San Jose State University
SJSU ScholarWorks

Master's Projects | Master's Theses and Graduate Research

Spring 2014

Text Summarization for Compressed Inverted Indexes and Snippets

Mangesh Dahale
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

 Part of the Computer Sciences Commons

Recommended Citation

Dahale, Mangesh, "Text Summarization for Compressed Inverted Indexes and Snippets" (2014). *Master's Projects*. 361.
DOI: https://doi.org/10.31979/etd.ekkr-4urb
https://scholarworks.sjsu.edu/etd_projects/361

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Text Summarization for Compressed Inverted Indexes and Snippets

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Mangesh Dahale

May 2014

© 2014
Mangesh Dahale
ALL RIGHTS RESERVED

The Designated Project Committee Approves the Project Titled

Text Summarization for Compressed Inverted Indexes and Snippets

by

Mangesh Dahale

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2014

Dr. Chris Pollett	Department of Computer Science
Dr. Sami Khuri	Department of Computer Science
Dr. Ronald Mak	Department of Computer Science

ABSTRACT

Text Summarization for Compressed Inverted Indexes and Snippets

by Mangesh Dahale

Text summarization is a technique to generate a concise summary of a larger text. In search engines, Text summarization can be used for generating compressed descriptions of web pages. For indexing, these can be used rather than whole pages when building inverted indexes. For query results, summaries can be used for snippet generation. In this project, we research on several techniques of text summarization. We evaluate these techniques for quality of the generated summary and time required to generate it. We implement the technique chosen from the evaluation in Yioop, an open source, PHP-based search engine.

ACKNOWLEDGEMENTS

It gives me immense pleasure to present my acknowledgement, a token of appreciation to all the persons involved directly and indirectly with my project.

I take this opportunity to express my profound sense of gratitude and inestimable respect to my project advisor, Dr. Chris Pollett, for his continuous guidance and support throughout the project.

I am grateful to my committee members, Dr. Sami Khuri and Dr. Ronald Mak, for their suggestions without which this project would not have been possible.

Table of Contents

List of Figures

List of Tables

1. Introduction

Search engines are often the first source of information when we want to do any research. To get this information, a search engine should understand our query and give results relevant to the query. Summarization is one of the key steps for obtaining these relevant results from the system. We will implement this summarization feature in a search engine to improve its ability to obtain these relevant results from the system. The major challenge in summarization lies in distinguishing the more informative parts of a document from the less informative ones. Text summarization is a technique to generate a concise summary of a larger text. In search engines, text summarization can be used for generating compressed descriptions of web pages. For indexing, these can be used rather than whole pages when building an inverted indexes. For query results, summaries can be used for snippet generation.

Text summarization is usually described as a three-step process: selection of salient portions of text, aggregation of the information for various selected portions and abstraction of this information, and finally, presentation of the final summary text. This process can be used in many applications such as information retrieval, intelligence gathering, information extraction, text mining, and indexing[5].

In this project, we experimented with three summarization techniques for the Yioop search engine. Yioop is an open source, PHP search engine which is designed to allow users to produce indexes of a web-site or a collection of web-sites. In the initial stage of the project, research was done on the text summarization topics to find out which methods are being used for text summarization and study three methods in depth so that we can implement them. Then, we evaluated the performance of three summarization techniques for which we created a sample document set so that we can compare these three methods and choose the one with high performance and which is best suited for Yioop search engine. Finally, we performed some experiments to compare the new summarizer with the previous summarizer in Yioop search engine. Also, we experimented the effects on speed using compiler versus interpreter for running these summarizers.

The rest of the report is organized as follows. Chapter 2 introduces the similarity measures that we have used for summarization. Chapter 3 contains detailed explanations of the three summarization techniques we have implemented. Chapter 4 explains the implementation of those three summarization techniques. In Chapter 5, we evaluated the performance of three summarization techniques to choose the technique which has a good performance and is best suited for the Yioop search engine. Chapter 6 contains the steps that we have performed to integrate the chosen summarizer into the Yioop search engine.

Chapter 7 contains the experiments that we have performed on the integrated summarizer. Chapter 8 concludes the project and also discusses about the future work in this project.

2. Similarity measures

A similarity measure gives us the degree of similarity between two objects[13]. Summarization techniques often use similarity measures to find the similarity between the sentences in the text. The three methods that we implemented to select the best method to integrate in Yioop use similarity measures to identify the more informative parts of the document from the less informative parts.

We used two similarity ranking algorithms in this project. The first summarization technique, the intersection method, uses the TextRank[8] algorithm as a similarity measure. The second and third method, the centroid method and the TF-ISF method, use the cosine ranking algorithm[2] as a similarity measure.

2.1. TextRank for Sentence Extraction

In this algorithm, we first represent the complete text as a graph. As we have to get the similarity of each sentence with every other sentence in the text, we represent the sentences as vertices of this graph. We measure the similarity between sentences by examining the content overlap between every pair of sentences. The content overlap can be simply measured by comparing the terms in both sentences. This relation between two sentences is also known as process of recommendation. When the contents of two sentences overlap that means they

share some common concepts, so one sentence recommends to the reader the other sentence which also has the same concepts in it [8].

For the long sentences in the text, we use a normalization factor, where we divide the result of the content overlap by length of each sentence. The result we get after these operations is the similarity score of two given sentences. This similarity score is represented on the graph as a weighted edge between two vertices representing those two given sentences. After calculating the similarity score of all the sentences, we get a highly connected graph as a result. For each sentence, we add the similarity scores of that sentence with every other sentence in the complete text to get a total score for that sentence. Finally, we sort the sentences in descending order of their total score to get the sentences with highest scores at the top. We include these sentences in our summary until the summary length threshold is reached [8].

2.2. Cosine similarity measure

Cosine similarity measure [2] is based on Bhattacharya's distance [1], which is an inner product of the two vectors divided by the product of their length. Given two vectors, we calculate the similarity between these two vectors by comparing the angle between them. The smaller the angle, the more similar the vectors [13].

Given two $|V|$-dimensional vectors $\overrightarrow{x} = \langle x_1, x_2, \dots, x_{|V|} \rangle$ and $\overrightarrow{y} = \langle y_1, y_2, \dots, y_{|V|} \rangle$,

we have [2],

$$\overrightarrow{x} \cdot \overrightarrow{y} = |\overrightarrow{x}| \cdot |\overrightarrow{y}| \cdot \cos(\theta)$$

where $\overrightarrow{x} \cdot \overrightarrow{y}$ represent the inner product between the vectors. This dot product is defined as [2]

$$\overrightarrow{x} \cdot \overrightarrow{y} = \sum_{i=1}^{|V|} x_i \cdot y_i$$

and the length of a vector can be computed by the Euclidean distance formula [2]

$$|\overrightarrow{x}| = \sqrt{\sum_{i=1}^{|V|} {x_i}^2}$$

Given the two vectors v_1 and v_2, the cosine similarity $sim(\overrightarrow{v_1}, \overrightarrow{v_2})$ is calculated as[2],

$$sim(\overrightarrow{v_1}, \overrightarrow{v_2}) = \frac{\overrightarrow{v_1}}{|\overrightarrow{v_1}|} \cdot \frac{\overrightarrow{v_2}}{|\overrightarrow{v_2}|}$$

Cosine similarity measure value lies between 0 and 1. The higher the value, the more similar are the two vectors [2].

In many search engines, cosine similarity measure is used for comparing the query and documents to retrieve the documents which are similar to the query. Another use of cosine similarity measure is to get the similar pages for a particular page in the search results. In this case, we replace the query vector by document vector [2].

