AUTOMATIC TEXT SUMMARIZATION

MTP Report submitted to

Indian Institute of Technology Mandi

for the award of the degree

of

B. Tech

by

SHUBHAM AJMERA

under the guidance of

Dr. ARTI KASHYAP

SCHOOL OF COMPUTING AND ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MANDI

JUNE 2015

CERTIFICATE OF APPROVAL

Certified that the thesis entitled **AUTOMATIC TEXT SUMMARIZATION**, submitted by **SHUBHAM AJMERA** , to the Indian Institute of Technology Mandi, for the award of the degree of **B. Tech** has been accepted after examination held today.

Date :

Mandi, 175001

Faculty Advisor

CERTIFICATE

This is to certify that the thesis titled **AUTOMATIC TEXT SUMMARIZATION**, submitted by **SHUBHAM AJMERA**, to the Indian Institute of Technolog, Mandi, is a record of bonafide work under my (our) supervision and is worthy of consideration for the award of the degree of **B. Tech** of the Institute.

Date :

Mandi, 175001

Supervisor(s)

DECLARATION BY THE STUDENT

This is to certify that the thesis titled **AUTOMATIC TEXT SUMMARIZATION**, submitted by me to the Indian Institute of Technology Mandi for the award of the degree of **B. Tech** is a bonafide record of work carried out by me under the supervision of **DR. ARTI KASHYAP**. The contents of this MTP, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Date :

Mandi, 175001

SHUBHAM AJMERA

Acknowledgments

I would like to take this opportunity to thank my guide Dr. Arti Kashyap for supporting me throughout the course. She whole heartedly supported me at all the stages of the project. I'd like to express my heartliest gratitude to her.

I would als like to thank Dr. Dileep A. D., who have offered me their invaluable experience and knowledge to guide me throughout the project and accomplish the project goals.

Further, I am grateful to my team mates, Saurabh Jain and Rohit Shukla, for all the contributions they have made to this project.

Shubham Ajmera

Abstract

With the increasing amount of information, it has become difficult to take out concise information. Thus it is necessary to build a system that could present human quality summaries. Automatic text summarization is a tool that provides summaries of a given document. In this project three different approaches have been implemented for text summarization. In all these three summarizers, sentences are represented as a feature vector. In the first approach, features like position of sentences, vocabulary intersections, resemblance to title, sentence inclusion of numerical data are used and model is trained using Genetic Algorithm. In the second approach, apart from the features used in the first approach, structure of the document, popularity of content are also used as features. In the third approach, word2vec algorithm is used to generate summary.

Keywords: *summarization, text, natural language processing, nlp, text mining, wikipedia, google search, feature extraction, wordnet*

Contents

Abbreviations

NN	- Neural Network
FV	- Feature Vector

Symbols

$\sum$	- Summation over all values

List of Tables

List of Figures

Chapter 1

Introduction

Accoring to WordNet(Princeton) summary is defined as *"a brief statement that presents the main points in a concise form"*. Automatic Summarization is a process of generating summaries by a computer program.

Summarization process involves interpretation, transformation and generation. There are two types of summarization: abstraction-based summarization and extraction-based summarization. In abstract-based summarization, an abstract is created by interpreting the text contained in the original document and generating summary that express the same in a more concise way. In extractive-based approach some sentences from the original text are taken out to form a meaningful summary. In this approach sentences are given scores based on different feature and sentences with higher rating are selected for summary. It uses various Natural Language processing approaches for information retrieval.

Sumamrization process can also be classified based upon the number of source documents, task specific contraints and use of external resources as shown in the Figure 1.1. Summarization is classified as *single-document* or *multi-document* based upon the number of source document. In multi document summaization information overlaps between different document makes task difficult. Based upon external resources summarization can also be classified as *knowledge-rich* or *knowledge-poor*. Knowledge rich summarizer uses external source external corpus like Wikipedia, WordNet etc. In *query-focussed or query oriented summarization* summary is constructed with information relevant to the query. In *update summarization* summarizer makes use of current trends for construction of the summary.

The purpose of the update summary is to identify new pieces of information from the document [1].

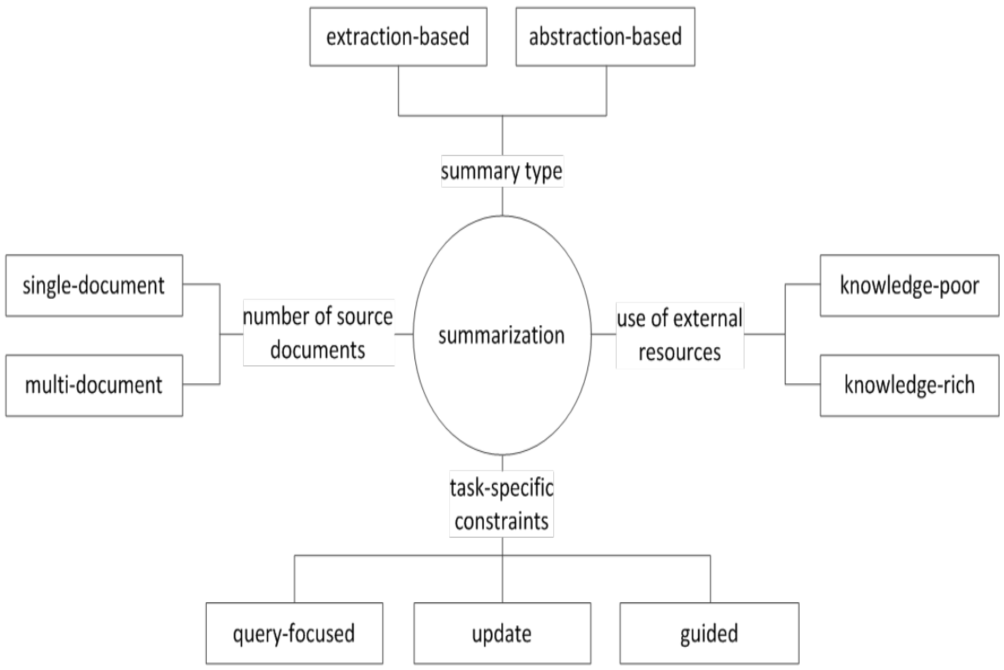


Fig. 1.1: Classification of summarization tasks. [1]

1.1 Objective

The objective of the project is to provide summaries of document using different summarization techniques and to compare the quality of the summaries generated. Different leaning models, data modelling techniques are also tested like Neural Networks(NN), Genetic Algorithms(GA).

1.2 Methods and Techniques

In this project we have implemented three different techniques for extractive based text summarization. Mainly the approaches used are different in terms of features used and machine learning model. In the first approach, we used sentence position, positive keywords in sentence, negative keywords in sentence, sentence centrality, sentence inclusion of name entit, sentence inclusion of numerical data, and sentence relative length [2]. The model is trained using Genetic Algorithms [2]. In the second approach, we chose to work on documents which are well structured and in which sentences are less connected. For summarization using this techniue, we have used Wikipedia articles, as they provide structured content, and sentences are less connected. In this approach google search, wordnet word similarity [3], tf-idf, and sentence position featues are used. In the third approach, continuous bag of words and skip-gram architecture are implemented using Word2Vec toolkit [4] and neural networks are used to train the summarizer.

