A Web Page Summarizer For Mozilla

Bhandari Harendra(harendra-b@is.naist.jp)
Computational Linguistics Lab
Nara Institute of Science and Technology(NAIST)

Takahiko Ito(takahi-i@is.naist.jp)
Computational Linguistics Lab
Nara Institute of Science and Technology(NAIST)

Masashi Shimbo(shimbo@is.naist.jp)
Computational Linguistics Lab
Nara Institute of Science and Technology(NAIST)

Yuji Matsumoto(matsu@is.naist.jp)
Computational Linguistics Lab
Nara Institute of Science and Technology(NAIST)

Abstract

While browsing the web pages, users are often forced to read through whole page to get the information from the page. The pages the user reads may not necessarily contain useful information. In this paper we present a software program designed to facilitate the browsing process for such users by giving them the ability to preview the summary of web pages and decide whether to read through the complete text or not. This program uses Mozilla Firefox as a platform and can be installed as an extension (add-on) of the browser. Once installed, users can right click any link and get the summarized text of the linked page. The program uses an algorithm called LexPageRank to summarize the web page.

1. Introduction

The Web has already become the prime source of information for a significant amount of people in the world, but the plethora of information in web provides users with a challenge to find the most relevant piece of information from a sea of information. Although, the search engines do a commendable job to help users find desired information, the amount of information generated by the search engine itself is quite large. Users generally scan through the search results manually until the satisfactory information is found. It will be greatly useful to the users browsing through a heap of web pages if they are presented with the summary of pages they plan to read through. The user can then scan through the large collection of documents more efficiently.

Summarization is an important field of Natural Language Processing and is a field of active research in recent years. Summarization can be classified as abstract-based and extract-based. In extract-based summarization, the summarization is performed by extracting text from the document. The summarized text does not contain any text outside the given text whereas the abstract-summary might employ the words that do not appear in the original document [Mani 99]. The LexPageRank which we deployed in this work is an extract based summarization.

The summarization task can also be categorized as either generic or query-oriented. The query oriented summary is the one where the summarized text is most relevant to the given query text. Generic summarization aims at giving overall sense of the document and the widest range of possible topics are included in the summarized text. [Goldstein et al 1999] present such a method.

In this paper we propose a software program that provides user with an option to preview summary of the web page. We have used a popular open source web browser, Mozilla Firefox, as the platform. Our software program is actually an add-on of the Mozilla Firefox browser. To our knowledge none of the browsers provide such a feature and is neither available as a third-party program. Integrating a software program like ours in the web browser is useful to the users because the summarization can be performed in real time as the user browses through the web pages. All the user needs to do is right click the link and click on the option presented on context menu (Figure 1 & Figure 2).

Our summarization algorithm is based on an algorithm

called LexPageRank [Erkan and Radev, 2004]. Lexpagerank is an extraction-based summarization algorithm, which uses PageRank [Brin and Page 1998] to find most important sentence in a document. LexPageRank will be discussed in section 3.

2. Related Work

Extractive summarization has mostly been based on scoring the sentences in the source document based on set of predefined features [Mani and Boledorn, 1998]. Lexpagerank is one of the extractive summarization procedures.

A method to generate generic summarizer is Maximal Margin Relevance (MMR) [Carbonell et al 1997]. According to MMR, a sentence is chosen for inclusion in summary such that it is maximally similar to the document and dissimilar to the already selected sentences. This approach works in ad hoc manner and tends to select long sentences.

In [Gong and Liu 2001] the authors observed that hidden topics can be discovered in a document as well as the projection of each sentence on each topic through Latent Semantic Analysis [Deerwester et al, 1990]. They selected the sentences which have the large projections on the salient topics to form the summary. In Mihalcea's work [Mihalcea 2005], the author constructed a graph in which each node is a sentence and the weight of the edge linking two of the nodes is the similarity between the corresponding sentences. The direction of edges can be determined by the order of appearance of sentences. After the graph was constructed the ranking algorithms such as HITS and PageRank were applied on the graph to rank the sentences.

3. LexPageRank

LexPageRank is an extractive summarization procedure. A document is represented as a graph where vertices represent the sentences present in the document and the edges are defined as the similarity measure between the sentences. The similarity measurement generally used is the cosine similarity between the sentences. The nodes (sentences) are ranked based on the fact that the most important sentence will have strong similarity with many nodes in the graph. Before starting the ranking procedure a threshold value **φ** for the similarity is set. If the similarity measure between the two sentences is less than the threshold value the edge between those two sentence nodes is removed, the edges will be retained otherwise. The ranking is done using PageRank algorithm. PageRank has been successfully been used by Google Inc. to rank the web pages using the hyperlinked structure of web pages. PageRank can be stated in equation 1.

$$PR(A) = (1-d) + d\left(\frac{PR(T_1)}{C(T_1)} + ... + \frac{PR(Tn)}{C(Tn)}\right) \quad (1)$$

where T_1…..T_n are number of links that link to A, $C(T_n)$ is number of outgoing links from page T_n and d is the dampening factor. The value of dampening factor is between 0 and 1. For this setup value of d was set to 0.85 as suggested by Brin and Page [Brin and Page 1998].