2.2.1. TF-IDF

The vectors we use to calculate the cosine similarity contains the TF-IDF weights. Here TF is the Term Frequency. This function measures how common the term is in the document and IDF is inverse document frequency which relates the document frequency to the total number of documents in the corpus (N) [2]. Formulas for calculating the TF and IDF is as follows [2]:

$$TF = \log(f_{t,d}) + 1 \quad if\ f_{t,d} > 0\ and\ 0\ otherwise,$$

$$IDF = \log\left(\frac{N}{N_t}\right)$$

where, $f_{t,d}$ is the frequency of the term t in document d and N_t represents the number of document containing the term t.

After calculating the TF and IDF, we save the TF·IDF weight score into the vector of the given document [2].

3. Study different methods of Text Summarization

We researched the text summarization topic to find out which methods are being used by the search engines for text summarization and studied three methods in depth so that we could implement them and choose one which is best suited for the Yioop search engine. The three methods are as follows: 1. Intersection method 2. Centroid method 3. TF-ISF method.

3.1. Intersection method

We calculate the intersection between two given sentences by simply counting the number of common tokens between them. The higher the common tokens, better the intersection. This method works on the principle that if two sentences have a good intersection, they probably hold the same information. So if one sentence has a good intersection with many other sentences, it probably holds some information from each one of them or in other words, this is probably a key sentence in our text [11]. We use an intersection function to calculate the intersection between two sentences and we create a key-value dictionary, where the sentence itself is the key and the value is the total score.

This method is based on "TextRank – a graph-based approach for text processing" [8]. We applied this model for sentence extraction for our summarizer. For this, we need to build a graph associated with the text where the graph vertices

are representative for the units to be ranked. Here the goal is to rank all the sentences which is why we add them as a vertex in the graph. Edges, in this graph, are the similarity between two sentences where similarity is measured as the function of their content overlap. The content overlap between two sentences can be calculated by simply counting the number of common tokens between given two sentences.[8]

For example, consider the following sentences:

3: BC-HurricaineGilbert, 09-11 339
4: BC-Hurricaine Gilbert, 0348
5: Hurricaine Gilbert heads toward Dominican Coast
6: By Ruddy Gonzalez
7: Associated Press Writer
8: Santo Domingo, Dominican Republic (AP)
9: Hurricaine Gilbert Swept towrd the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains, and high seas.
10: The storm was approaching from the southeast with sustained winds of 75 mph gustingto 92 mph.
11: "There is no need for alarm," Civil Defense Director Eugenio Cabral said in a television alert shortly after midnight Saturday.
12: Cabral said residents of the province of Barahona should closely follow Gilbert's movement.
13: An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo.
14. Tropical storm Gilbert formed in the eastern Carribean and strenghtened into a hurricaine Saturday night.
15: The National Hurricaine Center in Miami reported its position at 2 a.m. Sunday at latitude 16.1 north, longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo.
16: The National Weather Service in San Juan, Puerto Rico, said Gilbert was moving westard at 15 mph with a "broad area of cloudiness and heavy weather" rotating around the center of the storm.
17. The weather service issued a flash flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday.
18: Strong winds associated with the Gilbert brought coastal flooding, strong southeast winds, and up to 12 feet to Puerto Rico's south coast.
19: There were no reports on casualties.
20: San Juan, on the north coast, had heavy rains and gusts Saturday, but they subsided during the night.

21: On Saturday, Hurricane Florence was downgraded to a tropical storm, and its remnants pushed inland from the U.S. Gulf Coast.
22: Residents returned home, happy to find little damage from 90 mph winds and sheets of rain.
23: Florence, the sixth named storm of the 1988 Atlantic storm season, was the second hurricane.
24: The first, Debby, reached minimal hurricane strength briefly before hitting the Mexican coast last month.

We generate a graph for these sentences as follows:

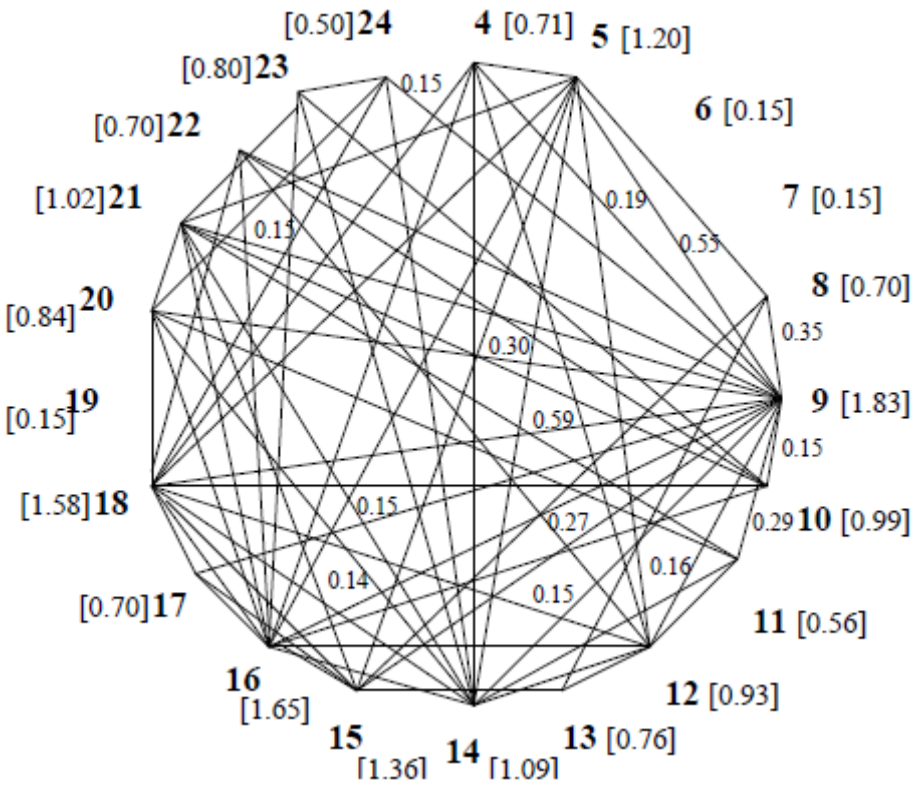

Figure 1: Sample graph build for sentence extraction using TextRank algorithm[8]

3.2. Centroid method

For this method, we start with the document we wish to summarize. This method gives us a word cloud while generating a summary for that document. To generate a word cloud, we use a technique called topic detection and tracking which is used in MEAD (multi-document summarizer) [9] to find all the documents with same topic and adding them to a cluster.

3.2.1. What is centroid?

"A centroid is a set of words that are statistically important to a cluster of documents. As such, centroids could be used both to classify relevant documents and to identify salient sentences in a cluster."[9]

In this method, we first find the centroid of the document, in other words, we find the main topic of the document. Then we calculate the TF-IDF score of each document in the cluster so that we can get the weight of that document in a cluster.

After calculating weights, we calculate the cosine similarity between the centroid (main topic of the document) and given document by the following formula [9]:

$$sim(D,C) = \frac{\sum_k (d_k \cdot c_k \cdot idf(k))}{\sqrt{\sum_k (d_k)^2}\sqrt{\sum_k (c_k)^2}}$$

where d_k represents the weight of the given term *k* in document D and c_k represents the weight of the given term in centroid C.

After getting similarity score between the centroid and each document, we add the document which have the score within a threshold to the cluster.

