Semantic Web Technology Project Report on

Text Summarization

Bhuvan M S (12IT16)

Vinay Rao D (12IT94)

Under the Guidance of,

Mr. Dinesh Naik

Department of Information Technology, NITK Surathkal

Date of Submission: 08 May 2015

in partial fulfillment for the award of the degree

of

Bachelor of Technology

In

Information Technology

At

Department of Information Technology

National Institute of Technology Karnataka, Surathkal

April 2015

Department of Information Technology, NITK Surathkal
Minor Project
End Semester Evaluation Report (April 2015)

Course Code : IT 412

Course Title: Semantic Web Technology

Project Title: *Text Summarization*

Project Group:

Name of the Student	Register No.	Signature with Date
Bhuvan M S	12IT16	
Vinay Rao D	12IT94	

Place: NITK, Surathkal

Date: 08 May 2015 *(Name and Signature of Minor Project Guide)*

Abstract

Text Summarization is condensing the source text into a shorter version preserving its information content and overall meaning. It is very difficult for human beings to manually summarize large documents of text. Text Summarization methods can be classified into extractive and abstractive summarization.In this project, we try out two different extractive summarization techniques. The first method uses standard IR methods to rank sentence relevances, while the second method uses the latent semantic analysis technique to identify semantically important sentences, for summary creations. Both methods strive to select sentences that are highly ranked and different from each other. This is an attempt to create a summary with a wider coverage of the document's main content and less redundancy. First method has more redundancy than the second method. Second method also considers the semantic relations between terms and hence is a better method. Evaluation of text summarization is a challenging task. Here, both the methods are compared to a standard online summarizer.

Keywords: *Generic Text Summarization; Relevance Measure; Latent Semantic Analysis*

Contents

List of Figures

List of Tables

1 Introduction

Text summarization [1] has become an important and timely tool for assisting and interpreting text information in todays fast-growing information age. It is very difficult for human beings to manually summarize large documents of text. There is an abundance of text material available on the internet. However, usually the Internet provides more information than is needed. Therefore, a twofold problem is encountered: searching for relevant documents through an overwhelming number of documents available, and absorbing a large quantity of relevant information. The goal of automatic text summarization is condensing the source text into a shorter version preserving its information content and overall meaning.

A summary [3] can be employed in an indicative way as a pointer to some parts of the original document, or in an informative way to cover all relevant information of the text. In both cases the most important advantage of using a summary is its reduced reading time. A good summary system should reflect the diverse topics of the document while keeping redundancy to a minimum. Summarization tools may also search for headings and other markers of subtopics in order to identify the key points of a document. Microsoft Words AutoSummarize function is a simple example of text summarization. Text Summarization methods can be classified into extractive and abstractive summarization.

An Abstractive summarization[4] attempts to develop an understanding of the main concepts in a document and then express those concepts in clear natural language. It uses linguistic methods to examine and interpret the text and then to find the new concepts and expressions to best describe it by generating a new shorter text that conveys the most important information from the original text document.

Extractive summaries [2] are formulated by extracting key text segments (sentences or passages) from the text, based on statistical analysis of individual or mixed surface level features such as word/phrase frequency, location or cue words to locate the sentences to be extracted. The most important content is treated as the most frequent or the most favorably positioned content. Such an approach thus avoids any efforts on deep text understanding. They are conceptually simple, easy to implement.

2 Literature Survey

2.1 Related Work

Text summarization has been actively researched in recent years. A majority of the research studies in the literature have been focused on creating query-relevant text summaries. M. Sanderson proposed a query-relevant summarizer that divides the document into equally sized overlapping passages, and uses the INQUERY text search engine to obtain the passage that best matches the user's query. This best passage is then used as a summary of the document [5]. A query expansion technique called Local Context Analysis (LCA, which is also from INQUERY) is used before the best passage retrieval. Given a topic and a document collection, the LCA retrieves top-ranked documents from the collection, examines the context surrounding the topic terms in each retrieved document, and then selects and adds the words/phrases that are frequent in this context to the query.

