Computational Linguistics

Automatic Summarizer

Project Report

Sulabh Bansal
Himanshu Jindal
Kavita Malani

Table of Contents

1. Objective [4]

To take a document as input and give another document with lesser number of words as output, while retaining the meaning of the original document. The main challenge here is that there is no one single method to evaluate the quality of the summaries. Moreover, there are no gold standard summaries, which can be used for comparison. Hence, our objective is to find summaries by using different features and evaluate them using different techniques and analyze which technique works for which feature and why.

2. Benchmark Summaries

Because of the aforementioned challenges, we have decided to form 2 categories of benchmark Data

2.1 Automated summaries [5]

Intellexer Document Summarizer is a desktop application that analyzes the document, extracts its main ideas and puts them into a short summary or creates annotation. This software currently sells for 400$ per copy. [1] However, its trial version can be used for free up to 1 month. We have collected the summaries of 500 different articles. The articles have been selected to be diverse, so that they cover many different topics. Hence, we have used these 500 summaries as the gold standard of automated summaries.

2.2 Manual Summaries

We have used the data collected from the DUC [1] (Document Understanding conferences) and have collected 500 summaries from this data. These form the gold standard of manual summaries

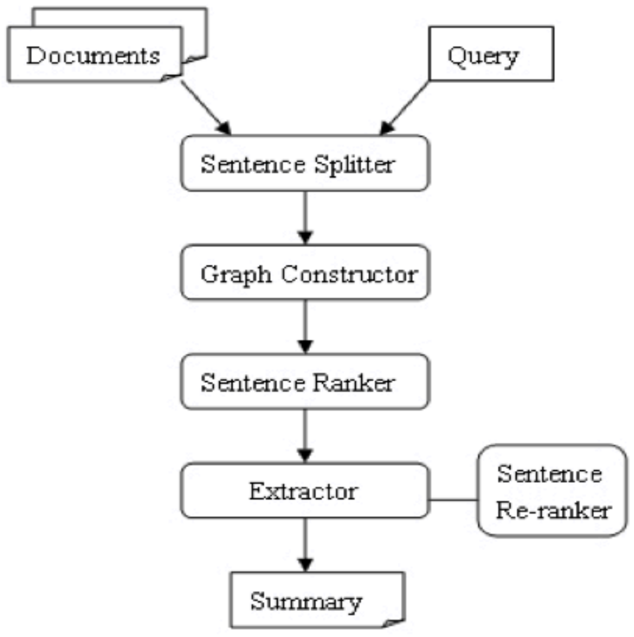
OUR GOLD STANDARD

Summaries generated by a renowned software	500
User Generated summaries	*500*

For evaluation, we wrote our own program to do an Ngram analysis to calculate precision, recall and FMeasure. Details of which are explained later.

3. Methodology

To implement this task, we have split the task into a number of sub tasks. For each sub task, we have employed different features and have tried to see which one performs the best, and analyze why.



The subtasks performed by our summarizer.

3.1 Document Preprocessing

- The original document will contain many words, which will not be important in the sentence. It would be obvious to remove the stop words such as “A”, “the” etc. from the sentence.
- There are many more sophisticated methods to process the documents, such as retrieving only nouns or nouns and adverbs etc. We will use these features and evaluate which gives the best summaries, which we have explained in section 6.11.

3.2 Graph Construction

- We then construct a graph from the words that we have filtered from each sentence.

- We assign each sentence a node and calculate the edge weight between each sentence by various similarity functions.
- In later sections, we have proposed another method to construct graph from a document (Bipartite Hits), which will be explained later.
- There are two methods to calculate similarity between sentences.
    - One is to calculate sentence similarity with the words in sentences.
    - The other is to calculate sentence similarity with stemmed words in sentences.
- We have chosen the original words in the sentences, because the stemmed words would miss the tense and voice information of the sentence.
- Moreover, there are many different Similarity functions, which can be used, with each having its own benefits and drawbacks.

3.3 Ranking Algorithms

Once we have obtained a graph with edges, we can apply various techniques, such as HITS or TextRank, to obtain weight for each node. Using these ranking algorithms, we can obtain a weight or importance for each node.

