Single Document Text Summarization

Major Project Report
Submitted in partial fulfillment of requirement of the degree
of
M.Sc.Computer Science

by

Surender Yadav (Roll no : 40829)
Rahul Dogra (Roll no : 40822)

Department of Computer Science
Faculty of Mathematical Sciences

University of Delhi, New Delhi, India
June 2017

Single Document Text Summarization

Major Project Report
Semester IV
M.Sc. Computer Science
by

Surender Yadav (Roll no : 40829)
Rahul Dogra (Roll no : 40822)

under supervision of
Dr. Vasudha Bhatnagar

Submitted in partial fulfillment of requirement of the degree of M.Sc. Computer Science

Department of Computer Science
Faculty of Mathematical Sciences
University of Delhi, New Delhi, India
June 2017

DECLARATION

This is to declare that the major project entitled "Single Document Text Summarization" being submitted in the partial fulfillment of the requirement for the award of the degree of M.Sc (Computer Science) is a record of original and bona-fide work carried out by the undersigned in Department of Computer Science, Delhi University, New Delhi, India.

The work presented in this Project has not been submitted to any other University or Institute for the award of any degree or diploma.

Surender Yadav (Roll no : 40829) **Rahul Dogra (Roll no : 40822)**

CERTIFICATE

The project entitled "Single Document Text Summarization" is being submitted in the partial fulfillment of the requirement for the award of the degree of M.Sc. (Computer Science), is a record of original and bona-fide work carried out by Rahul Dogra and Surender Yadav under the supervision of Prof. Vasudha Bhatnagar in the Department of Computer Science, Faculty of Mathematical Sciences, University of Delhi, New Delhi, India. The work presented in this Project has not been submitted to any other Institute or University for the award of any degree or diploma.

Supervisor
Prof. Vasudha Bhatnagar
Department of Computer Science
Faculty of Mathematical Sciences
University of Delhi

Head of Department
Prof. Vasudha Bhatnagar
Department of Computer Science
Faculty of Mathematical Sciences
University of Delhi

Date:_____

ACKNOWLEDGEMENT

We are thankful to our supervisor, **Dr.Vasudha Bhatnagar, Professor, Department of Computer Science, Delhi University**, for her invaluable contributions and generous suggestions towards the completion of this project. She always strived to provide us with best of her capability. We feel honored and privileged to have worked under her and guiding us at each and every step with her immense pool of knowledge and expertise of the topic.
We are also immensely grateful to the lab staff of Department of Computer Science, University of Delhi for providing us with lab facilities for carrying out the experiments. We truly believe that this project would not have been possible without the support of all of them.

Surender Yadav **Rahul Dogra**

ABSTRACT

The continuous expansion of World Wide Web and text collections online makes a large volume of information available to users. Automatic text summarization allows users to quickly understand documents and find whether they are useful or not. We implemented a published algorithm [1] for automatic single document summarization using the content-based and graph-based properties. We will be designing the evaluation measures for quantifying the extractive summary generated by the algorithm, i.e., Quality Factor and semantic similarity. We then will be evaluating the extractive summary generated by the algorithm on the 20 documents of DUC and benchmark summary given in DUC-2002 data collection.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Text Summarization

The quantity of data generated by continuing expansion of online text collections and World Wide Web is very large. Reading such a data for useful information is highly inconvenient, hence we use abstracts. An abstract is a brief summary of a document and is often used to help the reader quickly ascertain the topic and purpose of the document. The aspiration is to save reader's time and effort in finding useful information in a document. If a reader makes efforts in creating abstract, the resulting product is almost always effected by his perspective, context, and temperament. The quality of an abstract of a particular document may differ among abstracters, and if the same person at some other time were to abstract again, he might come up with a non-identical product.

An compressed version of a given document is called summary. The summary is expected to convey the main ideas of the document. Traditionally, language experts summarize documents when required. Recently, automatic document summarization has gained popularity.

The significant difference between automatic and human-based text summarization is that humans can capture and convey subtle themes that permeate documents, whereas automatic approaches have a large difficulty to do the same. To eliminate both human effort and bias, new tools and methods are needed to provide readers with summaries of document content so that they can quickly absorb the crux of documents.

The techniques for automatic text summarization can be classified into two categories: abstraction and extraction. The abstraction approach is the process of reducing a text document in order to create a summary that is semantically related. It is generally complicated due to the intricacy of natural language. However, the extraction approach consists of selecting important sentences from document based on statistical features.