The procedure to calculate the prestige of sentences with PageRank can be listed in Algorithm 1. The prestige of a sentence is defined as the score given by a ranking algorithm (this case PageRank) to the sentence. Higher the score higher the prestige is.

```
1   M: An array of n sentences in the document.
2   Array AdjacencyMatrix[n][n]
3   φ : Threshold value, between 0 and 1
4   for i ← 1 to n do
5       for j ← 1 to n do
6           if(AdjacencyMatrix [i][j]>= φ) then
7               AdjacencyMatrix[i][j]=1
8           end
9           else
10              AdjacencyMatrix[i][j]=0
11          end
12      end
13  end
14 R=CalculatePageRank(AdjacencyMatrix)
15 return R
```

Algorithm 1. Algorithm for calculating rank of the sentences present in the document. PageRank is calculated using equation 1. Equation 1 is an iterative procedure. Initially PR(A) for all the nodes is set to a constant value (typically 1) and the iteration is done for all the nodes. The number of iterations sufficient for the values PR(A) to converge is about 80-100.

4.Mozilla Web Page Summarizer

We used Mozilla Firefox as a platform to develop the summarizer for web pages. Firefox was chosen mainly because of its feature that lets a developer add his/her own extension (add-on). Once installed the extension becomes a functionality of the browser. Mozilla provides this feature with technologies called XPCOM and XPConnect.

The steps involved during the summarization process on the browser are listed below.

1. The web page to be summarized is downloaded and loaded into the memory.
2. The web page is then parsed to extract text.
3. The sentences in the extracted text are represented as a graph where vertices represent the sentences in extracted text.
4. Algorithm 1 is then applied on the graph.
5. The 25% of the top ranked sentences was used as summarized text.

We used Javascript as the programming language to program our summarizer. Javascript is a standard language used to program add-ons for Mozilla. In our test computer (AMD Athlon 3200+, 2.0Ghz) it generally took a few seconds to produce the summarized text for most of the pages. We used news sites (BBC, CNN etc) to test our summarizer. The overall time can be classified as follows.

1. Page download time: The amount of time required to download the page. This time can vary with the kind of internet connection.
2. Text Processing time: The time required to parse the HTML/XML file to extract text and create document matrix. The complexity of calculation is $O(n^2)$.
3. PageRank calculation time: The amount of time required to calculate PageRank for each sentence.

So the amount of time taken by our summarizer depends on the kind of internet connection (faster the better) and the size of web page.

The screenshots(figure 1 and figure 2) show how the program works in the browser.

5. Conclusions and Future work

In this paper we presented a way to effectively provide users an opportunity to view the summary of web pages they want to visit. The main aspects of this work were the ease with which users could seamlessly integrate an application like this in their browser and use it readily as well. In the future we plan to make the summarizer to handle generic summarization tasks as well.

References

[Mani 1999] Intdrajeet Mani. Advances in Automatic Text Summarization. MIT press, Cambridge, MA, USA, 1999.

[Brin and Page 1998] Sergey Brin and Lawrence Page. The anatomy of the large scale hypertextual search engine.
Computer Networks. 30(1-7):107-117,1998

[Kleinberg, 1999] Jon M. Klienberg. Authoritative sources in a hyperlinked environment. J.ACM, 46(5), 1999

[Mihalcea, 2005] Rada Mihalcea. Language independent extractive summarization. In AAAI, pages 1688-1689, 2005.

[Mani and Boledorn, 1998] Inderjeet Mani and Eric Bloedorn. Machine learning of generic and user-focused summarization. In AAAI/IAAI, pages 820-826, 1998.

[Carbonell et al 1997] Jamie Carbonell, Yibing Geng, and Jade Goldstein. Automated query-relevant summarization and diversity-based ranking. In IJCAI-97 Workshop on AI in digital
Libraries.

[Erkan and Radev, 2004], LexPageRank: Prestige in Multi document summarization. EMNLP 2004

Figure 1. The option to summarize appears in the context menu as “Summarizes link information”. The page pointed by the link being right clicked will be summarized.

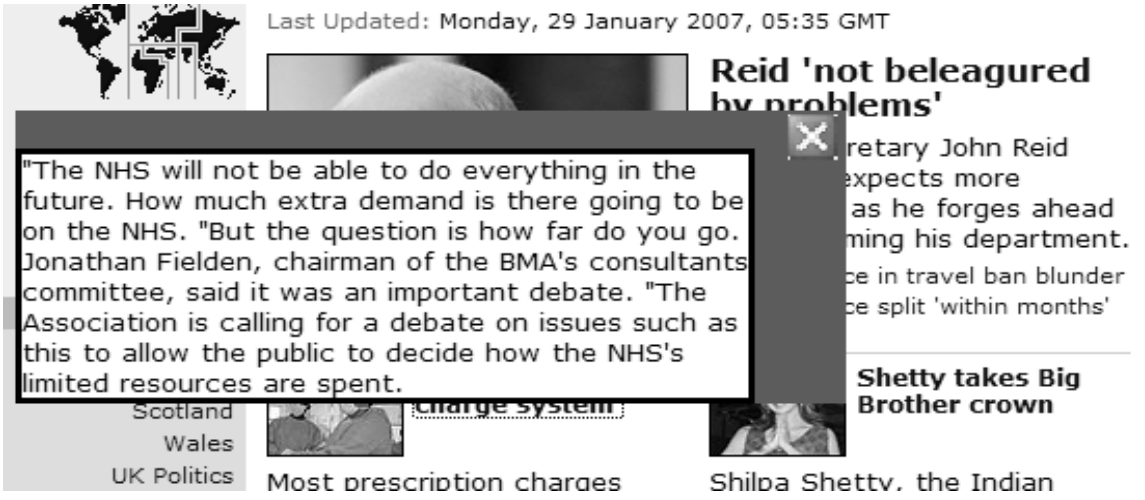

Figure 2. The summarized text appears on the browser after the option “Summarizes link information” is clicked.