B. Baldwin and T.S. Morton developed a summarizer that selects sentences from the document until all the phrases in the query are covered. A sentence in the document is considered to cover a phrase in the query if they co-refer to the same individual, organization, event, etc [6]. R. Barzilay and M. Elhadad developed a method that creates text summaries by finding lexical chains from the document [7]. The Cornell/Sabir system uses the document ranking and passage retrieval capabilities of the SMART text search engine to effectively identify relevant passages in a document [8]. The text summarizer from CGI/CMU uses a technique called Maximal Marginal Relevance (MMR) which measures the relevance of each sentence in the document to the user provided query, as well as to the sentences that have been selected an added into the summary [9]. The text summary is created by selecting the sentences that are highly relevant to the user's query, but are different from each other.

The SUMMARIST text summarizer from the University of Southern California strives to create text summaries based on the equation: summarization = topic identification + interpretation + generation . The identification stage filters the input document to determine the most important, central topics. The interpretation stage clusters words and abstracts them into some encompassing concepts. Finally, the generation stage generates summaries either by outputting some portions of the input, or by creating new sentences based on the interpretation of the document concepts [10]. However, this generation function was not realized in the paper. The Knowledge Management (KM) system from SRA International, Inc. extracts summarization features using morphological analysis, name tagging and co-reference resolution. They used a machine learning technique to determine the optimal combination of these features in combination with statistical information from the corpus to identify the best sentences to include in a summary.

Yihong Gong and Xin Liu [11] first proposed the extractive summarization by latent semantic analysis inspired by latent semantic indexing. They compared both summarization by relevance measure and latent semantic analysis and found that summarization by lsa was indeed better.

2.2 Objectives

1. Build a Text Summarization model by using relevance measure.
2. In order to accommodate Big Data and Streaming data, implementation in suitable frameworks.

3 Methodology

A document usually consists of several topics. Some topics are described intensively by many sentences, and hence form the major content of the document. Other topics may just be briefly mentioned to supplement the major topics, or to make the whole story more complete. A good generic summary should cover the major topics of the document as much as possible, and at the same time, keep redundancy to a minimum.

In this section, we propose two methods that create generic summaries by selecting sentences based on the relevance measure and the latent semantic analysis. Both methods need to first decompose the document into individual sentences, and to create a weighted term-frequency vector for each of the sentences. Let $T_i = \begin{bmatrix} t_{1i} & t_{2i} & \dots & t_{ni} \end{bmatrix}^T$ be the term-frequency vector of passage i , where element t_{ji} denotes the frequency in which term j occurs in passage i . Here passage i could be a phrase, a sentence, a paragraph of the document, or could be the whole document itself. The weighted term-frequency vector $A_i = \begin{bmatrix} a_{1i} & a_{2i} & \dots & a_{ni} \end{bmatrix}^T$ of passage i is defined as:

$$a_{ji} = L(t_{ji}).G(t_{ji}) \tag{1}$$

where $L(t_{ji})$ is the local weighting for term j in passage i , and $G(t_{ji})$ is the global weighting for term j in the whole document. When the weighted term-frequency vector A_i is created, we further have the choice of using A_i with its original form, or normalizing it by its length $|A_i|$. There are many possible weighting schemes. In the following subsections, the two text summarization methods are described in details.

3.1 Summarization by Relevance Measure

After the given document is decomposed into individual sentences, we compute the relevance score of each sentence with the whole document. We then select the sentence k that has the highest relevance score, and add it to the summary. Once the sentence k has been added to the summary, it is eliminated from the candidate sentence set, and all the terms contained in k are eliminated from the original document. For the remaining sentences, we repeat the steps of relevance measure, sentence selection, and term elimination until the number of selected sentences has reached the predefined value. The operation flow is as follows:

1. Decompose the document into individual sentences, and use these sentences to form the candidate sentence set S .

2. Create the weighted term-frequency vector A_i for each sentence $i \in S$, and the weighted term-frequency vector D for the whole document.