Chapter 2

Background and Related Work

Abdel Fattah, Mohamed *et al.* [2] has proposed a simple approach for text summarization. They have considered features like position, length, name entities, numerical data, bushy paths, vocabulary overlaps etc. to generate summary. In this approach, sentences are modelled as vectors of features. Sentences are marked as correct if they are to be put in summary while are marked as incorrected if not. While making the final choice of sentences, each sentence is given a value in between 0 and 1 and using a machine learning model, the sentences are selected using those scores.

Various other papers have also been published which are very useful for a particular class of document. Kamal Sarkar *et al.* [5] is built for summarization of medical documents using machine learning approach. Various features are very comman and specific to medical documents. It uses the concept of cue phases such that if a sentences contains n cue phases, it gets n as its score for the feature. It also uses position of cue phases in the document such that if it appears at the beginning or at the end of the sentence, it get an additional 1 point. Acronyms are also used as a feature and sentences having these gets extra points. In some papers [6], sentences are also broken down by special cue makers and sentences are represented as a feature vectors.

Ryang, D Seonggi *et al.* [7] proposed a method of automatic text summarization with reinforcement learning. Researches have also been done for summarization of Wikipedia articles. Hingu , D *et al.* [8] implemented various method for summaring Wikipedia pages. In one of their methods, sentences containing citations are given higher weigtage. In the other approach, the frequency of words are adjusted based on the root form of the word. The words are stemmed with the objective to assign equal weigtage to words with the same root word.

Edmundson (1969) [9] proposed an approach of extraction-based summarization using features like position, frequency of words, cue words and the *skeleton* of an article by manually assigning weight to each of these features. The system was tested using manually generated summaries of 400 technical documents. The results were good with 44% of summaries generated by it were matching the manual summaries.

Chapter 3

Wikipedia Articles Summarization

3.1 A Naive Summarizer

A naive summarizer was implemented in Java and the features used were:

1. **f1 = Sentence Position**: Sentences appearing in the beginning and at the end of the document are given higher weightage. [2]

2. **f2 = Positive keyword in the sentence**: It is the frequency of keyword in the summary. [2]

$$Score_{f2}(s) = \frac{1}{Length(s)} \sum_{i=1}^{n} tf_i * P(keyword_i|keyword_i) \tag{3.1}$$

   Where, s is a sentence, n is the number of keywords in s, tfi is the occurring frequency of keywordi in s. We divide the value by the sentence length to avoid the bias of its length.

$$P(s \in S|keyword_i) = \frac{P(keyword_i|s \in S)P(s \in S)}{P(keyword_i)} \tag{3.2}$$

$$P(keyword_i|s \in S) = \frac{\#(sentences\ in\ summary\ and\ contains\ keyword_i)}{\#(sentences\ in\ summary)} \tag{3.3}$$

$$P(s \in S) = \frac{\#(sentences\ in\ the\ training\ corpus\ and\ also\ in\ the\ summary)}{\#(sentences\ in\ the\ training\ corpus)} \tag{3.4}$$

$$P(keyword_i) = \frac{\#(sentences\ in\ the\ training\ corpus\ and\ contains\ keyword_i)}{\#(sentence\ in\ training\ corpus)} \tag{3.5}$$

3. **f3 = Negative keyword in the sentence**: The keywords that are unlikely to appear in summary. [2]

$$Score_{f3}(s) = \frac{1}{Length(s)} \sum_{i=1}^{n} tf_i * P(s \notin S | keyword_i) \tag{3.6}$$

4. **f4 = Sentence Centrality**: It is the overlap in vocabulary of the sentence and rest of the document. It demonstrate similarity of the sentence with the document. [2]

$$Score_{f4}(s) = \frac{Keywords\ in\ s \cap Keywords\ in\ other\ sentences}{Keywords\ in\ s \cup Keywords\ in\ other\ sentences} \tag{3.7}$$

5. **f5 =Sentence Resemblance to the title**: It is the overlap in vocabulary between the sentence and the title of the docuement. [2]

$$Score_{f5}(s) = \frac{Keywords\ in\ s \cap Keywords\ in\ title}{Keywords\ in\ s \cup Keywords\ in\ title} \tag{3.8}$$

3.1.1 Architecture

The project is built on Java. The process of summarization begins with processing of input document which is broken down into sentences and subsequently into words. Summarizer maintains a list of sentences of the document and each sentence is responsible to store the words contained in it. A UML class diagram of the same is shown in Figure 3.1.

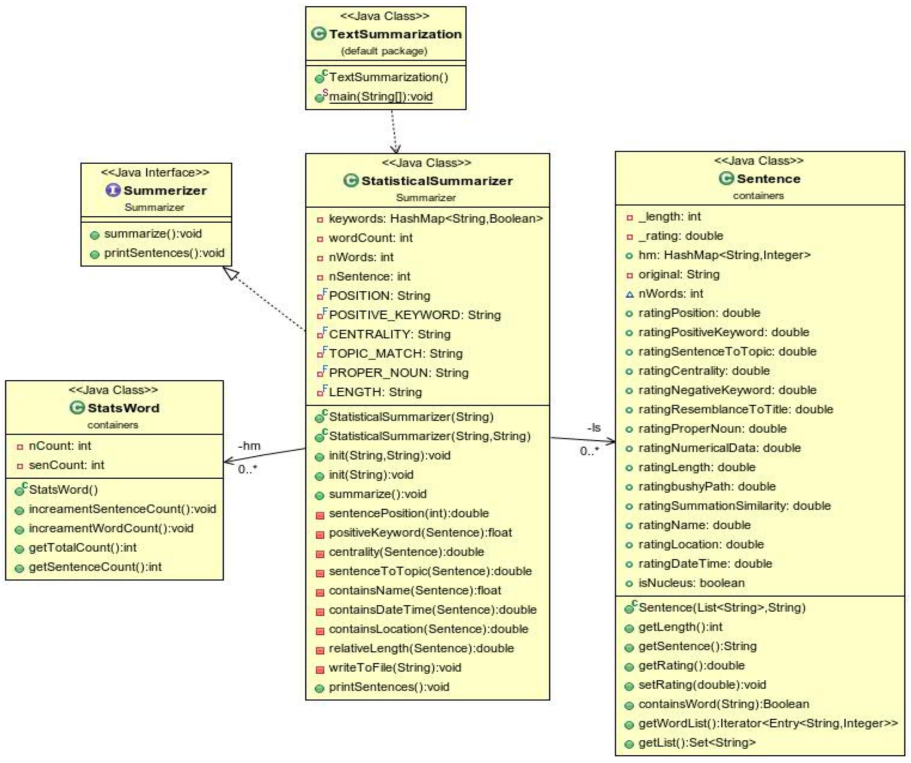


Fig. 3.1: UML Class diagram of Summarizer

The summarizer calculates score of each feature for every sentence. It uses ***Apache OpenNLP*** for *name finder(dataset - en-ner-person.bin)*, *location finder(dataset - en-ner-location.bin)*, and *data/time finder(dataset - en-ner-person.bin, data/en-ner-time.bin)*. After getting scores, it adds them up by giving equal weightage to each feature value using equation 3.9 and returns final score and based upon which the summary is generated.

$$Score(s) = \sum^{i} Score_{f_i} \tag{3.9}$$

3.2 Wikipedia Summarization

“Wikipedia is a free-access, free content Internet encyclopedia, supported and hosted by the non-profit Wikimedia Foundation” [10].

Wikipedia Summarizer provides summaries for given Wikipedia pages. The project is built using Django Web Framework, and uses features like *Google Search result count*, *word* *similarity using Wordnet*, *sentence position* etc. A custom summary option is also provided such that user can give keywords which he wants/doesn't want in summary.