3.4 Summarization

Once we have each sentence and a measure of its importance, we can sort the nodes in the order of their weights and display the sentences with most similarity.
This will give the summary.

4. Features used

To evaluate how to improve the summaries, will be using the following features,

- Nouns and verbs
- Nouns, adjectives and Verbs
- Nouns and adjectives

We will be using the following similarity function [6] to generate different summaries:

4.1 Jaccard Similarity

a. It is a statistic used for comparing the similarity and diversity of sample sets.

b. It uses overlap of words between sentences to calculate similarity

$$Jaccard(s_a, s_b) = \frac{|w(s_a) \cap w(s_b)|}{|w(s_a) \cup w(s_b)|}.$$

4.2 Dice Similarity

a) Dice coefficient, is a similarity measure over sets:

b) It uses overlap of words between sentences to calculate similarity

$$Dice(s_a, s_b) = \frac{2|w(s_a) \cap w(s_b)|}{|w(s_a)| + |w(s_b)|}.$$

4.3 Rada Mihalcea's Similarity Measure

a. This uses overlap of words between the two sentences. However, it uses a different normalization factor

$$Similarity(S_i, S_j) = \frac{|\{w_k | w_k \in S_i \& w_k \in S_j\}|}{log(|S_i|) + log(|S_j|)}$$

4.4 Cosine Similarity

Documents are represented by a bag of words, and vectors represent the meanings of the documents. The document similarity can be calculated by using cosine of the vectors, which actually depict the **TF IDF** value of each word.

TF = Count of Word in a sentence / Number of words in a sentence
IDF = log (Total number of sentences / Number of sentences in which the word appear)

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

5. Ranking Algorithms Used

We will be using the 3 features along with the 3 syntactic filtering mentioned above to obtain different graphs, on which we various ranking algorithms can be applied to obtain the sentence weights (importance).

5.1 TextRank [3]

TextRank gives a weight (importance) of each node in the graph by running the above formula till convergence

$$WS(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

Moreover, it goes beyond connectivity in text, that is, it takes into account both the connectivity and the strength of the connections (similarity between sentences) into account, when awarding weights. For the key phrase extraction based on keyword co-occurrence, we have referred to this source [8].

5.2. Hits Algorithm

Hyperlink-Induced Topic Search (*HITS*) (also known as **hubs** and **authorities**) is a link analysis algorithm that rates Web pages, developed by Jon Kleinberg.
Hubs - good hub represents a webpage that links to many other web pages
Authority - good authority represents a page that is linked by many different pages

$$HITS_A^W(V_i) = \sum_{V_j \in In(V_i)} w_{ji} HITS_H^W(V_j)$$

$$HITS_H^W(V_i) = \sum_{V_j \in Out(V_i)} w_{ij} HITS_A^W(V_j)$$

For each vertex, the algorithm iteratively produces 2 scores for each node (sentence)
Hub score represents the importance of the sentence as a hub
Authority score represents the importance of the sentence as an authority

Issue of Symmetry
However, HITS algorithm requires an unsymmetrical weighted graph for convergence.
Hence, to introduce asymmetry, we have used the following formula for calculating similarity between sentences