3. For each sentence $i \in S$, Compute the relevance score between A_i and D , which is the inner product between A_i and D .

4. Select sentence k that has the highest relevance score, and add it to the summary.

5. Delete k from S , and eliminate all the terms contained in k from the document. Recompute the weighted term-frequency vector D for the document.

6. If the number of sentences in the summary reaches the predefined value, terminate the operation otherwise, go to Step 3.

Cosine similarity is calculated by the formula :

$$\cos(A, B) = \frac{A.B}{||A||.||B||} \tag{2}$$

3.2 Summarization by Latent Semantic Analysis

Inspired by the latent semantic indexing, we applied the singular value decomposition (SVD) to generic text summarization. The process starts with the creation of a terms by sentences matrix $A = \begin{bmatrix} A_1 & A_2 & \dots & A_n \end{bmatrix}$ with each column vector A_i representing the weighted term-frequency vector of sentence i in the document under consideration. If there are a total of m terms and n sentences in the document, then we will have an $m \times n$ matrix A for the document. Since every word does not normally appear in each sentence, the matrix A is usually sparse.

Given an $m \times n$ matrix A , where without loss of generality $m \geq n$, the SVD of A is defined as [12]:

$$A = U \Sigma V^T \tag{3}$$

where $U = [u_{ij}]$ is an $m \times n$ column-orthonormal matrix whose columns are called left singular vectors= diag $\begin{pmatrix} \sigma_1 & \sigma_2 & \dots & \sigma_n \end{pmatrix}$ is an $n \times n$ diagonal matrix whose diagonal elements are non-negative singular values sorted in descending order, and $V = [v_{ij}]$ is an

$n \times n$ orthonormal matrix whose columns are called right singular vectors. If rank(A)= r , then Σ satisfies

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq \sigma_{r+1} = \ldots = \sigma_n = 0. \tag{4}$$

The interpretation of applying the SVD to the terms by sentences matrix A can be made from two different viewpoints. From transformation point of view, the SVD derives a mapping between the m -dimensional space spanned by the weighted term-frequency vectors and the r -dimensional singular vector space with all of its axes linearly-independent. This mapping projects each column vector i in matrix A , which represents the weighted term-frequency vector of sentence i , to column vector $\psi_i = \begin{bmatrix} v_{i1} & v_{i2} & \ldots & v_{in} \end{bmatrix}^T$ of matrix V^T , and maps each row vector j in matrix A , which tells the occurrence count of the term j in each of the documents, to row vector $\varphi_j = \begin{bmatrix} u_{j1} & u_{j2} & \ldots & u_{jr} \end{bmatrix}^T$ of matrix U . Here each element v_{ix} of ψ_i , u_{jy} of φ_j is called the index with the x'th, y'th singular vectors, respectively.

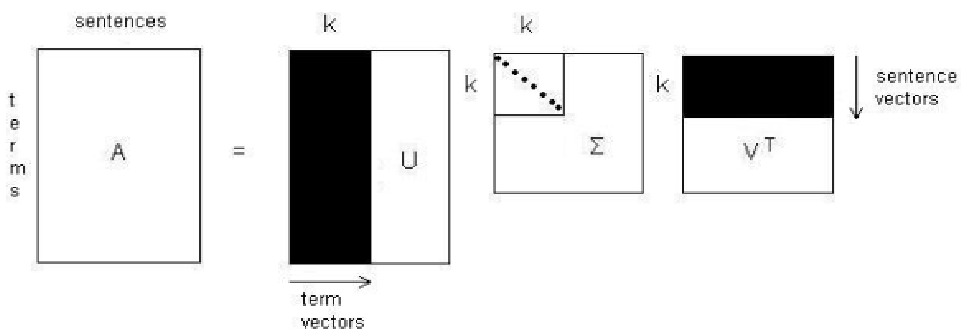

Figure 3.1: SVD Diagram.

