and comments.

We will demonstrate this version of our system using *syndication feeds* from websites such as `www.techcrunch.com` that has user comments on blogs and news.

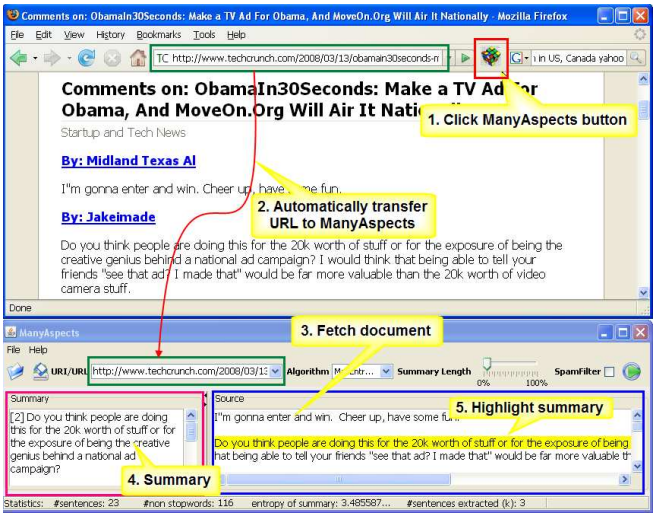

Figure 3: MANYASPECTS running as a Firefox add-on.

Acknowledgments: We want to thank Heikki Mannila for his useful discussions.

5. REFERENCES

[1] E. Amitay and C. Paris. Automatically summarising web sites - is there a way around it? In *CIKM*, pages 173–179, 2000.

[2] P. R. Christopher D. Manning and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[3] J. Goldstein, M. Kantrowitz, V. O. Mittal, and J. G. Carbonell. Summarizing text documents: Sentence selection and evaluation metrics. In *SIGIR*, pages 121–128, 1999.

[4] M. Hu and B. Liu. Opinion extraction and summarization on the web. In *AAAI*, 2006.

[5] I. Mani and M. T. Maybury. *Advances in Automatic Text Summarization*. MIT Press, 1999.

[6] R. Song, H. Liu, J.-R. Wen, and W.-Y. Ma. Learning block importance models for web pages. In *WWW*, pages 203–211, 2004.

[7] A. Turpin, Y. Tsegay, D. Hawking, and H. E. Williams. Fast generation of result snippets in web search. In *SIGIR*, pages 127–134, 2007.