1.2 Approaches

The single document summarization technique discussed in this report can be applied to a wide range of research needs. At the lowest level, keyword or key-phrase extraction summarizes the contents of a document. However, keywords give only a superficial idea about the topics covered in the document. Hence, they are of limited use. For example, given a research article, the algorithm can produce a list of keywords or key-phrases that capture the primary topics discussed.

Often, authors provide manually assigned keywords of articles, but most text lacks pre-existing key-phrases. For instance, news articles rarely have key-phrases attached, but it would be useful to be able to extract key-phrases automatically.

Single document summarization aims to identify the essence of a text. The only real difference is that now we are dealing with larger text units, whole sentences instead of words and phrases.

1.3 Preprocessing of Document

The format of the documents available on the online text collections are different enough, such that they require different preprocessing steps to make them usable for the input of the algorithm. Depending on the format of the document, appropriate steps need to be performed. For example, we have removed all the unicode characters and added newline characters between all the paragraphs of the text documents.

1.4 Goal

We aim to implement the summarization algorithm designed by Ohm Sornil and Kornnika Greeut [1]. We have tested the algorithm on 20 random documents from DUC 2002 dataset. We have also designed measures for computing the quality of summary produced by the algorithm.

Chapter 2

SVD based Summarization Algorithm

In this chapter, we descibe a text summarization algorithm proposed by O. Sornil and K. Greeut [1]. The algorithm uses the extraction approach for summarizing a single document which combines the content based and graph based techniques in two stages of computation.

In the first stage, sentences are represented by content-based 22 linguistic feature vectors. The feature matrix is then compressed into a lower dimensional matrix to uncover hidden association patterns and reduce small variations in sentence characteristics by using Singular Value Decomposition (SVD).

In the second stage, sentences are represented as nodes, and relationships between two sentences whose similarity scores are computed above a threshold are represented as edges in a document graph. A graph search technique is then used to recommend sentences to be extracted as summary. The proposed technique naturally combines content-based approach and link-based approach to text sentence extraction.

2.1 Algorithm

In order to extract a summary from a document, the document is divided into a set of text sentences. A sentence is a sequence of words that is delimited by stop marks (“.” “?”). Each sentence is pre-processed by stopword elimination and stemming using Porter stemming algorithm.

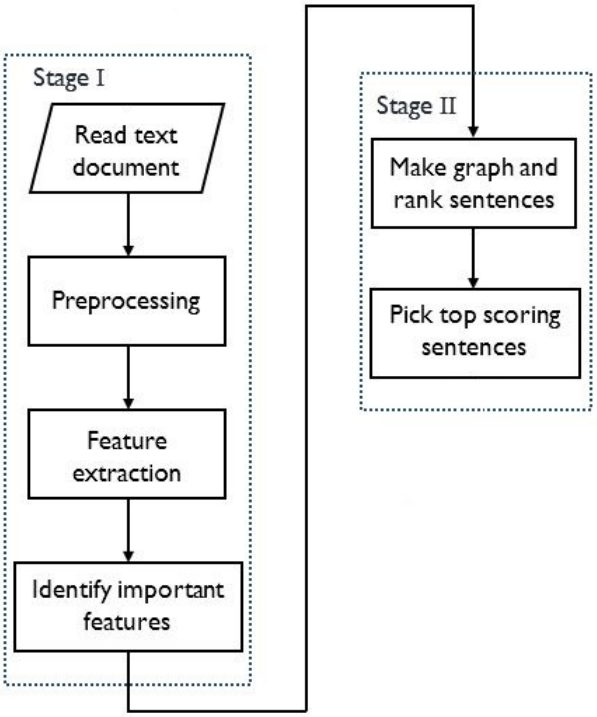


Figure 2.1: Basic Steps of the Algorithm

Calculation of feature matrix In order to get feature matrix, we compute 22 features of every sentence which can be classified into six categories: location of the sentence, relationship between the sentence and the document title, properties of terms in the sentence, relationship between the sentence and the entire document, existence of proper nouns and pronouns, relationship between sentence and significant terms.

i) Location of the sentence.

1. sentence Id: sentence identifier, normalized by the total number of sentences in the document.

$$segid = \frac{i}{n}, \tag{2.1}$$

where i is the respective sentence number and n is the total number of sentences in the document.