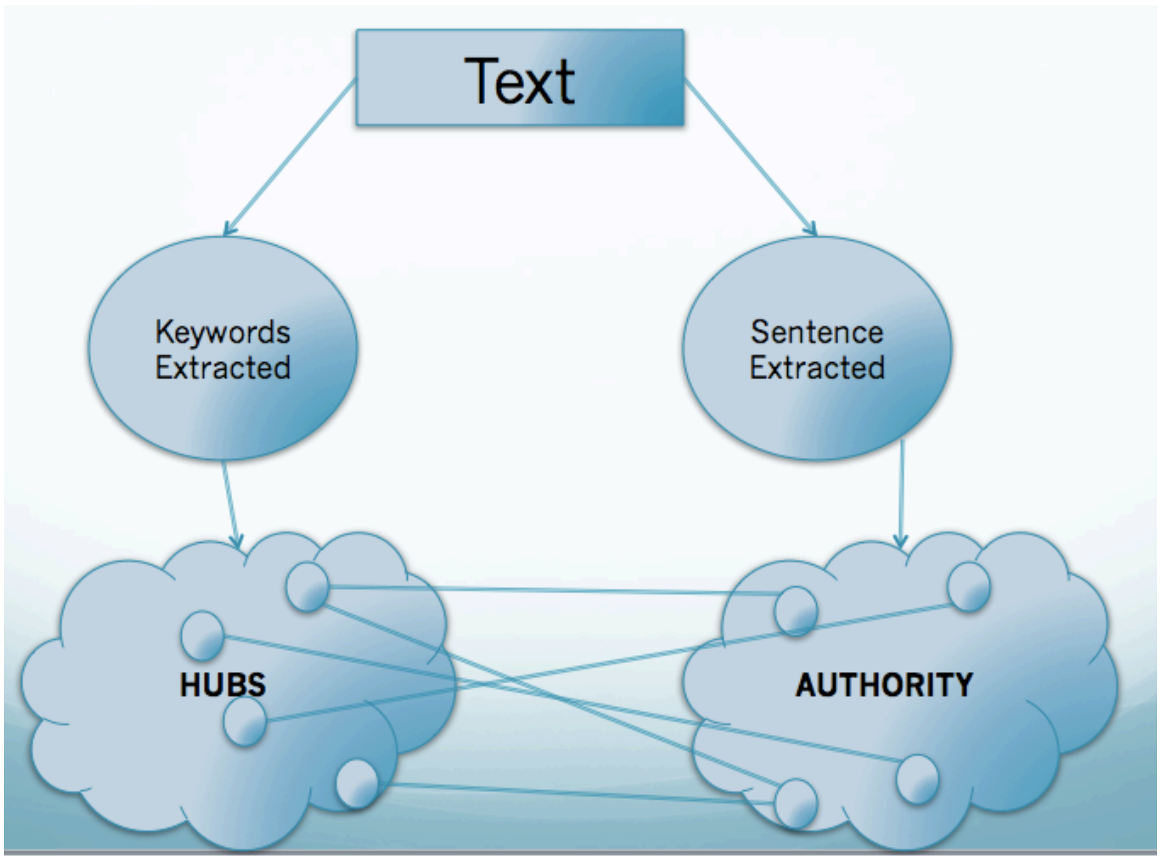
$$Similarity\ (S_i, S_j) = Intersection\ (W\ (S_i), W\ (S_j)))/\ |(S_i)|$$

After computing the similarities and obtaining an asymmetric graph, we can apply the aforementioned algorithm to get Hub scores and Authority scores for a document, each representing a different summary for the given document.

5.3 Bipartite HITS

The standard HITS algorithm described in the earlier section was not intuitive in its approach to summarizing.
Hence, we have devised a new algorithm to summarize the sentences.
The basis of our algorithm is that the *sentences with highest importance will be the ones with the highest number of keywords*.
Hence, we have converted this notion to Hubs and authorities to apply the HITS algorithm on the obtained values.

Construction of the Bipartite Graph
Hubs – We take the keywords from the documents as Hubs. Different ways are used to identify the keywords.
Authorities – All the sentences in the document are taken as the authorities.
There is an edge between a hub and an authority, if the keyword is present in the particular sentence.
The following figure demonstrates how to construct the graph from a document.



Since, Bipartite HITS focuses on getting the keywords from the document, we have used the following ways to extract the keywords from the document

1. *TextRank for Keywords*
   This picks the top keywords by applying TextRank algorithm on the node-weighted graph formed from the document [3].

$$S(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

2. *Top Occurring words*
   We can use the top occurring words in the paragraph as the keywords
This can be done in 2 ways
   a. With Stop Words
   b. Without Stop Words

3. *Random Keywords*

For the purpose of baseline, we have also used random occurring keywords, both with and without stop words to find the summaries.

4. *Weight Classes for Sentences*

In addition to different keywords, we have also assigned weights to different sentences on the basis of sentence length.
The reason behind this is that longer sentences tend to be more informative and intuitive.