3.3. TF-ISF method

In this method, we represent the document as a weighted vector of TF·ISF as we did in centroid method. We then calculate the cosine similarity of each sentence with every other sentence from the document by using the following formula [2]:

$$sim(s_i, s_j) = \frac{\sum_{k=1}^{m} w_{ik} w_{jk}}{\sqrt{\sum_{k=1}^{m} w_{ik}^2 \cdot \sum_{k=1}^{m} w_{jk}^2}}, \quad i, j = 1,...., n$$

where w_{ik} represents the weight of the term *k* in the sentence i.

With cosine similarity scores, we also calculate the coverage and the diversity of the summary. We enforce coverage and diversity to make the summary more informative and concise by ensuring that it covers all the topics from the document and removes redundant information from the summary.

3.3.1. Diversity

In diversity [10], we ensure that the sentences selected do not have the same information. Diversity is an important issue since sentences from different documents might convey the same information. A high quality summary should be informative and compact [10].

We model diversity with the following objective function [10]:

$$f_{\text{diver}} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (1 - \text{sim}(s_i, s_j))x_i x_j$$

Higher values of $f_{diver}(\cdot)$ correspond to lower overlaps in content between sentences s_i and s_j [10].

3.3.2. Coverage

In coverage [10], we ensure that the sentences in the summary cover all the topics from the document. We attempt to find a subset of the sentences $S = \{s_1, s_2, ..., s_n\}$ that covers the main content of the document collection [10].

Generally, a document contains a variety of information centered on a main topic, and covers different aspects of the main topic. In coverage, we ensure that all these subtopics are covered in the resulting summary [10]

$$f_{\text{cover}}(X) = \text{sim}(O, O^S) + \text{sim}(O, s_i).$$

Here O and O^S denote the centers of the collection S = {s_1, s_2,..., s_n} and the summary

$$S = \bigcup_{i=1}^{n} s_i \, x_i$$

respectively, where x_i denotes a binary variable of the presence of sentence s_i in the summary and $\cup$ is the concatenation operation. Sentence concatenation is an operation of joining the sentences end-to-end. Higher values of $f_{cover}(\cdot)$ correspond to higher content coverage of summary [10].

The k^{th} coordinate o_k of the mean vector O is calculated as [10]:

$$o_k = \frac{1}{n} \sum_{i=1}^{n} w_{ik}$$

and the k^{th} coordinate o_k^S of the mean vector O^S we define as [10]:

$$o_k^S = \frac{1}{|S|} \sum_{s_i \in S} w_{ik}$$

where |S| denotes the number of sentences in summary S and k = 1, . . ., m. [10]

3.3.3. Single Objective function

In general, in a multi-objective optimization problem it is not possible to find a single solution that optimizes all the objectives simultaneously. Therefore, we construct a single objective function [10].

maximize

$$f = \alpha \cdot \frac{f_{cover} + f_{diver}}{2} + (1 - \alpha) \cdot f_{cover} + f_{diver}$$

subject to

$\sum_{i=1}^{n} l_i x_i \leq L$

$x_i \in \{0,1\}, \forall i$

where 'L' is the length of the summary, 'l_i' is the length of the sentence 's_i' and $\alpha \in [0, 1]$ is the weighting parameter, specifying the relative contributions of the arithmetic and harmonic means to the hybrid function. [10]

4. Implementing three methods to evaluate their performances

After studying the above mentioned three methods in depth, we started coding these three methods so that we can evaluate their performance in order to choose the best. Following is the explanation of those three methods:

4.1. Intersection method

For implementation of this method, we divide the complete text into sentences and then all those sentences into terms. For storing all the ranks of each sentence, we created a sentence dictionary which is a collection of key value pairs where key is the sentence itself and value is score of that sentence.

We have implemented the intersection function to calculate the intersection (I) between each sentence and every other sentence in the document as follows [11]:

$$I = \frac{Number\ of\ common\ terms\ in\ two\ sentences}{(Total\ number\ of\ terms\ in\ first\ sentences + Total\ number\ of\ terms\ in\ second\ sentences)/2}$$

The score is calculated based on this intersection. The score of a sentence is the sum of all the intersections between that sentence and every other sentence in the document [11].

To decide the length of the summary, we implemented a graphical slider so that we can specify the length of the summary we want. Now, we start to add the

sentences with the highest scores to the summary until the specified summary length is reached.

4.2. Centroid method

For implementation of this method, we started with formatting the document to remove special characters. This method also generates a word cloud which contains the terms that covers the main theme of the document. Therefore, we have to remove stop words from the document. Then, we have calculated weights of each term in sentences based on term-frequency (TF) and inverse sentence frequency (ISF). Here TF-ISF is a modified version of TF-IDF where every sentence is treated at a document. Each sentence is represented as a weighted vector of TF-ISF scores.

After calculating weights, we took ten terms which have the highest score and showed them on the user interface by changing their font sizes based on their weights in document so that the term having highest weight will appear the biggest among all the other terms.

These ten terms are the centroid of the document. For scoring all the sentences in the document, we calculated the similarity measure between centroid vector and sentence vector. To specify the length of the summary, we also implemented the graphical slider similar to the slider implemented in the intersection method. To generate our final summary, we keep adding the sentences

with the highest similarity according to the centroid method in the summary until the specified summary length is reached.

4.3. TF-ISF method

For implementing this method, we formatted the document and removed stop words like we did in centroid method. Then, we calculated the TF-ISF scores where TF is the term frequency of the term and ISF is the inverse sentence frequency of the term. After calculating the weights of each term for each sentence, we calculated the similarity of each sentence with every other sentence in the document.

This method also enforces coverage and diversity measures to the summary. So, we calculated these two measures separately at first and then created a single objective function which mixes them and generates a summary which has good score. To implement a single objective function, we have used a simple genetic algorithm where we generated an initial population and generated next generation populations based on the coverage and diversity scores from the previously generated population.

5. Evaluate the performance of these three methods to find the best summarization method

5.1. Background

There are several methods to evaluate the text summarization techniques. Generally, evaluation methods for text summarization falls into two main categories: intrinsic and extrinsic [7]. Intrinsic evaluations mainly assess the informativeness and coherence of summaries. Extrinsic evaluations tests the impact of summarization on some other task. We evaluated the three summarization methods using intrinsic evaluation where we compared the summarizer generated summary with the human generated summary [7].

5.2. Evaluation

To evaluate the performances of these methods, we took ten documents related to sports from the Wikipedia and wrote a summary for each document by using our own judgment so that it can be considered as a human generated summary. Then, we ran all three methods on same set of documents. Now, we have both human generated and machine generated summary of each document. Then, we calculated the cosine similarity between the human generated summary and the summary generated by all three methods.

The above procedure gave us the performance of each method for same set of documents. While comparing these methods, we considered two factors, speed and quality of the summary.