2. Paragraph Id: Paragraph identifier, normalized by the total number of paragraphs in the document.

$$paraid = \frac{i}{n}, \tag{2.2}$$

where i is the respective paragraph number and n is the total number of paragraphs in the document.

3. Paragraph Offset: sentence offset within the paragraph containing it, normalized by the total number of sentences in the paragraph.

4. Paragraph Location: Location of the paragraph containing the sentence relative to the document (initial, medial, or final).

5. Sentence Location: Location of the sentence in the document (1st, 2nd, 3rd, or 4th quarter).

6. Sentence Length: Number of terms in the sentence, normalized by the length of the longest sentence.

ii) Relationship between the sentence and the document title.

7. Title Word: Number of title words that appear in the sentence, normalized by the total number of title words.

8. Sim(sentence, title): Cosine similarity between the sentence and the title.

$$CosineSimilarity = Cos\theta = \frac{A.B}{||A||.||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt[2]{(\sum_{i=1}^{n} A_i^2)}\sqrt[2]{(\sum_{i=1}^{n} B_i^2)}}, \tag{2.3}$$

where A is the sentence vector and B is the title words vector.

iii) Properties of terms in the sentence.

9. Total tf: Sum of term frequencies in the sentence.

10. Average tf: Average term frequency in the sentence.

11. Total $tf \times isf$: Sum (over all terms in the sentence) of each term's frequency multiplied by its inverse sentence frequency.

12. Average $tf \times isf$: Average $tf \times isf$ of terms in the sentence.

13. Total $tl \times tf$: Sum (over all terms in the sentence) of each term's total frequency multiplied by their frequencies in the sentence.

iv) Relationship between the sentence and the entire document.

14. Sim(sentence, text): Cosine similarity between the sentence and the whole document.

$$CosineSimilarity = Cos\theta = \frac{A.B}{||A||.||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt[2]{(\sum_{i=1}^{n} A_i^2)}\sqrt[2]{(\sum_{i=1}^{n} B_i^2)}}, \tag{2.4}$$

where A is the sentence vector and B is the document words vector

15. Lexical: Number of terms shared with other sentences divided by total number of sentences in the document.

v) Existence of proper nouns and pronouns.

16. Uppercase: Does the sentence contain an uppercase word (yes or no).

17. Pronoun: Does the sentence contain a pronoun (yes or no).

18. Proper Noun: Does the sentence contain a proper noun (yes or no).

vi) Relationship between sentence and significant terms.

19. Significance Term: Number of significant terms that appear in the sentence, normalized by the total number of significant terms in the document.

20. Sim(sentence, sig term): Cosine similarity between the sentence and the set of significant terms.

$$CosineSimilarity = Cos\theta = \frac{A.B}{||A||.||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt[2]{(\sum_{i=1}^{n} A_i^2)}\sqrt[2]{(\sum_{i=1}^{n} B_i^2)}}, \tag{2.5}$$

where A is the sentence vector and B is the significant terms vector

21. Sig.Term-Luhn: Significant term factor calculated as proposed by Luhn [2]. It is described in appendix A.

22. Modified Sig.Term-Luhn: Modified Luhn's significant term factor [3] as described in appendix B.

Singular value Decomposition Feature matrix is Compressed using Singular Value Decomposition because: 1.) A feature matrix with n rows is generally large, since size of documents is large. So processing is computationally expensive due to high dimensionality.
2.) The organization of the segment-feature matrix assumes that all features are independent which is generally not true in practice.

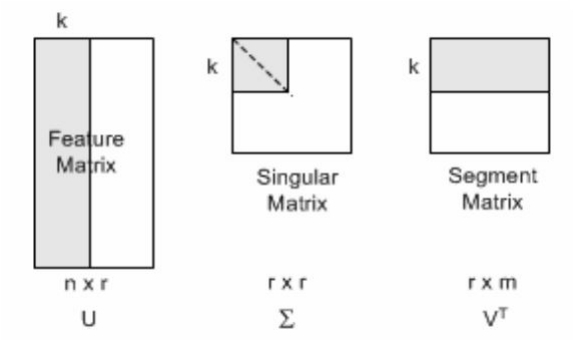


Figure 2.2: Singular Value Decomposition of a Matrix.