3.2.1 Methodology

The summarizer fetches pages from Wikipedia and filters out tags, references and other meta content from it. The processed input page is then broken down section. Each section maintains a separate list of sentences contained in it. After processing and storing of page content, summarizer calculates feature values for each sentences. List of features used are as follow:

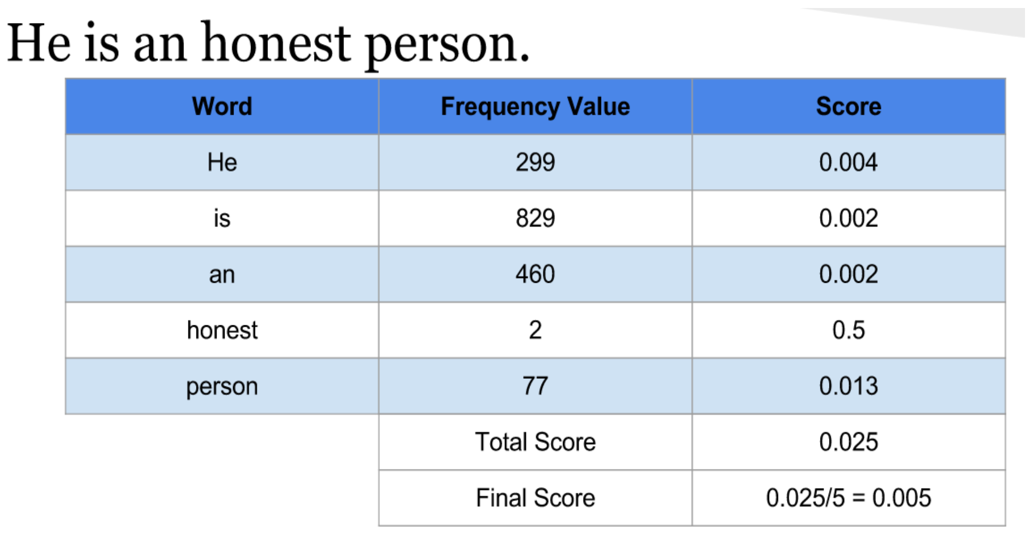
1. **Sentence Position**: It is a traditional method for providing score for the sentences [2]. Each sentence in a section is given a score based on its relative position in its section. Sentences appearing in the beginning of the section are given higher weightage as shown in equation 3.10.

$$Score(s) = \frac{1}{position\ in\ the\ section} \tag{3.10}$$

2. **TF-IDF**: TF-IDF for a word in a sentence is inversely proportional to the number of documents which also contain that word. Words with high TF-IDF numbers imply a strong relationship with the sentence they appear in, suggesting that if that word were to appear in a sentence, it could be interest to the user [5]. Data frequency values are constructed using 1000 randomly selected Wikipedia pages. Higher the frequency of the word in corpus, lower will be its value. The score of sentence is calculated using equation 3.11

$$Score(s) = \frac{\sum_{word_i \in W} \frac{1}{occurance\ of\ word_i\ in\ the\ corpus}}{No\ of\ words\ contained\ in\ the\ sentence} \tag{3.11}$$

where W is the list of words contained in the sentence. An illustration is shown in figure 3.2.

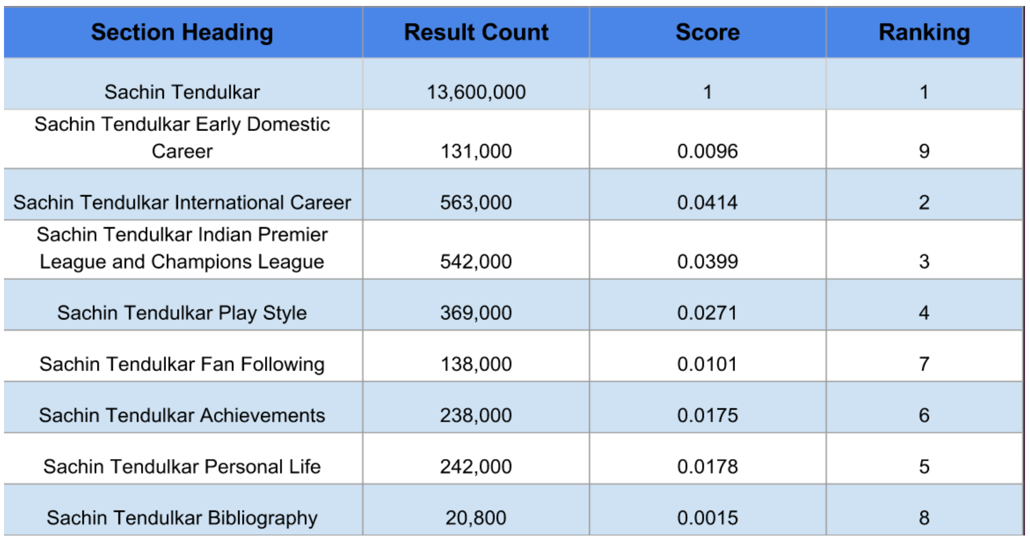


He is an honest person.

Word	Frequency Value	Score
He	299	0.004
is	829	0.002
an	460	0.002
honest	2	0.5
person	77	0.013
	Total Score	0.025
	Final Score	0.025/5 = 0.005

Fig. 3.2: TF-IDF Illustration

3. **Google Search Result Count**: This is one of the most important feature in which importance of section is computed by comparing Google Search result count. A search is performed for each section using query formed by concatenating section heading and title of the document. Score are given relative to the section which has most number of counts as shown in figure 3.3.



Section Heading	Result Count	Score	Ranking
Sachin Tendulkar	13,600,000	1	1
Sachin Tendulkar Early Domestic Career	131,000	0.0096	9
Sachin Tendulkar International Career	563,000	0.0414	2
Sachin Tendulkar Indian Premier League and Champions League	542,000	0.0399	3
Sachin Tendulkar Play Style	369,000	0.0271	4
Sachin Tendulkar Fan Following	138,000	0.0101	7
Sachin Tendulkar Achievements	238,000	0.0175	6
Sachin Tendulkar Personal Life	242,000	0.0178	5
Sachin Tendulkar Bibliography	20,800	0.0015	8

Fig. 3.3: Google Search Result Count Illustration

4. **Word Similarity**: This feature provides customizability to the summarizer, in which a

user can select keywords that he wants/doesn't want in the summary. The summarizer then compares similarity of that word with every word in the sentence using Wordnet dataset. If a sentence contains similar words then depending upon the preference of user it scores the sentence. For positive keywords, it increments the score for sentences containing similary words on the other hand for negative keywords, it decrements the score for sentences containing similary keywords. Some examples of words similarity are shown in Table 3.1.