From semantic point of view, the SVD derives the latent semantic structure from the document represented by matrix A [13]. This operation reflects a breakdown of the original document into r linearly-independent base vectors or concepts. Each term and sentence from the document is jointly indexed by these base vectors/concepts. A unique SVD feature which is lacking in conventional IR technologies is that the SVD is capable of capturing and modeling interrelationships among terms so that it can semantically cluster terms and sentences. Consider the words doctor , physician , hospital , medicine , and nurse . The words doctor and physician are synonyms, and hospital , medicine , nurse are the closely related concepts. The two synonyms doctor and physician generally appear in similar contexts that share many related words such as hospital , medicine , nurse , etc. Because of these similar patterns of word combinations, the words doctor and physician will be mapped near to each other in the r -dimensional singular vector space. Further- more, as demonstrated in [14], if a word combination pattern is salient and recurring in a document, this pattern will be captured and represented by one of the singular vectors. The magnitude of the corresponding singular value indicates the importance degree of this pattern within the document. Any sentences containing this word combination pattern will be projected along this singular vector, and the sentence that best represents this pattern will have the largest index value with this vector. As each particular word combination pattern describes a certain topic/concept in the document, the facts described above naturally lead to the hypothesis that each singular vector represents a salient topic/concept of the document, and the magnitude of its corresponding singular value represents the degree of importance of the salient topic/concept.

Based on the above discussion, we propose the following SVD-based document summarization method.

1. Decompose the document D into individual sentences, and use these sentences to form the candidate sentence set S , and set $k = 1$.
2. Construct the terms by sentences matrix A for the document D .
3. Perform the SVD on A to obtain the singular value matrix , and the right singular vector matrix V^T . In the singular vector space, each sentence i is represented by the column vector $\psi_i = \begin{bmatrix} \upsilon_{i1} & \upsilon_{i2} & \dots & \upsilon_{ir} \end{bmatrix}^T$ of V^T .
4. Select the k'th right singular vector from matrix V^T .
5. Select the sentence which has the largest index value with the k'th right singular vector, and include it in the summary.
6. If k reaches the predefined number, terminate the operation otherwise, increment k by one, and go to Step 4.

In Step 5 of the above operation, finding the sentence that has the largest index value with the k'th right singular vector is equivalent to finding the column vector ψ_i whose k'th element υ_{ik} is the largest. By the hypothesis, this operation is equivalent to finding the best sentence describing the salient concept/topic represented by the k'th singular vector.

Since the singular vectors are sorted in descending order of their corresponding singular values, the k'th singular vector represents the k'th important concept/topic. Because all the singular vectors are independent of each other, the sentences selected by this method contain the minimum redundancy.

4 Work Done

4.1 System Architecture

Operating System	Microsoft Windows 8 Pro and Ubuntu 14.04 LTS
Processor	Intel(R) Core(TM) i7-3610QM CPU @2.30GHz
RAM	8.00 GB (7.89 GB usuable)
System Type	64-bit OS, x64-based processor

Table 1: System Architecture

4.2 Implementation Details

Python is used for implementing text summarization using the two methods mentioned above. Python is chosen because of its useful libraries and its also dynamic typed language. Python's NLTK library is used. When a document is given as input, the document is preprocessed first. The aim of the preprocessing step is to reduce the dimensionality of the representation space, and it normally includes: (i) stop-word elimination common words with no semantics and which do not aggregate relevant information to the task (e.g., the, a) are eliminated; (ii) case folding: consists of converting all the characters to the same kind of letter case - either upper case or lower case; (iii) stemming: syntactically-similar words, such as plurals, verbal variations, etc. are considered similar; the purpose of this procedure is to obtain the stem or radix of each word, which emphasize its semantics.

All the above steps are done using NLTK's methods. Then the term frequency-inverse sentence frequency matrix i.e a_{ji} is calculated. The whole document's ts-isf score is also calculated by adding all the sentences scores and normalizing it. Next, the cosine similarity measure for each sentence and the document is calculated using numpy library with the given code.

```
cosine_function = lambda a,b : round(np.inner(a, b)/(LA.norm(a)*LA.norm(b)),3)
```

Now, cosine similarity score scores are sorted and the top k sentences are chosen and then rearranged in the original order. This is the required summary.