SVD is used to compress the feature matrix into a lower dimensional feature space. It can uncover hidden relationships among features and segments, and reduce effects of noises in segment characteristics. It decomposes the feature matrix into 3 components: an orthogonal matrix of singular values, where r = min(m, n), and the left and the right singular vectors (i.e., U and V, respectively).
By keeping $k < r$ largest values of the singular matrix along with their corresponding columns in U and V, the resulting matrix is the matrix of rank k which is closest to the original matrix A in the least square sense. The attributes are no longer independent from each other with respect to this new space of k dimensions.

Graph Construction The compressed segment-feature matrix is used as the basis for constructing a document graph. Sentences are represented as vertices, and their relationships (whose values are above a threshold) are represented as edges in the graph. The cosine similarity is used as measure for calculating the relationship between the vertices.

A graph is represented as an undirected weighted graph with orientation of edges set from a sentence to sentences.

PageRank Algorithm PageRank algorithm is part of the Web Page scoring mechanism used by Google which has been used in a variety of research including text summarization. It integrates the impact of both incoming and outgoing edges into one single model.

PageRank algorithm is used for finding significant nodes in the graph. The PageRank algorithm is applied on the graph and then top n% nodes are selected according to compression rate which is taken as 20% in all experiments in this report.

Chapter 3

Evaluation Measures

There are two ways of assessing the quality of summary generated by the algorithm based on whether summary of the document is available or not. When summary of the document is not given we use the Quality Factor measure, while if the summary of the document is known, Semantic Similarity method is utilized.

3.1 Quality Factor

The algorithm generates different summaries for different values of k for the same document. For example, the value of k for which we obtained a notable quality summary varied from 9 to 21 for different chapters of MHRD report on National Policy on Education 2016. Since the best value of k is not known to the user, we varied k for each document and found out the value which gives best quality summary. Therefore, to evaluate the quality of different summaries obtained from the document for the different values of k, we formulate a metric "Quality Factor".

The basic assumptions that the Quality Factor takes into account are:
1. The document is well written.
2. A paragraph of the document contributes only one sentence to the summary, describing the central theme of the paragraph.

Let X be the number of paragraphs contributing to the summary, Y be the number of paragraphs contributing more than one sentence in the summary, R be the compression rate, T be the total number of paragraphs in the document and, S be the total number of sentences in the document. Then, the quality factor is computed as,

$$Q.F. = \Big\{\frac{X}{min(T, R \times S)} \times \frac{X - Y}{X}\Big\} = \Big\{\frac{X - Y}{min(T, R \times S)}\Big\}, \tag{3.1}$$

The maximum possible value of the extractive summary is 1, when the following conditions are fulfilled:

1. When number of paragraphs contributing to summary, (X) and minimum of total number of paragraphs, (T) and total number of sentences in the summary, i.e., $R \times S$ are equal.

2. A paragraph from a document must not contribute more than one sentence to the summary, i.e., $Y = 0$.

In this case it can be inferred that the document is well written and each paragraph has a single central theme.

The lowest possible value of the extractive summary is 0, when the following conditions are satisfied:

1. When all the sentences of the summary belong to a single paragraph of the document, then the Quality Factor becomes 0.

2. All the paragraphs comprising the summary serve more than one sentence to the summary, i.e., $X = Y$ then, the numerator of eq. (3.1) reduces to 0, or in other words the quality factor becomes 0.

In this case it can be concluded that the document is not well written and all the paragraphs composing the summary are contributing more than one sentence.

3.2 Semantic Similarity

For measuring the quality of summary when an abstract or extract summary is available, we use calculate semantic similarity method using Semilar [4], which is a free tool for finding semantic similarity. In the semantic similarity approach, the meaning of the extractive summary obtained after applying the algorithm is inferred by assessing how similar it is to another summary, called the benchmark summary, whose meaning is known. Benchmark summary is the abstractive or extractive summary obtained from the dataset.

Each sentence of extractive summary is compared with all the sentences of the benchmark summary, and the maximum similarity value obtained is used to indicate the degree upto which a sentence is similar to a sentence of the benchmark summary. The process of finding maximum similarity is shown in the complete weighted bipartite graph drawn between sentences of extractive and benchmark summary, weighted by the semantic similarity between two sentences as shown in figure 3.1. The maximum similarity value between sentences is denoted by red coloured lines.