Word1	**Word2**	**Score**	**Explaination**
cat	tiger	0.5	Both belong to *Felidae family*.
apple	tiger	0.1	Less related
apple	orange	0.25	Both are fruits
cat	cat	1.0	Same words
cat	feline	0.5	Synonyms
history	past	0.5	Both referring to time

Table 3.1: Illustration of Word Similarity

3.2.2 Summarizer Class

Wikipedia summarizer is implemented in Python and using Django Web Framework. It is a *multi-threaded* tool which performs feature extraction concurrently as shown in figure 3.4. Detailed implementation of each component is provided below:

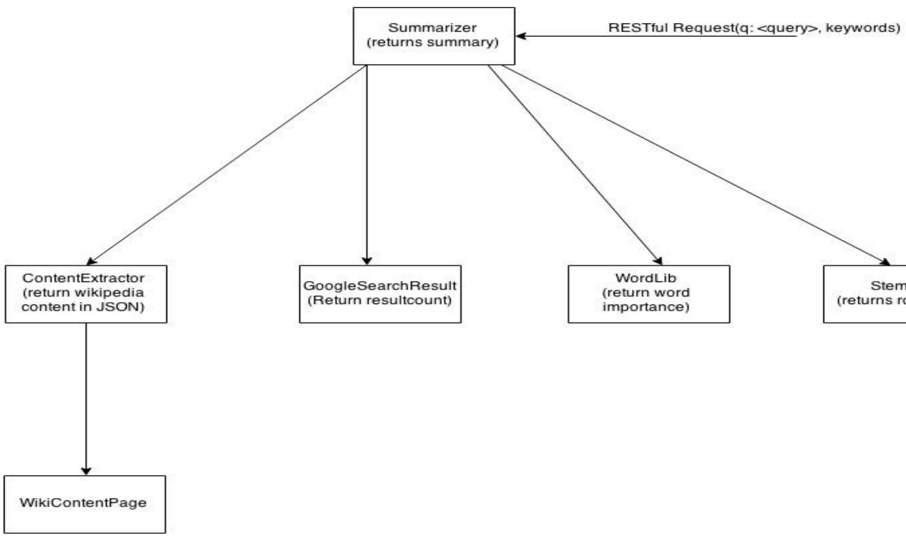


Fig. 3.4: Call flow of summarizer.

1. **Extractor**: It extracts out content from Wikipedia pages using `urllib2`. Extractor is provided with the title of the Wikipedia article, using which it forms url to extract out Wikipedia pages.

```
url = 'en.wikipedia.org/w/index.php?action=raw&title=' + title
```

2. **Text Processor**: After receiving data from the extractor, it removes various content including references, tags, tables, and unwanted sections. Then it breaks down the article into section using regular expressions and returns a JSON object to the summarizer as shown in Figure 3.5

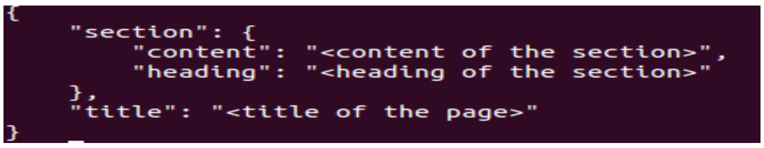

```
{
    "section": {
        "content": "<content of the section>",
        "heading": "<heading of the section>"
    },
    "title": "<title of the page>"
}
```

Fig. 3.5: Structure of JSON object returned after Extraction and Processing

3. **Image Extractor**: It provides image to the front-end. It extracts image from `infobox` of the Wikipedia page using *BeautifulSoup*

4. **Summarizer**: After receiving processed data, it starts evaluating scores of sentences for different feature. It concurrently calculates sentences' score for faster processing. It eventually calculates score of each sentences.

5. **Google Search Result Count**: It uses *selenium web driver* and *BeautifulSoup* for extracting Google Search result count. Selenium web driver uses mozilla to open Google result page for the input query. After that the text from the page is processes by BeautifulSoup and subsequently count is fetched from there using regular expressions. It takes a pause of two seconds before making a new query.

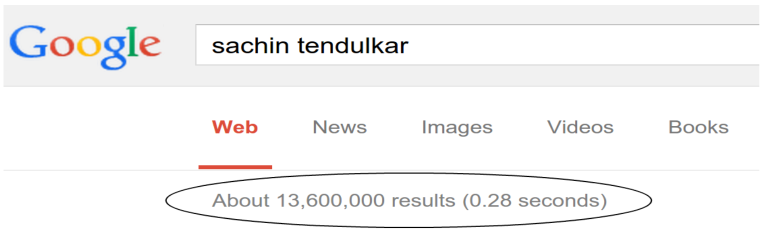
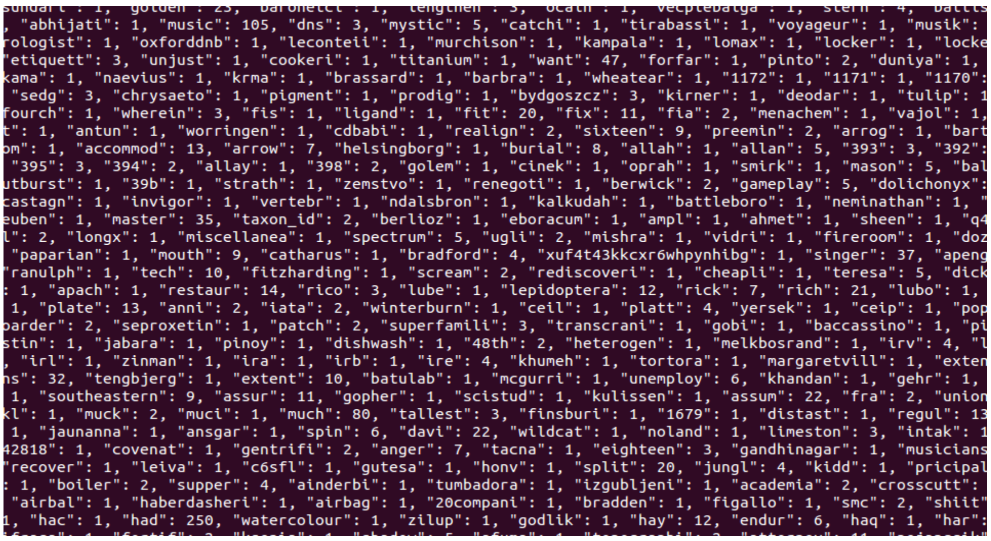