The documents we have used for these experiments are as follows:

Doc No.	Document Name	Document Length (in characters)
1	Hockey	98707
2	Cricket	266223
3	SJSU	282666
4	Football	306395
5	Volleyball	216200
6	Cycling	179671
7	Wrestling	135080
8	Shooting	80583
9	Boxing	255909
10	Karate	261855

Table 1: Document set used for experiments

Doc No.	Document Name	Similarity score with Human generated summary		
		Intersection Method	Centroid method	TF-ISF method
1	Hockey	0.62	0.76	0.61
2	Cricket	0.27	0.67	0.26
3	SJSU	0.63	0.70	0.51
4	Football	0.57	0.73	0.52
5	Volleyball	0.37	0.55	0.22
6	Cycling	0.51	0.52	0.50
7	Wrestling	0.76	0.78	0.51
8	Shooting	0.69	0.70	0.70
9	Boxing	0.42	0.42	0.42
10	Karate	0.55	0.68	0.45

Table 2: Similarity score between human generated summary and summarizer generated summary

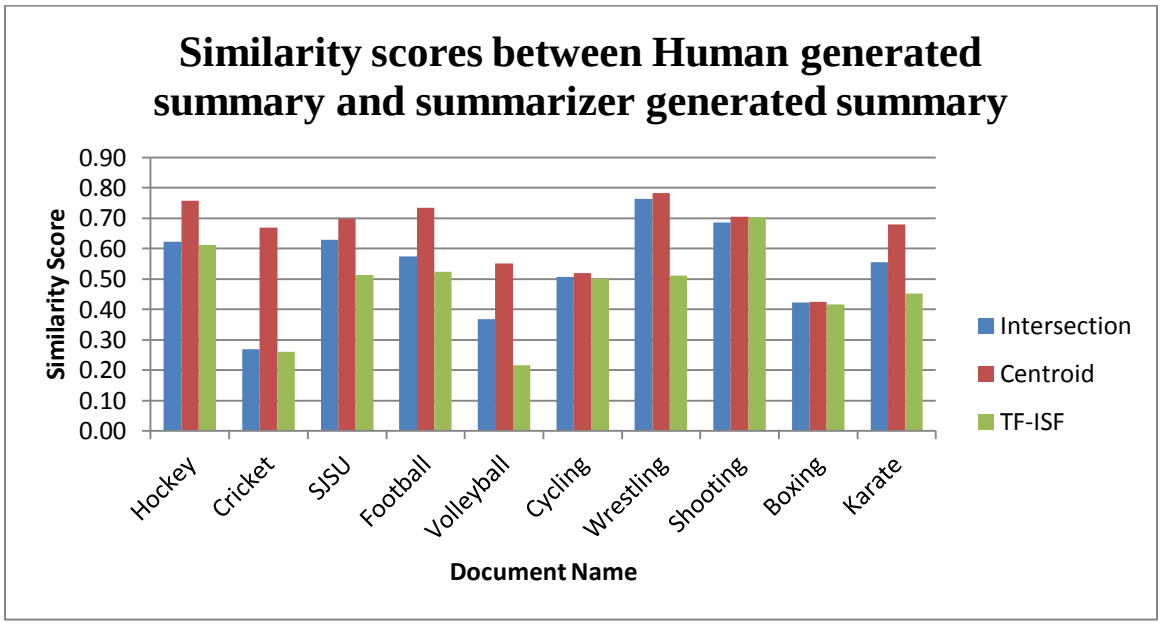

Figure 2: Similarity scores between Human generated summary and summarizer generated summary

Doc No.	Document Name	Time required to extract the summary		
		Intersection Method	Centroid method	TF-ISF method
1	Hockey	0.18	0.28	4.74
2	Cricket	1.33	1.35	21.99
3	SJSU	0.68	1.01	27.73
4	Football	0.68	0.89	19.26
5	Volleyball	0.80	0.85	20.36
6	Cycling	0.77	1.36	32.47
7	Wrestling	0.25	0.43	7.90
8	Shooting	0.01	0.02	0.13
9	Boxing	0.71	1.27	23.85
10	Karate	1.05	1.91	29.17

Table 3: Time required to generate the summary for each of the three methods

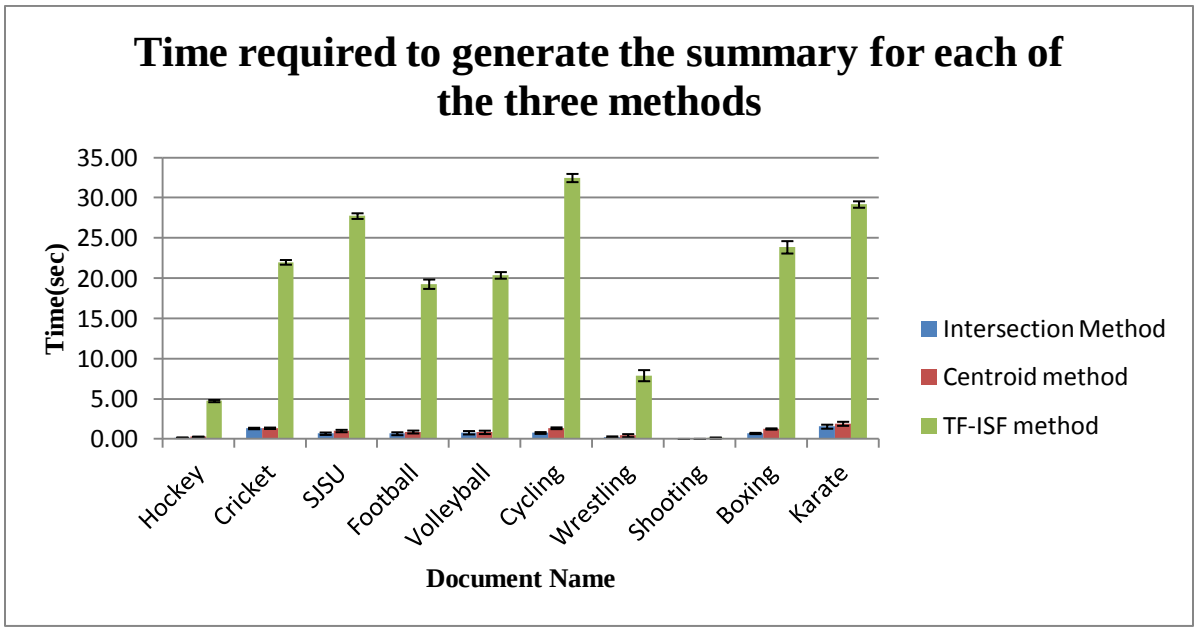

Figure 3: Time required to generate the summary for each of the three methods

Time required to generate the summary by TF-ISF method was not practical, so we compared only intersection and centroid method.

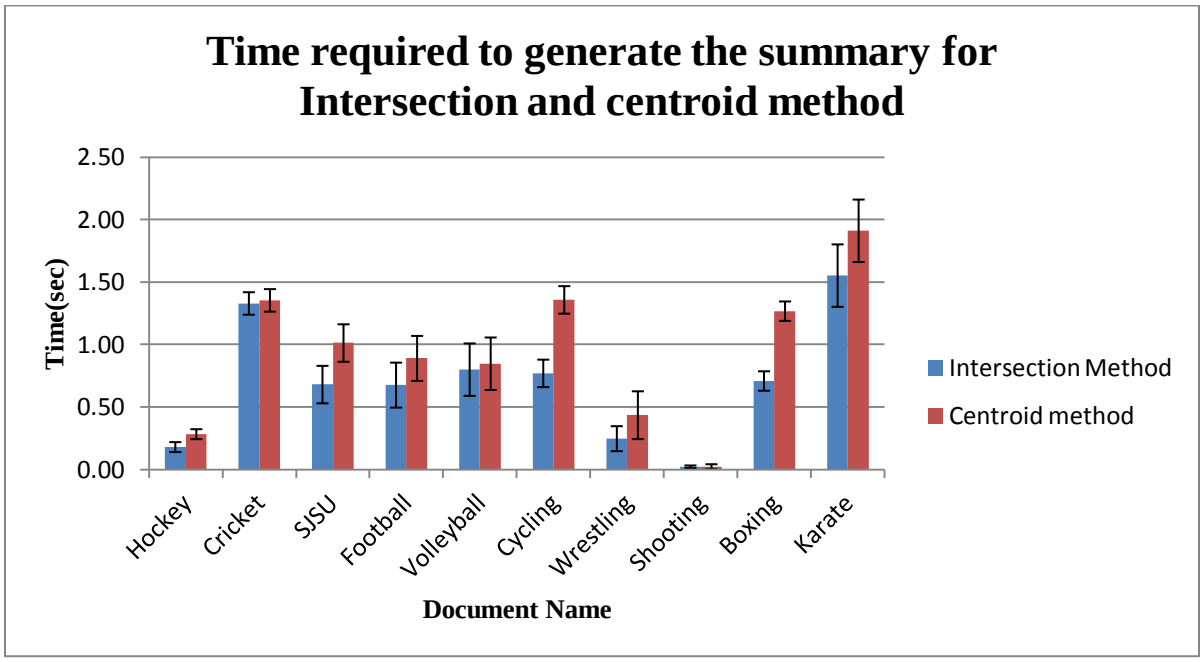

Figure 4: Time required to generate the summary for Intersection and centroid method