Note that, two sentences from the extractive summary can acquire maximum similarity value on the same sentence from the benchmark summary, given the case that benchmark summary is abstractive because, an abstract sentence can be composed by joining two or more extract sentences. Finally, the average of all the computed sentence similarities is calculated. This gives us a measure of similarity between extractive summary and benchmark summary.

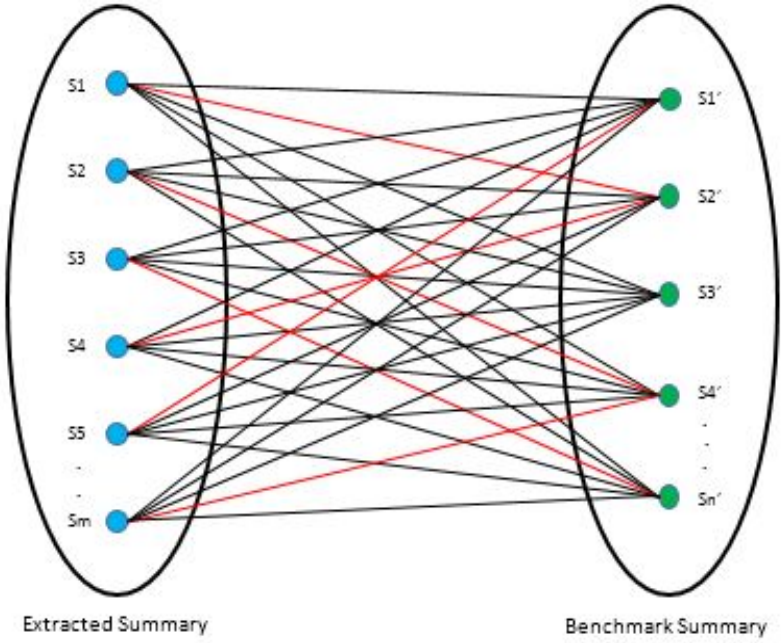


Figure 3.1: Complete Weighted Bipartite Graph between sentences of Extractive and Abstractive Benchmark Summary

Chapter 4

Implementation and Performance Evaluation

In this chapter, we will briefly introduce the software tools used in implementation and evaluation of the algorithm. Then, the dataset used is discussed concisely. Finally, the observations of the results obtained after performance evaluation using evaluation measures from Chapter-3 are explained.

4.1 Software Tools Used

We have used the Windows operating system with the Python version 3.5.2 for implementing the algorithm. The package names, function names and their usage in the implementation of the program are shown in table 4.1.

Package	**Function**	**Used For**
nltk	sent_tokenizer	Sentence boundary detection
nltk	word_tokenizer	Tokenizing sentences into words
nltk	pos_tag	Finding tags of words,e.g. noun, pronoun etc.
ntlk.corpus	stopwords	Removing stopwords
nltk.stem.porter	PorterStemmer	Stemming
sklearn.metrices	cosine_similarity	Finding cosine similarity
numpy	column_stack	Creating a matrix
scipy.linalg	svd	Finding the singular value decomposition of matrix
networkx	from_numpy_matrix	Creating a graph from matrix
networkx	pagerank	Calculating rank of nodes in Graph

Table 4.1: The Functions used for Algorithm Implementation

Softwares used for the evaluation of automatic summary generated by the algorithm:

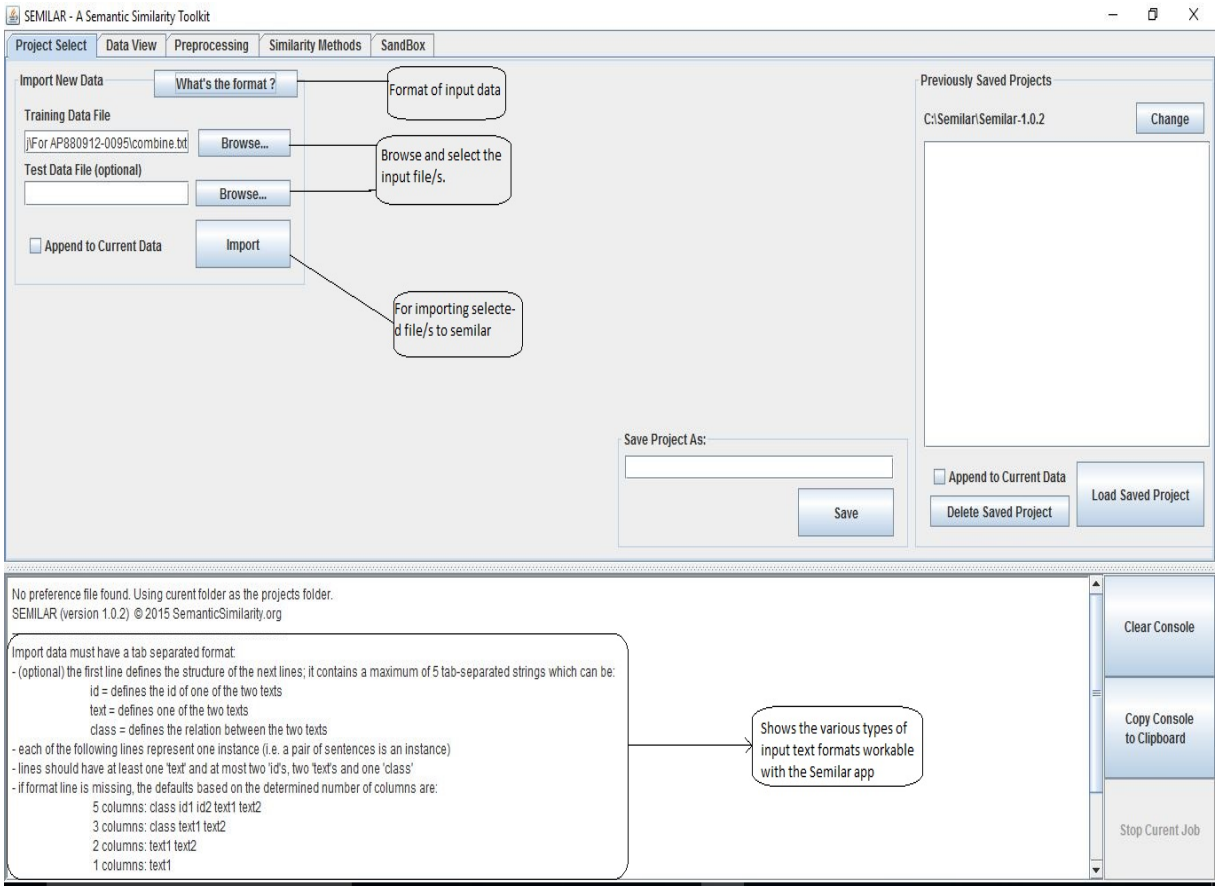
- RStudio: RStudio is a free and open-source integrated development environment (IDE) for R, a programming language for statistical computing and graphics.

- Weka: Waikato Environment for Knowledge Analysis (Weka) is a suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. It is free software licensed under the GNU General Public License.
- Semilar: Described in section 4.1.1.
- MS-Excel: Microsoft Excel is a spreadsheet developed by Microsoft, for features calculation, graphing tools, pivot tables, and a macro programming language called Visual Basic for Applications.