Fig. 3.6: Google Result Count Example

6. **TF-IDF**: A database of stemmed words is generated from randomly selected 1000 Wikipedia articles. Using the same database, and for the input Wikipedia article, words are stemmed and scores are calculated using Equation 3.11 for calculating the score of the sentence. A glimse of database content is shown in Figure 3.7.



```
, "abhijati": 1, "music": 105, "dns": 3, "mystic": 5, "catchi": 1, "tirabassi": 1, "voyageur": 1, "musik":
rologist": 1, "oxforddnb": 1, "leconteii": 1, "murchison": 1, "kampala": 1, "lomax": 1, "locker": 1, "locke
"etiquett": 3, "unjust": 1, "cookeri": 1, "titanium": 1, "want": 47, "forfar": 1, "pinto": 2, "duniya": 1,
kama": 1, "naevius": 1, "krma": 1, "brassard": 1, "barbra": 1, "wheatear": 1, "1172": 1, "1171": 1, "1170":
 "sedg": 3, "chrysaeto": 1, "pigment": 1, "prodig": 1, "bydgoszcz": 3, "kirner": 1, "deodar": 1, "tulip": 1
fourch": 1, "wherein": 3, "fis": 1, "ligand": 1, "fit": 20, "fix": 11, "fia": 2, "menachem": 1, "vajol": 1,
t": 1, "antun": 1, "worringen": 1, "cdbabi": 1, "realign": 2, "sixteen": 9, "preemin": 2, "arrog": 1, "bart
om": 1, "accommod": 13, "arrow": 7, "helsingborg": 1, "burial": 8, "allah": 1, "allan": 5, "393": 3, "392":
 "395": 3, "394": 2, "allay": 1, "398": 2, "golem": 1, "cinek": 1, "oprah": 1, "smirk": 1, "mason": 5, "bal
utburst": 1, "39b": 1, "strath": 1, "zemstvo": 1, "renegoti": 1, "berwick": 2, "gameplay": 5, "dolichonyx":
castagn": 1, "invigor": 1, "vertebr": 1, "ndalsbron": 1, "kalkudah": 1, "battleboro": 1, "neminathan": 1, "
euben": 1, "master": 35, "taxon_id": 2, "berlioz": 1, "eboracum": 1, "ampl": 1, "ahmet": 1, "sheen": 1, "q4
l": 2, "longx": 1, "miscellanea": 1, "spectrum": 5, "ugli": 2, "mishra": 1, "vidri": 1, "fireroom": 1, "doz
 "paparian": 1, "mouth": 9, "catharus": 1, "bradford": 4, "xuf4t43kkcxr6whpynhibg": 1, "singer": 37, "apeng
"ranulph": 1, "tech": 10, "fitzharding": 1, "scream": 2, "rediscoveri": 1, "cheapli": 1, "teresa": 5, "dick
: 1, "apach": 1, "restaur": 14, "rico": 3, "lube": 1, "lepidoptera": 12, "rick": 7, "rich": 21, "lubo": 1,
 1, "plate": 13, "anni": 2, "iata": 2, "winterburn": 1, "ceil": 1, "platt": 4, "yersek": 1, "ceip": 1, "pop
oarder": 2, "seproxetin": 1, "patch": 2, "superfamili": 3, "transcrani": 1, "gobi": 1, "baccassino": 1, "pi
stin": 1, "jabara": 1, "pinoy": 1, "dishwash": 1, "48th": 2, "heterogen": 1, "melkbosrand": 1, "irv": 4, "l
, "irl": 1, "zinman": 1, "ira": 1, "irb": 1, "ire": 4, "khumeh": 1, "tortora": 1, "margaretvill": 1, "exten
ns": 32, "tengbjerg": 1, "extent": 10, "batulab": 1, "mcgurri": 1, "unemploy": 6, "khandan": 1, "gehr": 1,
 1, "southeastern": 9, "assur": 11, "gopher": 1, "scistud": 1, "kulissen": 1, "assum": 22, "fra": 2, "union
kl": 1, "muck": 2, "muci": 1, "much": 80, "tallest": 3, "finsburi": 1, "1679": 1, "distast": 1, "regul": 13
 1, "jaunanna": 1, "ansgar": 1, "spin": 6, "davi": 22, "wildcat": 1, "noland": 1, "limeston": 3, "intak": 1
42818": 1, "covenat": 1, "gentrifi": 2, "anger": 7, "tacna": 1, "eighteen": 3, "gandhinagar": 1, "musicians
"recover": 1, "leiva": 1, "c6sfl": 1, "gutesa": 1, "honv": 1, "split": 20, "jungl": 4, "kidd": 1, "pricipal
: 1, "boiler": 2, "supper": 4, "ainderbi": 1, "tumbadora": 1, "izgubljeni": 1, "academia": 2, "crosscutt":
 "airbal": 1, "haberdasheri": 1, "airbag": 1, "20compani": 1, "bradden": 1, "figallo": 1, "smc": 2, "shiit"
1, "hac": 1, "had": 250, "watercolour": 1, "zilup": 1, "godlik": 1, "hay": 12, "endur": 6, "haq": 1, "har":
```

Fig. 3.7: The Database for TF-IDF: Stemmed words are stored with their corresponding occurrence

7. **Positive Keywords/Negative Keywords**: This is an important feature of the summarization. It is used for generating customizable summaries based on user requirements. For input keywords, similarity is calculated using `Wordnet` data. For each sentences, a word is selected which has the maximum similarity with the input keywords and then score of the sentence is calculated using the selected word.

   If the keyword is positive then,

$$Score(s) = 1 + \text{similarity value} \tag{3.12}$$

   If the keyword is negative then,

$$Score(s) = 1 - \text{similarity value} \tag{3.13}$$

3.2.3 Score Calculation

Final scores of each sentences are calculated by multiplying the value of each feature for a sentence as shown in equation 3.14

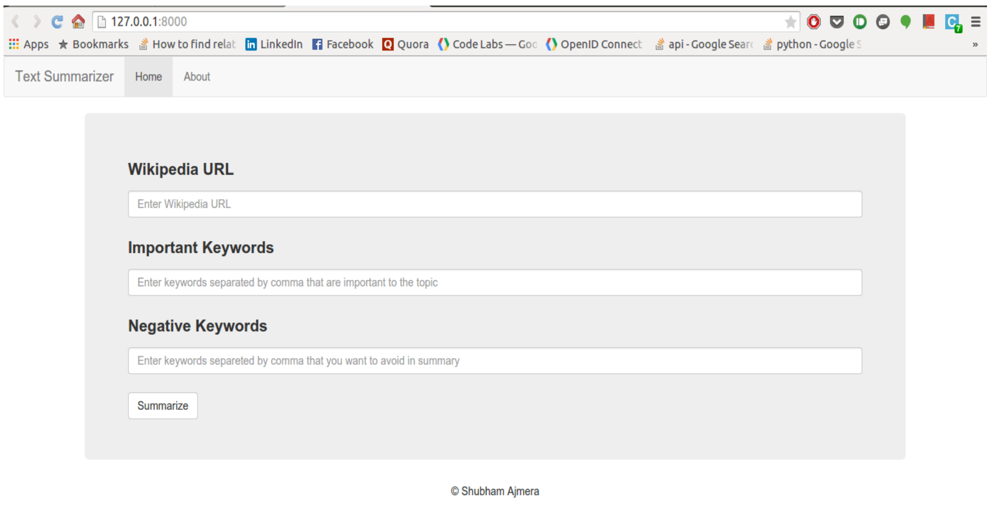
$$Score(s) = \prod_{f_i \in F} f_i \tag{3.14}$$

where **F** is the feature vector for the sentence **s**.

3.2.4 User Interface

3.2.4.1 Querying page

In the webpage, three boxes have been provided for the Wikipedia page for which summary has to be generated, important keywords and negative keywords as shown in Figure 3.8.



Text Summarizer

Wikipedia URL

Enter Wikipedia URL

Important Keywords

Enter keywords separated by comma that are important to the topic

Negative Keywords

Enter keywords separeted by comma that you want to avoid in summary

© Shubham Ajmera

Fig. 3.8: Web version of querying page.