In terms of speed, intersection method is at the top and in terms of quality of the summary, centroid method is at the top. Also, the centroid method has the feature of creating a word cloud which can be used to show in search results which will help users to identify the main theme of the webpage in the result.

According to the above performance analysis, we have decided to implement centroid method for Yioop search engine.

6. Integrating the centroid summarizer into Yioop search engine

6.1. Integration

After evaluating performances of three methods and choosing the right method for Yioop, we started integrating the centroid based summarizer. While integrating this summarizer we needed to make sure we are not disturbing the current summarizer in Yioop. We implemented a feature to switch between the two summarizers, Basic (the previous summarizer in Yioop) and Centroid (the new summarizer).

Yioop will use the selected summarizer while crawling the web pages from the internet. When the summarizer is set to "Centroid", all the web pages will be fed to the centroid based summarizer which will create a concise summary and a word cloud from it. This word cloud also gets stored with the summary and is used on the search results page besides the URL of the web page.

To change the summarizer in Yioop, you can follow the steps listed below:

1. Login into Yioop

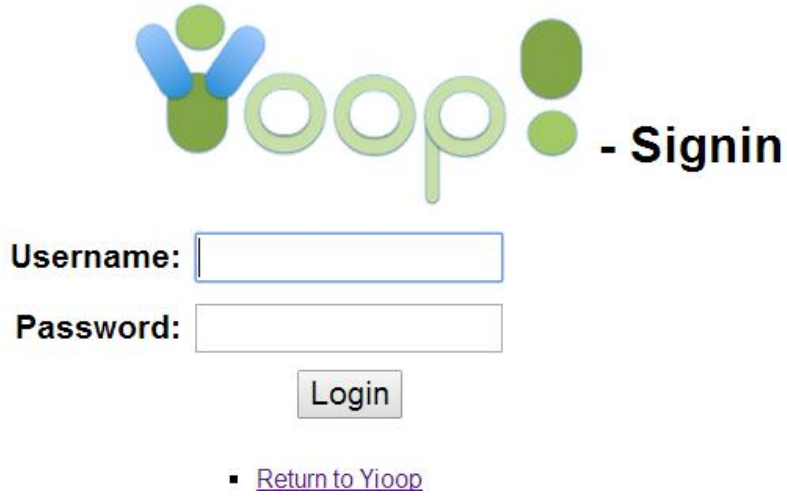

Figure 5: Yioop login page

2. Click on "Manage Crawl"

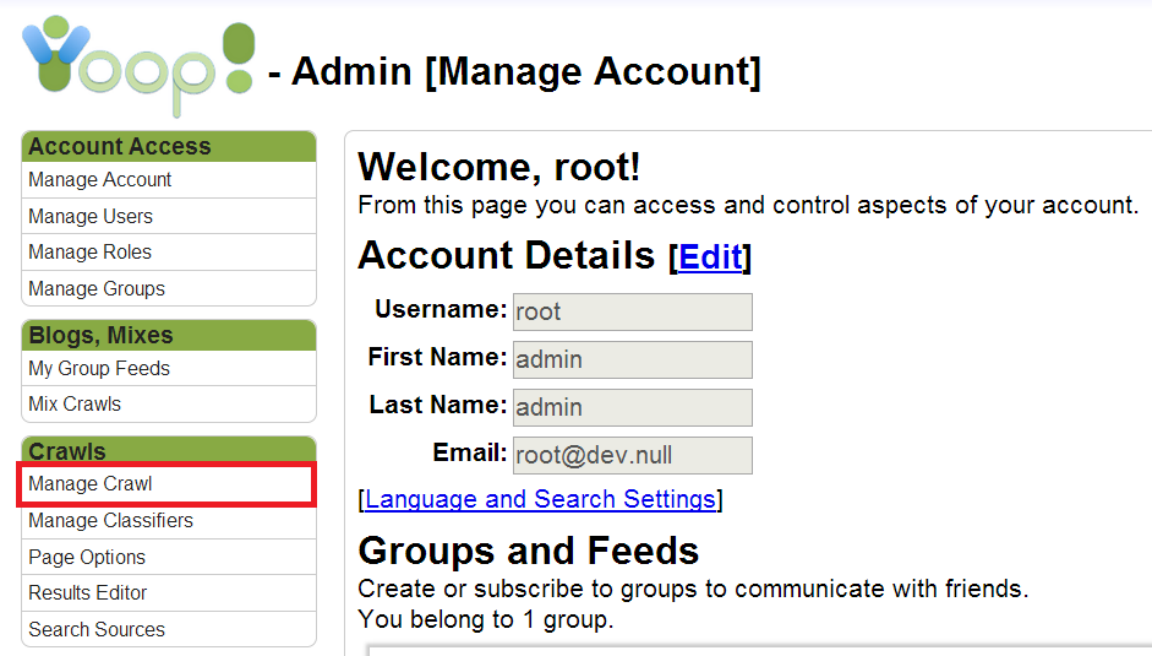

Figure 6: Yioop admin manage account page

3. Click on the "Options" link in "Create Crawl" section to modify the crawl options.

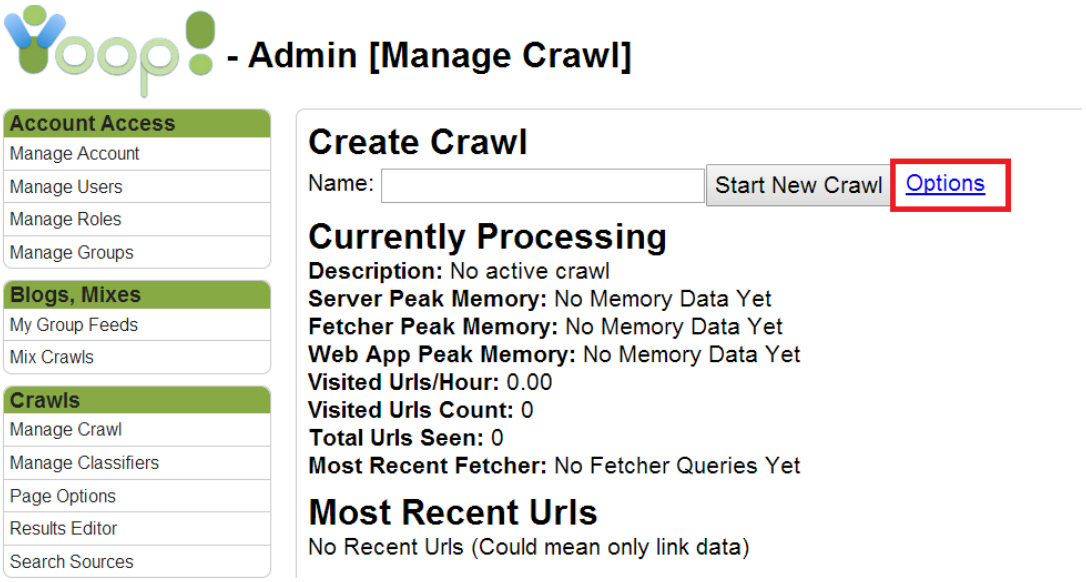

Figure 7: Yioop manage crawl page

4. Select the "Centroid" in Summarizer dropdown list as shown in figure.

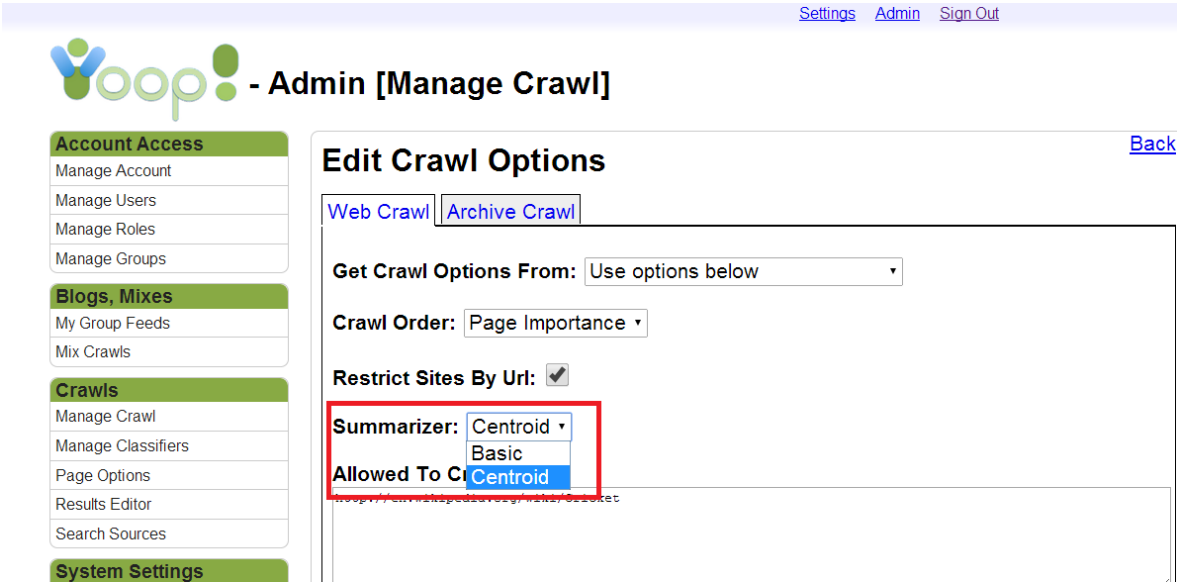

Figure 8: Feature to switch between the two summarizers

6.2. Word Cloud

Word cloud can be defined as a visual representation of keywords from the webpage [4]. These keywords are the important words from the webpage which describes the complete webpage just by displaying some keywords from that webpage. We often do not want to read the complete webpage to get the idea about the theme of the content. The word cloud helps us to get the overall picture of the complete webpage so that we don't need to read the