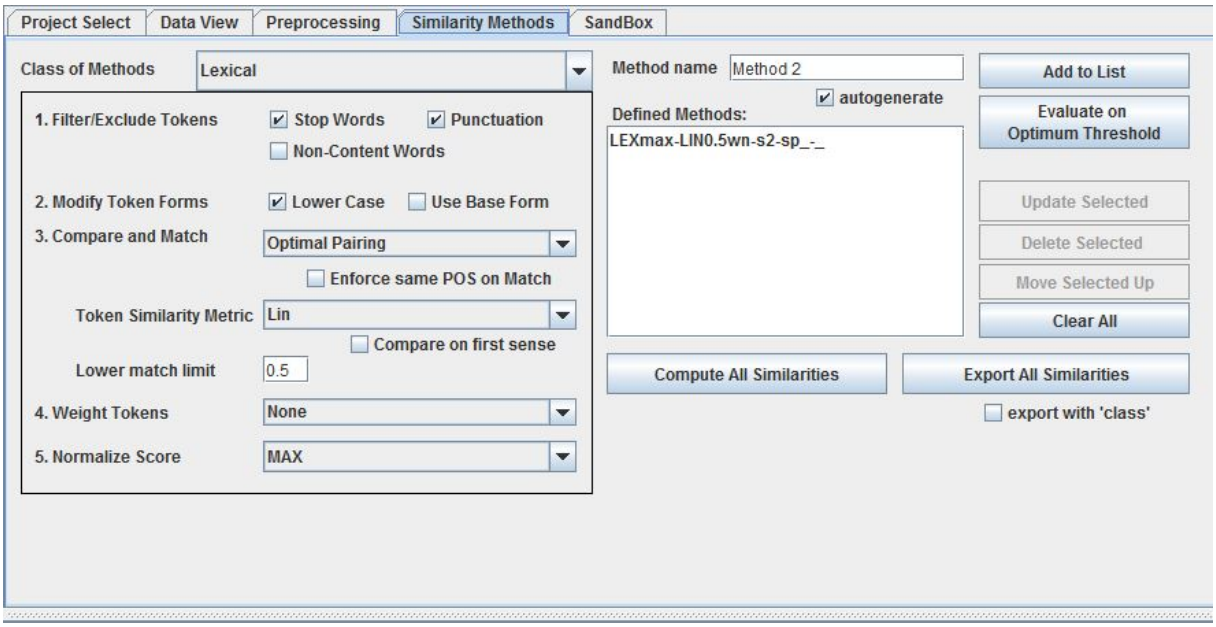
4.1.1 Semilar

SEMILAR, a SEMantic simILARity toolkit. SEMILAR implements a number of algorithms for assessing the semantic similarity between two texts. It is available as a Java library and as a Java standalone application offering GUI-based access to the implemented semantic similarity methods.

Screenshots of various tabs in SEMILAR [4] application are shown below.



In the project folder you can load datasets or previously save projects.



From the Similarity Methods, you can choose similarity methods and their configuration parameters.

4.2 Dataset

We have taken the DUC(Document Understanding Conferences) [5] - 2002 dataset. The dataset contains document sets consisting of newspaper articles from Wall Street Journal 1987-1992, AP newswire 1989-1990 etc.

There will be per-document summaries for each document with an approximate length of 100 words.Also, four multi-document summaries extrative and abstractive sumaries are present for each set.

Since the given extractive summary is multi document so that doesn't meet our requirement. Thus we will only be using the abstractive summary and text documents of the DUC-2002 dataset [5].

4.3 Performance Evaluation

We selected the 20 random text documents and their summaries of the DUC-2002 dataset [5]. And we computed the quality factor and the semantic similarity between the extracted and benchmark summaries of these documents.

Note that for finding the semantic similarity, the value of k for which summary has the maximum quality factor has been taken as the extractive summary for finding the semantic similarity with the benchmark summary.

Table 4.2: Basic information of Documents

Document No.	**Number of Paragraphs**	**Number of Sentences**
1	25	32
2	23	30
3	38	56
4	18	35
5	18	27
6	8	26
7	20	42
8	21	28
9	20	29
10	17	27
11	14	22
12	28	36
13	24	43
14	17	31
15	19	27
16	17	30
17	25	43
18	29	49
19	14	28
20	19	26

Table 4.3: Performance statistics of 20 documents of DUC dataset

Document No.	Quality Factor	Semantic Similarity
1	0.9375	0.701311
2	1	0.724326
3	0.982142857	0.785737
4	1	0.778403
5	0.925925926	0.720037
6	0.576923077	0.743817
7	0.595238095	0.745641
8	0.892857143	0.79838
9	0.862068966	0.757115
10	0.925925926	0.720988
11	0.909090909	0.735788288
12	0.972222222	0.752596
13	0.930232558	0.809376
14	0.967741935	0.708908
15	0.925925926	0.710994
16	1	0.744014
17	0.930232558	0.773272
18	0.918367347	0.780798
19	0.892857143	0.71454
20	0.961538462	0.704392
Average	0.90354	0.745522

4.4 Observations

It is evident from table 4.3 that the documents have large number of small paragraphs. Consequently, the quality factor of all the documents is reasonably high. Furthermore, the value of Semantic Similarity is also notably high. This can be due to the fact that the sentences are closely related.

The structure of the text document affects the Quality Factor of summary produced by the algorithm.

Chapter 5

Conclusion

We implemented a document summarization algorithm based on SVD. The algorithm extracts 22 features for each sentence in the document to construct a feature matrix. SVD is then applied to the feature matrix to reduce the number of features. The reduced feature matrix is used to construct an undirected, weighted graph of sentences. Page rank algorithm is applied to this graph and top 20% sentences are selected as extractive summary of the document.

We also designed two measures for judging the quality of extracted summary. The first measure is based on the intuition that in a well written document no more than one sentence will be contributed by a paragraph composing the summary. The second measure is based on semantic similarity.