A mobile interface is also developed for the same as shown in Figure 3.9

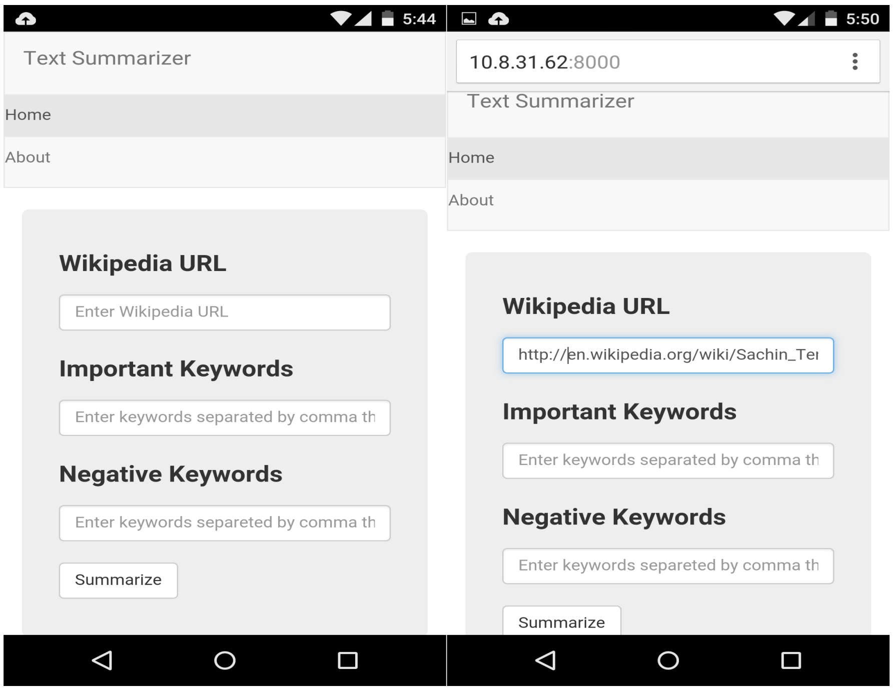
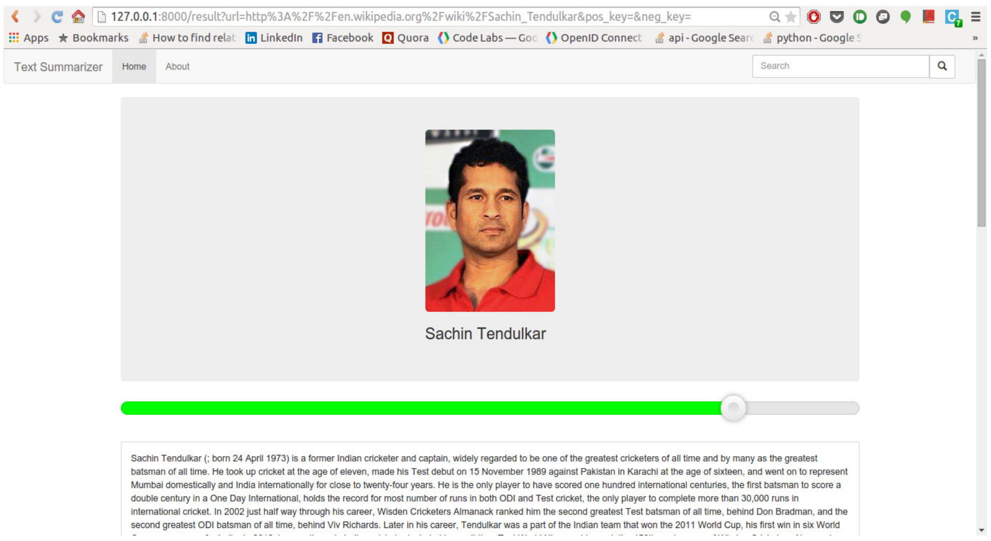


Fig. 3.9: Mobile version of querying page.

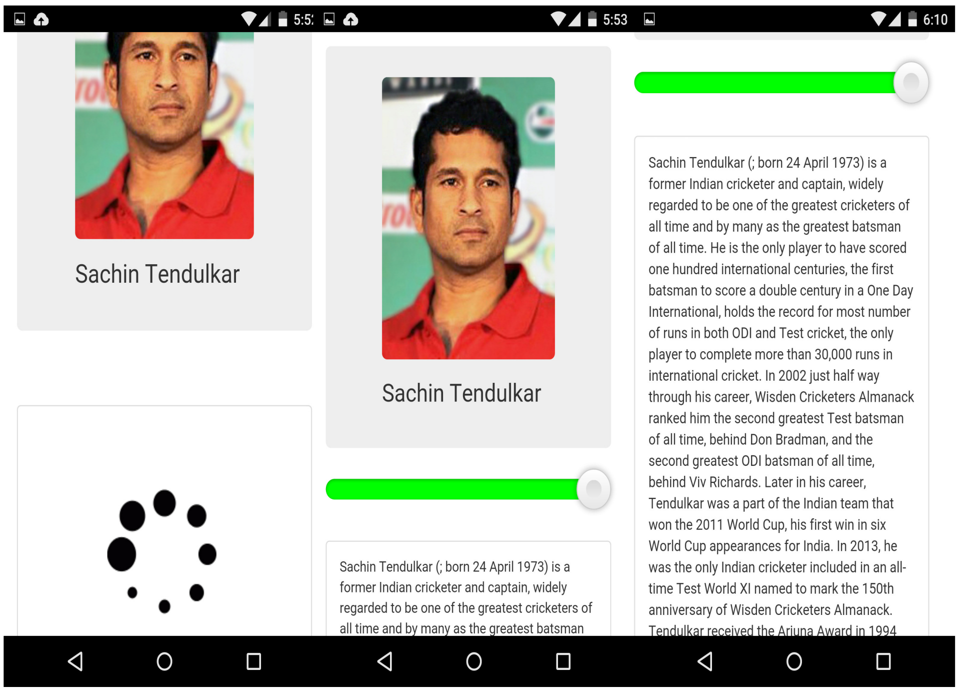
3.2.4.2 Summary Page

This page is built using Bootstrap and Javascript. This page gets GET content from the querying page using which it fetches relevant summary, image and title of the document. It also has a slider using which a user can select the percentage shrinkage according to his requirement as shown figure 3.10 and figure 3.11



Sachin Tendulkar (; born 24 April 1973) is a former Indian cricketer and captain, widely regarded to be one of the greatest cricketers of all time and by many as the greatest batsman of all time. He took up cricket at the age of eleven, made his Test debut on 15 November 1989 against Pakistan in Karachi at the age of sixteen, and went on to represent Mumbai domestically and India internationally for close to twenty-four years. He is the only player to have scored one hundred international centuries, the first batsman to score a double century in a One Day International, holds the record for most number of runs in both ODI and Test cricket, the only player to complete more than 30,000 runs in international cricket. In 2002 just half way through his career, Wisden Cricketers Almanack ranked him the second greatest Test batsman of all time, behind Don Bradman, and the second greatest ODI batsman of all time, behind Viv Richards. Later in his career, Tendulkar was a part of the Indian team that won the 2011 World Cup, his first win in six World

Fig. 3.10: Web version of summary page.



Sachin Tendulkar (; born 24 April 1973) is a former Indian cricketer and captain, widely regarded to be one of the greatest cricketers of all time and by many as the greatest batsman

Sachin Tendulkar (; born 24 April 1973) is a former Indian cricketer and captain, widely regarded to be one of the greatest cricketers of all time and by many as the greatest batsman of all time. He is the only player to have scored one hundred international centuries, the first batsman to score a double century in a One Day International, holds the record for most number of runs in both ODI and Test cricket, the only player to complete more than 30,000 runs in international cricket. In 2002 just half way through his career, Wisden Cricketers Almanack ranked him the second greatest Test batsman of all time, behind Don Bradman, and the second greatest ODI batsman of all time, behind Viv Richards. Later in his career, Tendulkar was a part of the Indian team that won the 2011 World Cup, his first win in six World Cup appearances for India. In 2013, he was the only Indian cricketer included in an all-time Test World XI named to mark the 150th anniversary of Wisden Cricketers Almanack. Tendulkar received the Arjuna Award in 1994

Fig. 3.11: Different screenshots of mobile interface.

3.2.4.3 Webpage interface for non-Wikipedia Context

An interface for non-wikipedia context is also developed in which data can be put manually which includes document's title, sections' heading and its content. According to the need, sections can be added dynamically.