For the implementation of Latent Semantic Analysis method, a python library sumy is used. The same preprocessing steps done for the above method is applicable here also. Then the document is parsed using sumy's plaintext parser. Its output is fed into LSASummarizer's input with the k parameter. The output is a list with the required summary.

The web-app is implemented using python's Flask library. Flask is a microframework for Python based on Werkzeug, Jinja 2 and its very easy to set-up. The front-end is designed with the help of twitter's bootstrap based theme.

```
from flask import Flask, render_template, g, request
from document_generator import *
from lsa_summary import *

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/', methods=['POST'])
def show():
    inp = request.form["text"]
    summary = get_summary(inp, 10)
    #summary = get_lsa_summary(inp, 10)
    return render_template('index.html', text=summary)

if __name__ == '__main__':
    app.run()
```

5 Results and Analysis

Here are some of the screen caps of the web-app and its sample results.

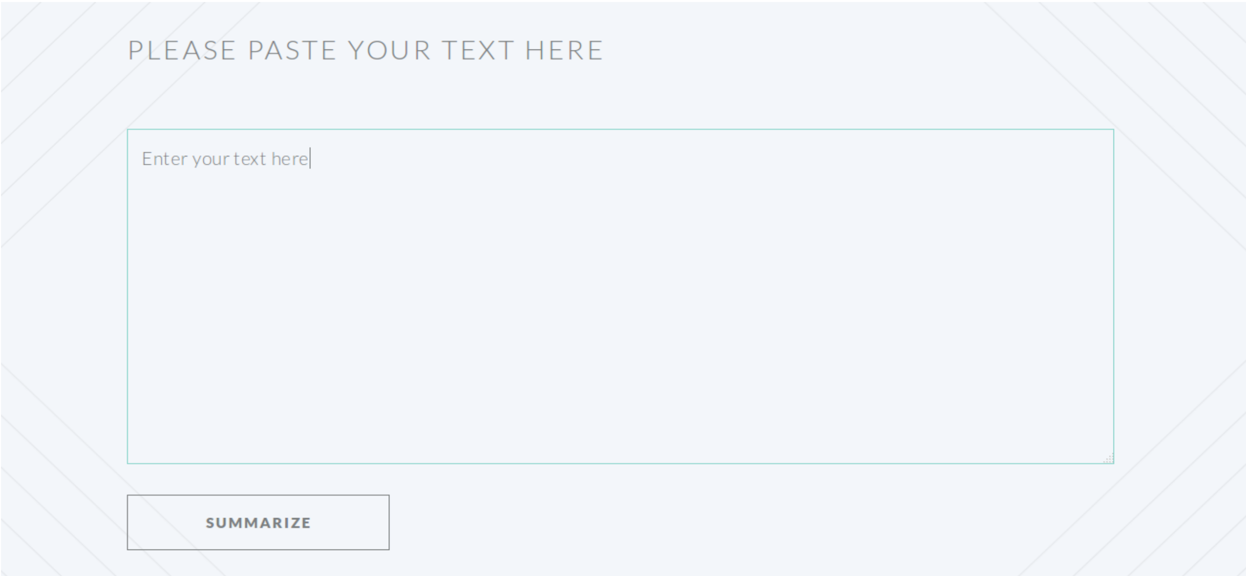

Figure 5.1: Text Summarizer web app.