- **2 weight classes**
  We assign the longer sentence (length > 25 words) as initial weight = 2, and the rest as 1
- **3 weight classes**
  We assign 3 classes
  - Longer sentence (length > 25 words) as initial weight = 3
  - Longer sentence (length >10 and < 25 words) as initial weight = 2
  - Longer sentence (length < 10 words) as initial weight = 1

6. Experiments Performed

We have done the following experiments to summarize the documents

6.1 Experiment 1 - TextRank

Baseline

1.Selecting all words
2.Cosine Similarity
3.TextRank

The results for this baseline were the following

	Machine Generated Summaries			Human Summaries		
N	Precision	Recall	FMeasure	Precision	Recall	FMeasure
1	70.58	44.87	50.67	24.98	50.47	30.87
2	59.21	34.73	40.49	9.79	18.21	11.65
3	56.32	32.26	37.96	6.14	10.67	7.10

Experiment 6.1.1

For this experiment, we compared the results of syntactic filtering and compared which one gives the best results. We have used the average of various similarity measures to compute the value for each syntactic filter.

	Nouns + Adjective			Nouns + Adjective + Verbs			Nouns + Verbs		
N	1	2	3	1	2	3	1	2	3
Precision	74.40%	57.50%	51.85%	74.10%	56.60%	51.00%	72.00%	56.00%	52.70%
Recall	54.40%	36.80%	31.90%	54.30%	35.80%	30.90%	48.50%	37.40%	34.10%
FMeasure	57.20%	41.20%	36.50%	57.00%	40.30%	35.70%	52.70%	41.30%	38.25%

Inferences

- The Nouns and adjectives outperform the other two features for N = 1.
- However, for N = 3, Nouns and verbs perform much better.

Experiment 6.1.2

Then, we computed different summaries using various similarity measures mentioned in part 4 of this report. We have used the average of the results obtained from different syntactic filters to compute the value for each similarity measure

Machine generated summaries

	Rada Mihalcea's			Jaccard			Dice		
N	Precision	Recall	FMeasure	Precision	Recall	FMeasure	Precision	Recall	FMeasure
1	74.30%	54.60%	57.30%	74.10%	51.70%	55.60%	72.10%	50.80%	54.10%
2	57.10%	36.30%	40.80%	56.70%	34.50%	39.50%	57.00%	39.10%	42.40%
3	51.40%	31.40%	36.10%	51.00%	29.80%	35.00%	53.10%	35.70%	39.30%

Human generated summaries

	Rada Mihalcea's			Jaccard			Dice		
N	Precision	Recall	FMeasure	Precision	Recall	FMeasure	Precision	Recall	FMeasure
1	23.43%	59.06%	30.79%	24.47%	56.38%	31.26%	24.32%	56.46%	31.12%
2	11.00%	26.08%	13.99%	11.00%	23.93%	13.72%	11.00%	23.94%	13.64%
3	7.06%	15.92%	8.85%	7.17%	14.43%	8.59%	7.11%	14.44%	8.54%

Cosine Similarity

	Machine Generated Summaries			Human Summaries		
N	Precision	Recall	FMeasure	Precision	Recall	FMeasure
1	71.53	56.35	57.29	23.31	58.16	30.63
2	62.98	47.09	48.85	10.50	24.98	13.49
3	60.45	44.51	46.43	6.69	15.11	8.41

Inference

- The FMeasure is significantly lower than the FMeasure obtained from the comparison with Manual Summaries.
- However a decrease in FMeasure does not mean that our summaries are not good. The evaluation method used by us compares the ngrams in each document. However, if the user generates summary using other words and changing the order of sentences, then any computer-generated summary will score low on FMeasure. Hence, we should not compare the FMeasure values obtained with human summaries and computer summaries.