complete webpage. These words in the word cloud are shown in different styles to show their importance in that webpage. Suppose there are five words in the word cloud. To show the importance of each word in the webpage we use different font sizes and/or colors. The word with highest importance is displayed with biggest font size among those five or given a darkest color [4].

A weighted list is a type of word cloud used in geographic maps which represents the relative sizes of countries and cities with relative font sizes. Different font sizes and colors are used to show the association between words and features in map.[6]

The following screenshot shows how used the word cloud in the Yioop search results page.

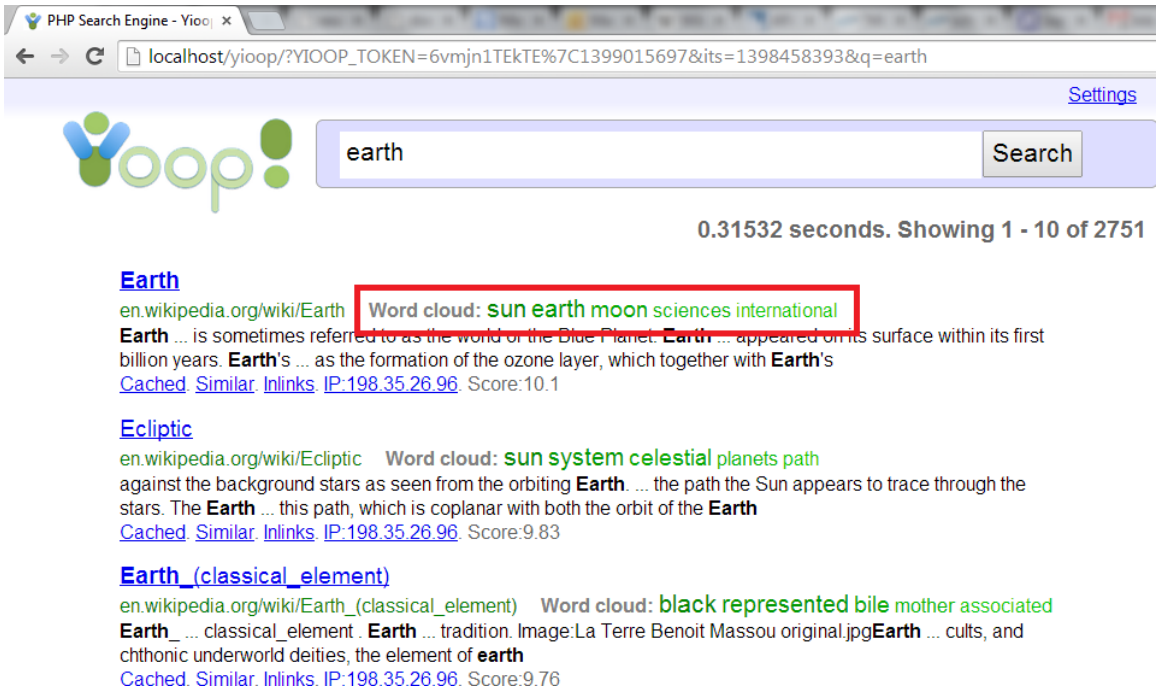

Figure 9: Word cloud in Yioop search results page

Here the user will get the theme of the webpage even before clicking on the link. There are top ten results on a search page and a word cloud associated with each one. After entering the query, user can look at all these ten word clouds and choose the most relevant page for given query.

Word clouds also has a hyperlink associated with them to search that particular word on the Yioop search engine. This feature also helps the user to get the synonyms or words closely related to the word they searched for.

6.3. Multi-language support

Centroid summarizer also supports any other languages than English. We are using special regular expressions in the implementation of centroid summarizer to preserve the Unicode characters. For example, instead of using [a-z] in a regular expression, we used p{L} so that it will search for a letter not only from English language but from any language in the text. We have tested the centroid summarizer on Chinese, Marathi, German etc. languages. For testing this, We crawled the Wikipedia's databases for that particular language and queried the database to check the summary and word cloud. Following is the screenshot of the search results page for Chinese language.

Figure 10: Yioop search results page for Chinese language

In the screenshot above, we queried for word "Wikipedia" and got the Wikipedia pages in the search results and word cloud for each returned web page. The word cloud also contains the important words from the web pages like "Wikipedia", "Encyclopedia", "Internet" etc.