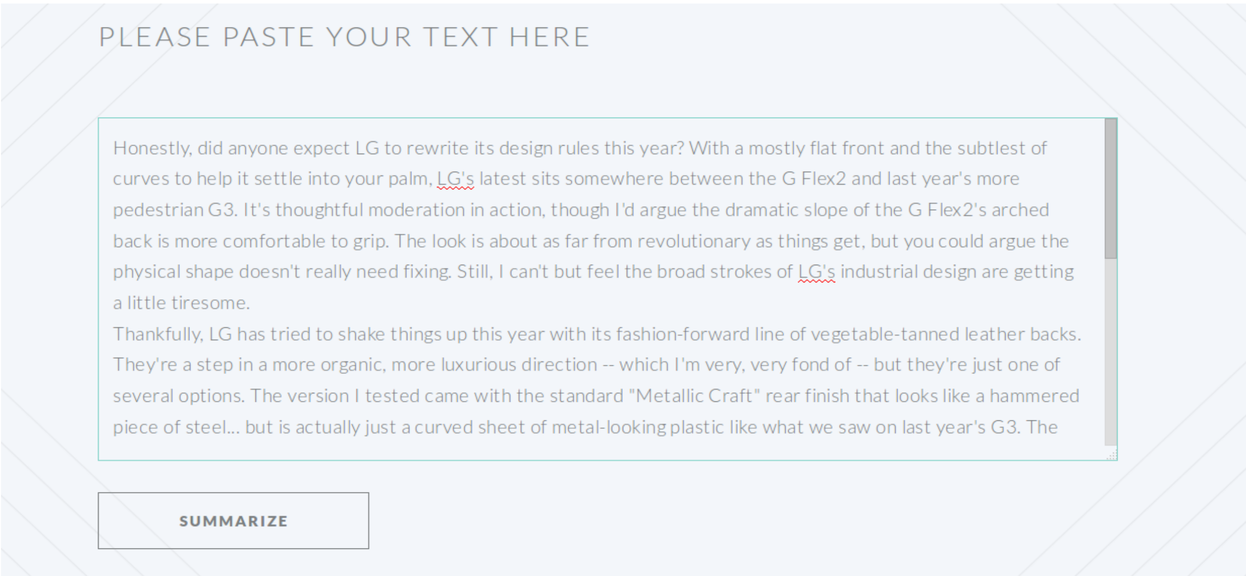

Figure 5.2: Sample Input

Generic text summarization and its evaluation are very challenging. Because no query nor topic are provided to the summarization task, summarization outputs and performance judgments tend to lack consensus. To summarize a document, some people strive to select sentences that maximize the coverage of the document's main content, while

SUMMARIZED TEXT

Honestly, did anyone expect LG to rewrite its design rules this year? With a mostly flat front and the subtlest of curves to help it settle into your palm, LG's latest sits somewhere between the G Flex2 and last year's more pedestrian G3. It's thoughtful moderation in action, though I'd argue the dramatic slope of the G Flex2's arched back is more comfortable to grip. The look is about as far from revolutionary as things get, but you could argue the physical shape doesn't really need fixing. Still, I can't but feel the broad strokes of LG's industrial design are getting a little tiresome. Thankfully, LG has tried to shake things up this year with its fashion-forward line of vegetable-tanned leather backs. They're a step in a more organic, more luxurious direction -- which I'm very, very fond of -- but they're just one of several options. The rest of the G4's posterior is an exercise in repetition. Just below all of that is LG's signature volume rocker/power button combo, except this time, the power button is actually a hair smaller, making it a little tougher to find by feel. The G4's face looks downright spartan compared to its rump, and LG plans to keep it that way -- Dr. Ramchan Woo, LG's head of smartphone planning, stressed the importance of crafting a distinct identity for LG phones, and that means these dark, monolithic faces aren't going anywhere yet.

Figure 5.3: Summarization by Relevance measure.