Quality Factor can be used to select the best summary generated by the algorithm for different values of k. It can also help to deduce whether the structure of the document is such that, the useful information is uniformly distributed throughout the document, instead of clustered distribution of valuable data. Semantic similarity is used to determine whether the summary generated is as good as the given abstract or extract summary of the document.

Bibliography

[1] O. Sornil and K. Gree-ut. An automatic text summarization approach using content-based and graph-based characteristics. In *2006 IEEE Conference on Cybernetics and Intelligent Systems*, pages 1–6, June 2006.

[2] H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165, Apr 1958.

[3] Youjin Chang, Ikkyu Choi, Jongpill Choi, Minkoo Kim, and Vijay V Raghavan. Conceptual retrieval based on feature clustering of documents. *Proc MF/IR 2002*, 2002.

[4] Vasile Rus, Mihai C Lintean, Rajendra Banjade, Nobal B Niraula, and Dan Stefanescu. Semilar: The semantic similarity toolkit. pages 163–168, 2013.

[5] Document Understanding Conference. . http//www-nlpir.nist.gov/projects/duc/. 2002.

Appendices

Appendix A

Hans Peter Luhn's Significant Factor

The "significance" factor is a measure which determines the information content of a sentence. It is computed by evaluating the following two factors:

(i) Significant Words

(ii) Relative Significance

A.1 Significant Words

A set of significant words is established by measuring word frequencies for every unique word appearing in the document and then removing recurrent and rare words. The frequency of words is used as a measure because of the fact that a writer normally repeats certain words as he advances or varies his arguments and as he elaborates on an aspect of a subject. In a technical discussion, there is a very small probability that a given word is used to reflect more than one notion. The probability is also small that an author will use different words to reflect the same notion. Even if the author makes a reasonable effort to select synonyms for stylistic reasons, he soon runs out of legitimate alternatives and falls into repetition.

Using frequency for "significance" avoids linguistic implications such as grammar and syntax. In general, the method does not even propose to differentiate between word forms. Thus the variants differ, differentiate, different, differently, difference and differential could ordinarily be considered identical notions and regarded as the same word. This is achieved by using simple word stemming procedure of Python. Recurrent common words that tether significant words together are also excluded by using simple stopword removal procedures of python. Rare words are eliminated by simply dropping low frequency words. Procedures as simple as these are beneficial from the perspective of economy.

A.2 Relative Significance

Relative significance of sentences is established by relative position of significant words within a sentence. Two basic characteristics of spoken and written language provides the support for using relative position of significant words as a measure.

i) The greatest number of significant words found closer to each other, the probability is very high that the information being conveyed is most representative of the document.

ii) The ideas most closely associated intellectually are found to be implemented by words most closely associated physically.

From these characteristics a "significance factor" can be derived which reflects the number of occurrences of significant words within a sentence and the linear distance between them due

to the intervention of non-significant words. All sentences may be ranked in order of their significance according to this factor, and the highest ranking sentences may then be selected to serve as the auto-abstract. Therefore the criterion is the relationship of the significant words to each other. It therefore appears proper to consider only those portions of sentences which are bracketed by significant words and to set a limit for the distance at which any two significant words shall be considered as being significantly related. A significant word beyond that limit would then be disregarded from consideration in a given bracket, although it might form a bracket, or cluster, in conjunction with other words in the sentence.

An analysis of many documents has indicated that a useful limit is four or five non-significant words between significant words. If with this separation two or more clusters result, the highest one of the several significance factors is taken as the measure for that sentence.

$$Significance factor, S1 = \frac{X^2}{Y} \tag{A.1}$$

where, X is the number of significant words contained in the cluster and, Y is the total number of words within cluster.

Appendix B

Luhn's Significant Factor

The title of an article often reveals the major subject of that document, it conveys the general idea of its contents. In order to utilize this attribute in scoring sentences, each constituent term in the title section is looked up in the sentence. For each sentence a title score is computed as follows:

The title score for a sentence is given by

$$S2 = \frac{P}{Q}, \tag{B.1}$$

where, P is the total number of title terms found in a sentence and, Q is the total number of terms in a title

The final score for each sentence is calculated by summing its Luhn's significance factor and corresponding Title score.

$$SentenceSignificanceScore, S3 = S1 + S2. \tag{B.2}$$