7. Experiments

After integrating the centroid summarizer in the Yioop search engine, we performed some experiments to see the effectiveness of the new summarizer on the search engine. We evaluated the summarizer on basis of quality of the generated summary and time required to crawl 10,000 documents.

7.1. Quality of the generated summary

7.1.1. Results

The main purpose of doing this project was to improve the quality of the summary which will also improve the search results. To evaluate the summary generated by the summarizer, we carried out some experiments. For better evaluation, we made a set of ten documents of various lengths and generated a summary for each document using our own judgment so that it can be considered as human generated summary. Then, we generated the summary for these ten documents by the basic summarizer and centroid summarizer. Now, we have calculated the cosine similarity between the human generated summary with the summary generated by two summarizers, basic and centroid, one at a time.

Following are the results from this experiment:

Doc No.	Document Name	Document Length (in characters)	Similarity score with Human generated summary	
			Basic Method	Centroid Method
1	Hockey	98707	0.69	0.76
2	Cricket	266223	0.65	0.67
3	SJSU	282666	0.65	0.70
4	Football	306395	0.69	0.73
5	Volleyball	216200	0.51	0.65
6	Cycling	179671	0.45	0.62
7	Wrestling	135080	0.69	0.78
8	Shooting	80583	0.70	0.70
9	Boxing	255909	0.39	0.42
10	Karate	261855	0.66	0.68

Table 4: Cosine similarity of summary generated by basic and centroid summarizer with a human generated summary

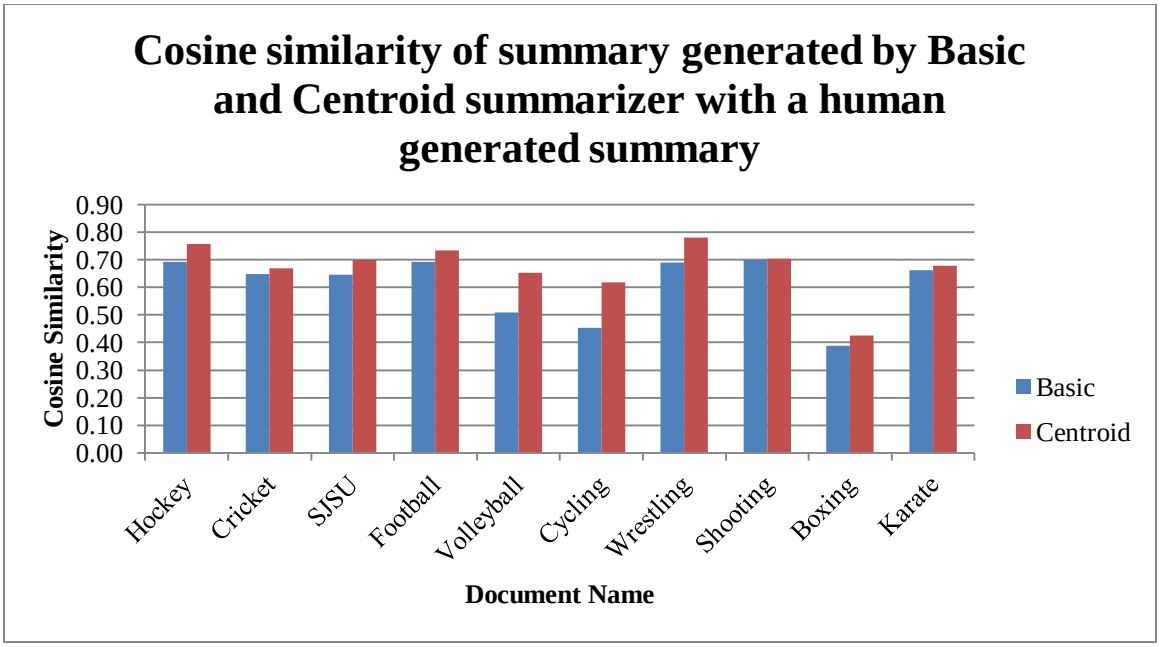

Figure 11: Cosine similarity of summary generated by Basic and centroid summarizer with Human generated summary

7.1.2. Example

Here we present the example summaries generated by basic summarizer, centroid summarizer and human.

Football refers to a number of sports that involve, to varying degrees, kicking a ball with the foot to score a goal. The various codes of football share certain common elements. Players in American football, Canadian football, rugby union and rugby league take-up positions in a limited area of the field at the start of the game. The Ancient Greeks and Romans are known to have played many ball games, some of which involved the use of the feet. Games played in Mesoamerica with rubber balls by indigenous peoples are also well-documented as existing since before this time, but these had more similarities to basketball or volleyball, and since their influence on modern football games is minimal, most do not class them as football. A game known as "football" was played in Scotland as early as the 15th century: it was prohibited by the Football Act 1424 and although the law fell into disuse it was not repealed until 1906. King Henry IV of England also presented one of the earliest documented uses of the English word "football".

Figure 12: Summary generated by human for football Wikipedia web page

Various forms of football can be identified in history, often as popular peasant games. Contemporary codes of football can be traced back to the codification of these games at English public schools in the eighteenth and nineteenth centuries. [2] [3] The influence and power of the British Empire allowed these rules of football to spread to areas of British influence outside of the directly controlled Empire, [4] though by the end of the nineteenth century, distinct regional codes were already developing: Gaelic Football, for example, deliberately incorporated the rules of local traditional football games in order to maintain their heritage. [5] In 1888, The Football League was founded in England, becoming the first of many professional football competitions. During the twentieth century, several of the various kinds of football grew to become among the most popular team sports in the world. [6] .. The various codes of football share certain common elements.

Figure 13: Summary generated by basic summarizer for football Wikipedia web page

Football.
Football refers to a number of sports that involve, to varying degrees, kicking a ball with the foot to score a goal. The most popular of these sports worldwide is association football, more commonly known as just "football" or "soccer". Unqualified, the word football applies to whichever form of football is the most popular in the regional context in which the word appears, including association football, as well as American football, Australian rules football, Canadian football, Gaelic football, rugby league, rugby union, and other related games.
Association football, Australian rules football and Gaelic football tend to use kicking to move the ball around the pitch, with handling more limited. In most codes, there are rules restricting the movement of players offside, and players scoring a goal must put the ball either under or over a crossbar between the goalposts. It is widely assumed that the word "football" or "football "references the action of the foot kicking a ball.

Figure 14: Summary generated by centroid summarizer for football Wikipedia web page

7.2. Effect on crawl time

To evaluate the effect of centroid summarizer on time required to crawl the web pages, we crawled 10,000 pages by basic and centroid summarizer. We downloaded the Wikipedia database [12] to make sure we are crawled the same set of pages.

Crawling 10,000 pages with basic summarizer took 28 minutes while crawling the same set of pages with centroid summarizer took 39 minutes.

8. Conclusion and future work

We researched the text summarization topic to find out which methods are being used for text summarization and studied three methods in depth so that we can implement them and choose one which is best suited for the Yioop search engine. We created a sample document set by which we can compare these three methods and chose the one with high performance and which is best suited for Yioop search engine.

According to the performance analysis done, we have found that intersection method is the fastest method among the three and the centroid method generates the best summary among the three. We calculated the quality of summary by comparing it with a human generated summary. Also, the centroid method generates a word cloud which helps the user to understand the main topic of the document by just looking at the word cloud. The TF-ISF method also generated a good summary. However, it is not practical in terms of speed. After doing this performance analysis, we have decided to implement centroid method for Yioop search engine.

We implemented the centroid summarizer and integrated it into Yioop. After integrating, we performed several experiments to test the performance and to see improvements in results and quality of summary.