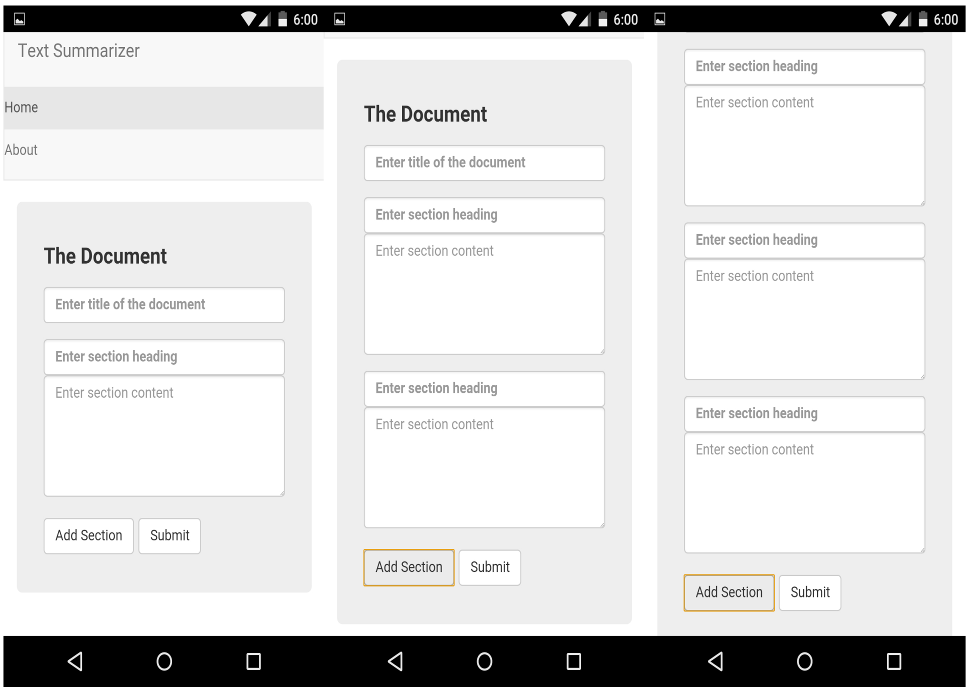


Fig. 3.12: Mobile interface for quering page of non-Wikipedia content

3.2.5 Features

1. Variable length summaries can be generated.

2. Custom summary option is provided to user to remove unwanted topics.

3. Multithreaded feature extraction.

4. Stores summaries of articles so to provide quick results in future.

Chapter 4

Experimental Results

For quality comparisons of summaries generated by summarizers, we have manually created summaries of various Wikipedia articles and anime data. We have generated summaries of 10 Wikipedia articles and 10 anime article. The comparison is done between the Wikipedia Articles and Genetic Algorithm summarizer using the manually generated summaries of Wikipedia articles while in between GAS and Word2Vec summarizer using summaries of animes articles. Also, the comparisons are performed between the summarizers and MS-Word summarizer. Results and explainations of these experiments are as follows:

1. **GA summarizer VS Wikipedia summarizer with CR 30%**

   In this comparison, the output of the generated summary is compared with the hand-written summaries of Wikipedia articles using the equation 4.1. Higher the intersection of sentences between the summaries, higher will be the scores.

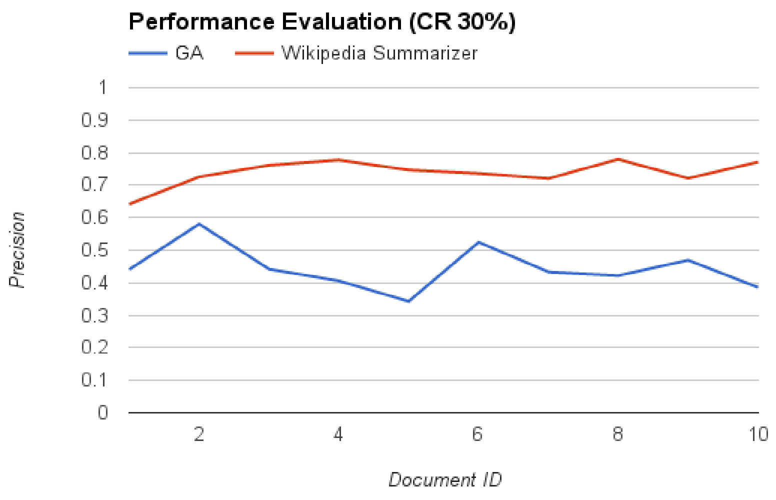


Fig. 4.1: Comparison between summarizer using GA approach and Wikipedia summarizer with Compression rate of 30% against the hand-written summary.

By observing the figure 4.1, we have found out that the average rate of intersection of GA is coming out in between 0.4 and 0.5 with high deviation while the Wikipedia summaries have shown more consistency with values ranging from 0.7 to 0.8.

$$Score(s1, s2) = \frac{\#(sentences\ in\ s1 \cap sentences\ in\ s2)}{\#(sentences\ in\ s2)} \tag{4.1}$$

2. **GA summarizer VS Wikipedia summarizer with CR 20%**
   Again by observing the graph in the figure 4.2, it can be said that Wikipedia summarizer is giving higher percentage of matching as compared to the summarizer using GA approach. However, with comparison to the graph in figure 4.1, we can see that percentage similarity between the hand-written summaries and the summaries generated by two summarizers has decreased.

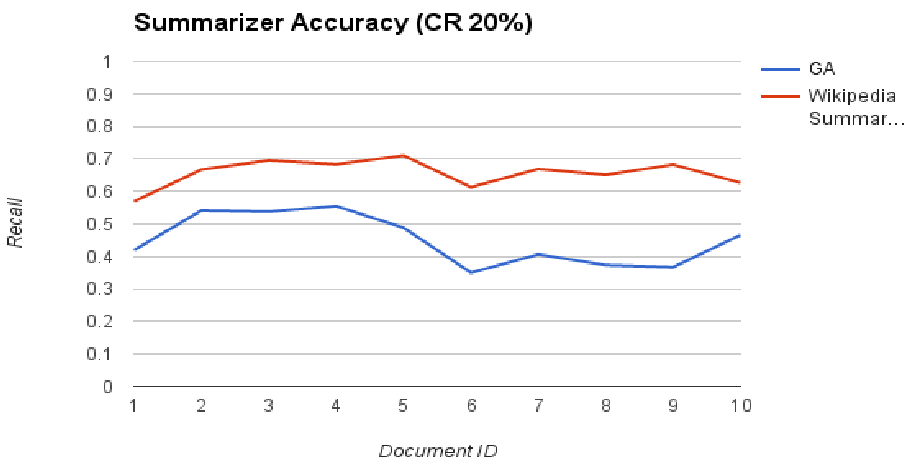


Fig. 4.2: Comparison between summarizer using GA approach and Wikipedia summarizer with Compression rate of 20% against the hand-written summary.

3. **Summarizer using GA approach VS Word2Vec Summarizer**

   For the comparison of these two approaches, 10 hand-written summaries of anime articles are used and scores are calcuated using the equation 4.1. It can be observed from the graph that there is no much difference between the outcome of the two summarizer and the range is varying from 0.35 to 0.6.