SUMMARIZED TEXT

Honestly, did anyone expect LG to rewrite its design rules this year?With a mostly flat front and the subtlest of curves to help it settle into your palm, LG's latest sits somewhere between the G Flex2 and last year's more pedestrian G3.It's thoughtful moderation in action, though I'd argue the dramatic slope of the G Flex2's arched back is more comfortable to grip.Still, I can't but feel the broad strokes of LG's industrial design are getting a little tiresome.Thankfully, LG has tried to shake things up this year with its fashion-forward line of vegetable-tanned leather backs.The version I tested came with the standard "Metallic Craft" rear finish that looks like a hammered piece of steel... but is actually just a curved sheet of metal-looking plastic like what we saw on last year's G3.Just like the last two powerhouse phones the company churned out, the rear camera (now boasting 16 megapixels) sits high on the back, flanked by a two-tone LED flash on the right and the infrared autofocus module on the left.The G4's face looks downright spartan compared to its rump, and LG plans to keep it that way -- Dr. Ramchan Woo, LG's head of smartphone planning, stressed the importance of crafting a distinct identity for LG phones, and that means these dark, monolithic faces aren't going anywhere yet.The 5.5-inch IPS Quantum display deserves a lot more verbosity than I should muster in this section, but know this: It's easily among the best smartphone screens I've ever seen, despite what I may have said in the past.When it's off, though, it's scarcely distinguishable from the dark gray bezels that surround it, making the teensy speaker grille, 8-megapixel camera and LG logo the only things that break up the dusky monotony.

Figure 5.4: Summarization by LSA.

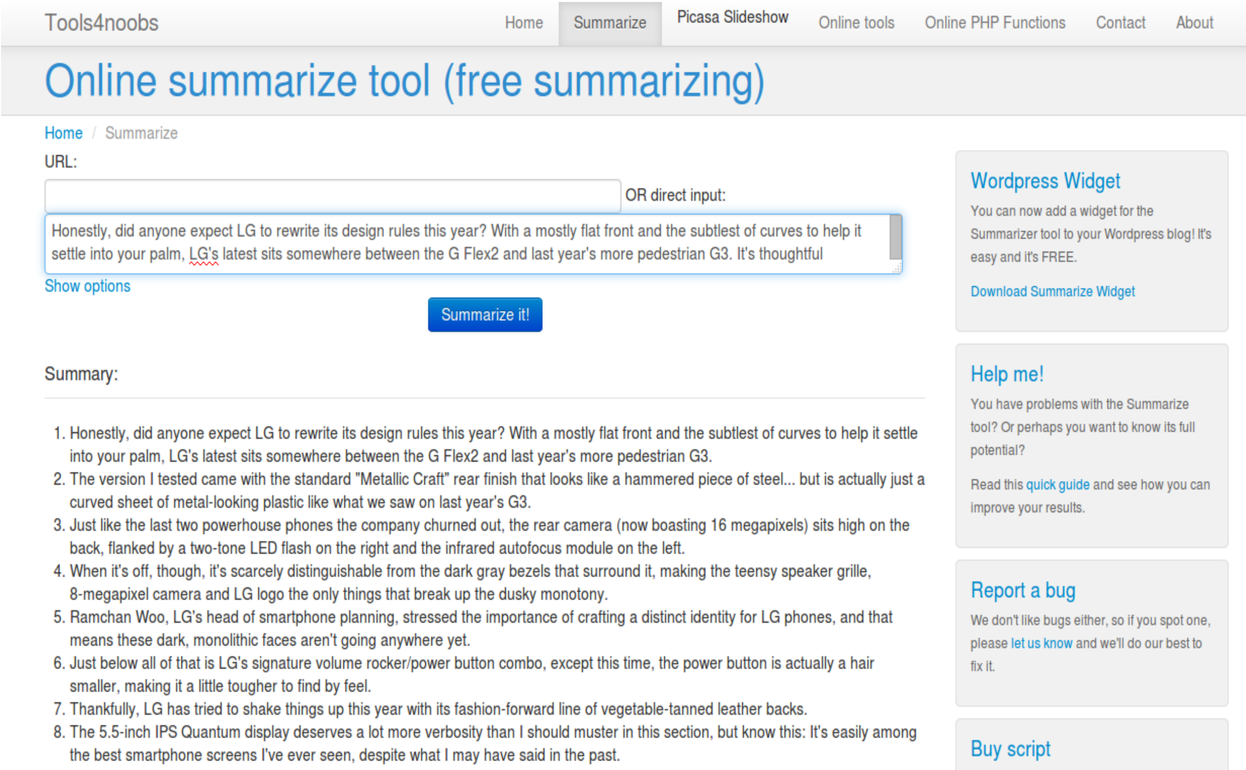

Figure 5.5: Standard Summarizer.

others tend to first deter- mine the most important topic of the document, and then collect only the sentences that are relevant to this topic.