Currently, centroid summarizer removes stop words from English web pages only. In future, we can implement the stop words remover for other languages so that the word cloud will contain only informative words.

Bibliography

[1] Bhattacharyya, A. (1946). On a measure of divergence between two multinomial populations. *Sankhyā: The Indian Journal of Statistics*, 401-406.

[2] Büttcher, S., Clarke, C. L., & Cormack, G. V. (2010). *Information retrieval: Implementing and evaluating search engines*. Mit Press.

[3] Dinu, L.P.; Ionescu, R.-T., "A Rank-Based Approach of Cosine Similarity with Applications in Automatic Classification," *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2012 14th International Symposium on* , vol., no., pp.260,264, 26-29 Sept. 2012

[4] Halvey, M. J., & Keane, M. T. (2007, May). An assessment of tag presentation techniques. In *Proceedings of the 16th international conference on World Wide Web* (pp. 1313-1314). ACM.

[5] Jones, K. (2007) Automatic summarising : a review and discussion of the state of the art. *University of Cambridge Computer Laboratory*.

[6] Krygier, J., & Wood, D. (2013). *Making maps: a visual guide to map design for GIS*. Guilford Press.

[7] Mani, I. (2001). Summarization evaluation: An overview.

[8] Mihalcea, R., and Tarau, P. (2004) TextRank: Bringing order into texts. InProceedings of EMNLP.

[9] Radev, D. R., Jing, H., Styś, M., & Tam, D.(2007) Centroid-based summarization of multiple documents. *Information Processing & Management, 40*(6), 919-938.

[10] Rasim M. Alguliev, Ramiz M. Aliguliyev, Nijat R. Isazade. (2007) Formulation of document summarization as a 0–1 nonlinear programming problem. *Computers & Industrial Engineering, Volume 64, Issue 1 ISSN 0360-8352*.

[11] The Tokenizer. Retrieved on Aug 30, 2013, from website: http://thetokenizer.com/2013/04/28/build-your-own-summary-tool/

[12] Wikipedia:Database download Retrieved on Apr 10, 2013, from web site: http://en.wikipedia.org/wiki/Wikipedia:Database_download

[13] Ye, J. (2011). Cosine similarity measures for intuitionistic fuzzy sets and their applications. *Mathematical and Computer Modelling*, *53*(1), 91-97.

[14] Zhao, H., Proctor, I., Yang, M., Qi, X., Williams, M., Gao, Q., ... & Tu, S. (2012, October). The HipHop compiler for PHP. In *ACM SIGPLAN Notices* (Vol. 47, No. 10, pp. 575-586). ACM.

Appendix

A. Additional experiment with HipHop Compiler for PHP

A.1. PHP Background

PHP is a scripting language developed in 1995, mainly used for dynamic web pages. It is an object oriented language and today its use is not limited to web development. Some key features of PHP includes: dynamic typing, dynamic name binding, dynamic name resolution, dynamic symbol inspection, reflection, dynamic code evaluation [14].

A.2. Standard PHP Implementation

The standard implementation of PHP is an interpreter to support all the dynamic features of PHP. This interpreter is called Zend which is a bytecode interpreter which uses a lower level program implementation called the Zend bytecode [14].

For a new file invoked, Zend parses that file and translates it into bytecode. It loads various program components during execution. This feature is called dynamic loading. It is expensive for classes which requires composing class methods, properties and constants. When interpreter needs access to a symbol, it finds the symbol name in the lookup table. This process has a runtime cost called as dynamic lookups. Dynamic loading, dynamic lookups and dynamic typing are the major overhead in Zend interpreter [14].

A.3. HipHop Compiler

HipHop is a static compiler developed by Facebook which is different from the PHP's standard implementation. The main differences includes: First, HipHop compiler needs all source code to be known in advance which boosts the performance. Second, HipHop doesn't support all features of the PHP like dynamic code evaluation. HipHop also does not support the automatic promotion from integer to floating point numbers in case of overflow. Third, HipHop analyzes, compiles and loads all the symbols in advance. Finally, a small amount of change in a code can result in rebuilding the system which reduces programmer productivity. Facebook addresses this problem by combining the use of HipHop for production code with the use of PHP's standard interpreter for code development [14].

A.4. Experiment

After studying the high performance of HipHop compiler, we decided to run all three summarizer methods on HipHop compiler and compare the time required to generate the summary with time required on the Zend interpreter.

We have used a set of ten documents for this experiment. Average size of a document in that set was 50KB. We ran each summarizer on ten documents at a time to compare the HipHop compiler and Zend interpreter.

Following is the table showing the results of the experiment:

	Zend Interpreter	HipHop Facebook Compiler	Improvement in speed
Intersection	7.57 ±0.8	5.22 ±0.7	2.36 ±0.10
Centroid	21.25 ±1.0	12.12 ±0.5	9.12 ±0.50
TFISF	59.24 ±1.9	54.65 ±1.0	4.60 ±0.9

Table 5: Comparison between using interpreter and compiler for running summarizers

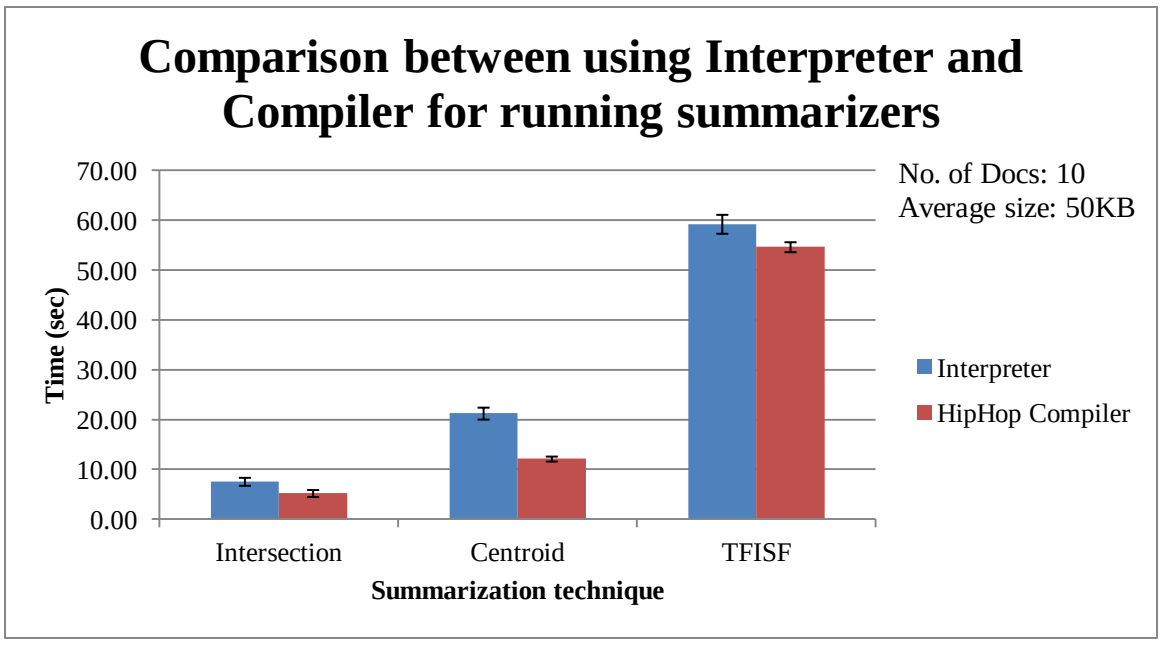

Figure 15: Comparison between using Interpreter and Compiler for running summarizers