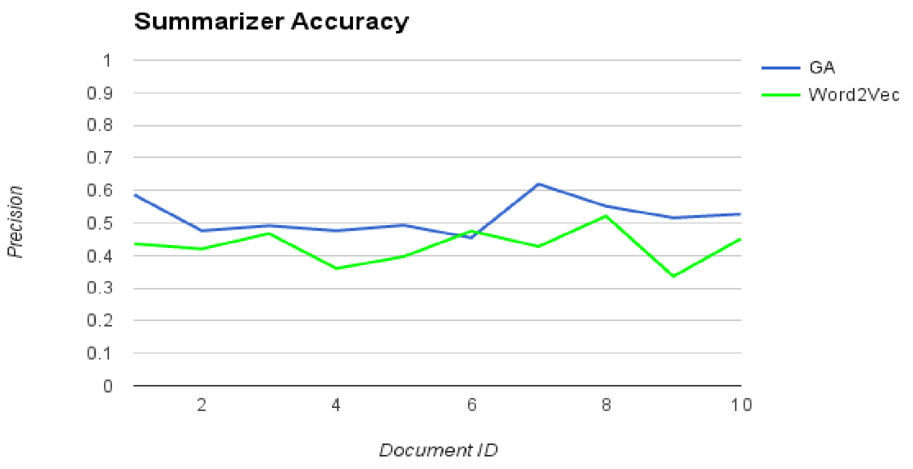


Fig. 4.3: Comparison between The GA summarizer and the Word2Vec summarizer.

4. **Wikipedia Summarizer vs Microsoft Word Summarizer**

   For the comparison of the summarizers, the Wikipedia articles are fetched to Microsoft Word and summaries are compared with the equation 4.1. In Contrast to the comparison with the hand-written summaries where the Wikipedia summarizer had shown very less deviation with a high precision value, has shown a wide variation between the range 0.38 to 0.6 as shown in graph in figure 4.4.

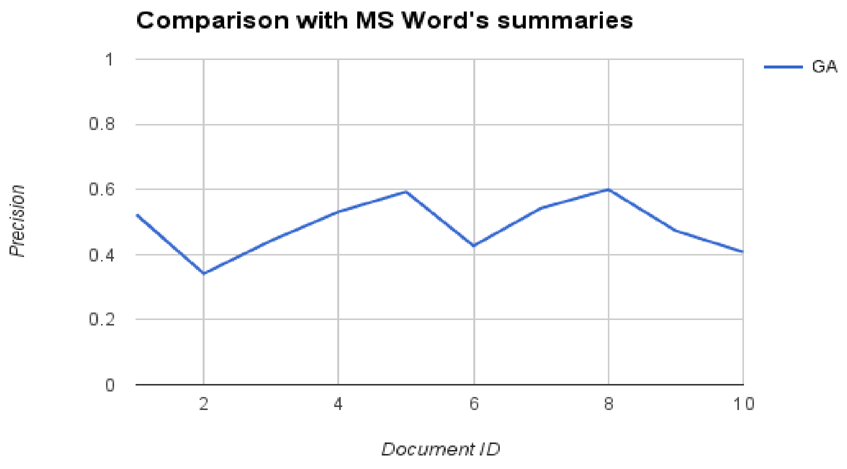


Fig. 4.4: Comparison of summaries generated between Wikipedia summarizer and Microsoft Word.

5. **GA Summarizer vs Microsoft Word Summarizer**

   The summarizer has shown similarity in results with the Wikipedia summarizer as shown in graph in figure 4.5. There is a moderate deviation with a low percentage of similarity between 0.45 to 0.55.

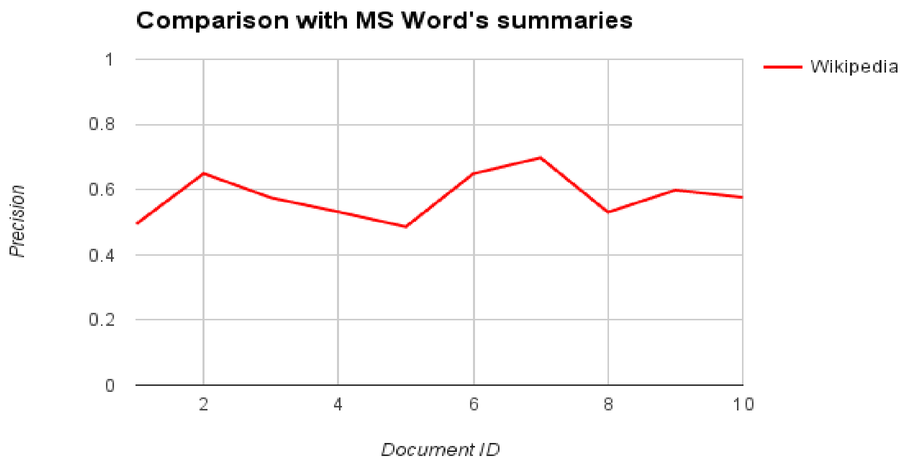


Fig. 4.5: Comparison of summaries generated between GA summarizer and Microsoft Word.

Chapter 5

Conclusion and Future Work

In this project, we have implemented various techniques for extraction-based text summarization and experimented their performance analysis using various datasets.

5.1 Conclusion

1. The summarizer built using genetic algorithms has shown satifactory results for wide variety of documents.
2. The Wikipedia summarizer, though not quite generic, has shown excellent results for generating summaries for Wikipedia pages.
3. The Word2Vec summarizer is very much dependent on the training data but shows good result as compared to other two for the data which is closely knitted.

5.2 Future Score

1. Building a single system which can automatically identify the type of input text and use the best method out of these three technique to generate summary.
2. Improving accuracy of features like name entity identifier, location finder etc.
3. Extending the domains for Word2Vec summarizer by training it using various other datasets in diverse domains.

References

[1] G. Sizov, “Extraction-based automatic summarization: Theoretical and empirical investigation of summarization techniques,” 2010.

[2] M. A. Fattah and F. Ren, “Automatic text summarization,” *World Academy of Science, Engineering and Technology*, vol. 37, p. 2008, 2008.

[3] “About wordnet - wordnet - about wordnet,” https://wordnet.princeton.edu/, accessed: 2015-05-15.

[4] “word2vec - tool for computing continuous distributed representations of words. - google project hosting,” https://code.google.com/p/word2vec/, accessed: 2015-05-15.

[5] J. Ramos, “Using tf-idf to determine word relevance in document queries,” in *Proceedings of the first instructional conference on machine learning*, 2003.

[6] W. T. Chuang and J. Yang, “Extracting sentence segments for text summarization: a machine learning approach,” in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2000, pp. 152–159.

[7] S. Ryang and T. Abekawa, “Framework of automatic text summarization using reinforcement learning,” in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, 2012, pp. 256–265.

[8] D. Hingu, D. Shah, and S. S. Udmale, “Automatic text summarization of wikipedia articles,” in *Communication, Information & Computing Technology (ICCICT), 2015 International Conference on*. IEEE, 2015, pp. 1–4.

[9] H. P. Edmundson, “New methods in automatic extracting,” *Journal of the ACM (JACM)*, vol. 16, no. 2, pp. 264–285, 1969.

[10] “Wikipedia - wikipedia, the free encyclopedia,” http://en.wikipedia.org/wiki/Wikipedia, accessed: 2015-05-15.

Curriculum Vitae

Name: Shubham Ajmera

Date of birth: 09 October, 1992

Education qualifications:

- [2011 - 2015] Bachelor of Technology (B. Tech),
  Computer Science and Engineering,
  Indian Institute of Technology Mandi
  Mandi, Himachal Pradesh, India.

Permanent address:

'Keluka Bhawan',
3/219, Near CTS Bus Stand, Sanganer
Jaipur, Rajasthan, India, 302029.
Ph: +918894849404