We compare our method with a standard online summarizing tool from 'Tools4Noobs' [15]. It can be observed that for the given sample input, our methods fares equally well as the standard online summarizing tool.

6 Conclusion & Future Work

In this project, two text summarization methods that create generic text summaries by ranking and extract- ing sentences from the original documents were tested. The first method uses standard IR methods to rank sentence relevances, while the second method uses the latent semantic analysis technique to identify semantically important sentences, for summary creations. Both methods strive to select sentences that are highly ranked and different from each other. This is an attempt to create a summary with a wider coverage of the document's content and a less redundancy. Despite the very different approaches taken by the two summarizers, they both produced quite compatible performance scores. This fact suggests that the two approaches interpret each other. Finally,the two methods are compared with an online standard summarizer and challenges of evaluation are explained.

In future work, we plan to investigate machine learning techniques to incorporate additional features for the improvement of generic text summarization quality. The additional features we are currently considering include relation with title, position of the sentence in the document etc. and semantic features such as name entities, time, location information, etc.

References

[1] Karel Jezek and Josef Steinberger, "Automatic Text summarization", Vaclav Snasel (Ed.): Znalosti 2008, pp.1- 12, ISBN 978-80-227-2827-0, FIIT STU Brarislava, Ustav Informatiky a softveroveho inzinierstva, 2008.

[2] Farshad Kyoomarsi, Hamid Khosravi, Esfandiar Eslami and Pooya Khosravyan Dehkordy, Optimizing Text Summarization Based on Fuzzy Logic, In proceedings of Seventh IEEE/ACIS International Conference on Computer and Information Science, IEEE, University of Shahid Bahonar Kerman, UK, 347-352, 2008.

[3] Weiguo Fan, Linda Wallace, Stephanie Rich, and Zhongju Zhang, Tapping into the Power of Text Mining, Journal of ACM, Blacksburg, 2005.

[4] G Erkan and Dragomir R. Radev, LexRank: Graph-based Centrality as Salience in Text Summarization, Journal of Artificial Intelligence Research, Re-search, Vol. 22, pp. 457-479 2004.

[5] M. Sanderson, "Accurate user directed summarization from existing tools," in Proceedings of the 7'th International Conference on Information and Knowledge Management (CIKM98) , 1998.

[6] B. Baldwin and T. Morton, "Dynamic coreference-based summarization," in Proceedings of the Third Conference on Empirical Methods in Natural Language Processing (EMNLP3) , (Granada, Spain), June 1998.

[7] R. Barzilay and M. Elhadad, "Using lexical chains for text summarization," in Proceedings of the Workshop on Intelligent Scalable Text Summarization , (Madrid, Spain), Aug. 1997.

[8] C. Buckley and et al., "The smart/empire tipster ir system," in Proceedings of TIPSTER Phase III Workshop , 1999.

[9] J. Goldstain, M. kantrowitz, V. Mittal, and J. Carbonell, "Summarizing text documents: Sentence selection and evaluation metrics," in Proceedings of ACM SIGIR'99 , (Berkeley, CA), Aug. 1999.

[10] E. Hovy and C. Lin, "Automated text summarization in summarist," in Proceedings of the TIPSTER Workshop , (Baltimore, MD), 1998.

[11] Y. Gong, X. Liu: Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, New Orleans, Louisiana, United States 2001, pp. 19-25

[12] W. Press and et al., Numerical Recipes in C: The Art of Scienti c Computing . Cambridge, England: Cambridge University Press, 2 ed., 1992.

[13] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman, "Indexing by latent semantic analysis," Journal of the American Society for Information Science , vol. 41, pp. 391407, 1990.

[14] M. Berry, S. Dumais, and G. O'Brien, "Using linear algebra for intelligent information retrieval," Tech. Rep. UT-CS-94-270, University of Tennessess, Computer Science Department, Dec. 1994.

[15] http://www.tools4noobs.com/summarize/