Consolidated Result for TextRank

Experiment	Algorithm	Intellexer FMEASURE			DUC FMEASURE		
		N = 1	N = 2	N = 3	N = 1	N = 2	N = 3
Baseline	All Words + Cosine Similarity + TextRank	50.67	40.49	37.96	30.87	11.65	7.10
Experiment 1	TextRank with Dice Similarity	55.32	46.10	43.57	31.12	13.64	8.54
Experiment 2	TextRank with Jaccard Similarity	54.93	45.65	43.12	31.26	13.72	8.59
Experiment 3	TextRank with Rada Mihalcea's Similarity	56.67	47.83	45.37	30.79	14.00	8.85
Experiment 4	TextRank with Cosine Similarity	57.29	48.85	46.43	30.63	13.49	8.41

6.2 Experiment 2

We then performed the HITS algorithm on the same graph using the asymmetric similarity measure explained in section 5.2

Human generated summaries

	Hub Scores			Authority Scores		
N	**Precision**	**Recall**	**FMeasure**	**Precision**	**Recall**	**FMeasure**
1	24.74	55.57	31.05	23.05	60.08	30.76
2	11.08	23.75	13.54	11.03	27.25	14.38
3	7.07	14.45	8.45	7.26	16.97	9.26

Machine generated summaries

	Hub Scores			Authority Scores		
N	**Precision**	**Recall**	**FMeasure**	**Precision**	**Recall**	**FMeasure**
1	70.96	48.86	52.19	69.80	56.92	57.38
2	60.88	39.26	42.88	61.29	47.38	48.81
3	58.07	36.81	40.41	58.80	44.79	46.41

Inference

- The authority scores seem to work better than the hub scores.
- However, the results of the summaries obtained are close to the summaries obtained previously by TextRank.
- Hence, we modified the algorithm and tried Experiment 3.

6.3 Experiment 3 - *Bipartite HITS*

We tried various different ways to select keywords, as explained in section 5.3

Baseline Experiment

Picking random keywords including Stop Words

	Intellexer			DUC		
N	Precision	Recall	FMeasure	Precision	Recall	FMeasure
1	66.34	65.08	64.05	25.07	47.41	32.39
2	54.76	53.91	52.91	9.24	17.44	11.90
3	51.96	51.17	50.21	5.56	10.42	7.13

Picking random keywords without Stop Words

	Intellexer			DUC		
N	Precision	Recall	FMeasure	Precision	Recall	FMeasure
1	68.00	71.63	68.21	24.59	45.21	31.39
2	57.97	60.61	57.89	8.85	16.44	11.32
3	55.36	57.77	55.22	5.35	9.87	6.81

Experiment 6.3.1

Using TextRank to pick keywords from the document

	Intellexer			DUC		
N	Precision	Recall	FMeasure	Precision	Recall	FMeasure
1	68.67	68.49	66.89	26.49	53.24	34.95
2	58.11	57.74	56.46	11.23	22.22	14.71
3	55.40	54.97	53.78	7.14	13.97	9.31

Experiment 6.3.2
Using maximum occurring words along with stop words as the keywords

	Intellexer			DUC		
N	Precision	Recall	FMeasure	Precision	Recall	FMeasure
1	68.00	71.63	68.21	25.22	54.63	34.13
2	57.97	60.61	57.89	10.68	22.60	14.31
3	55.36	57.77	55.22	6.74	14.04	8.97

Experiment 6.3.3
Using maximum occurring words without stop words as the keywords

	Intellexer			DUC		
N	Precision	Recall	FMeasure	Precision	Recall	FMeasure
1	69.39	68.70	67.42	27.06	53.79	35.57
2	59.04	58.28	57.24	11.76	23.07	15.36
3	56.35	55.55	54.58	7.55	14.69	9.82

Experiment 6.3.4
Using 2 weight classes for sentences with Top occurring words as Keywords with stop words

	Intellexer			DUC		
N	Precision	Recall	FMeasure	Precision	Recall	FMeasure
1	67.96	71.63	68.19	25.24	54.68	34.16
2	57.92	60.58	57.84	10.70	22.65	14.34
3	55.30	57.75	55.17	6.76	14.09	9.00

Experiment 6.3.5
Using 3 weight classes for sentences with Top occurring words as Keywords with stop words

	Intellexer			DUC		
N	Precision	Recall	FMeasure	Precision	Recall	FMeasure
1	68.00	71.65	68.23	25.23	54.63	34.13
2	57.98	60.63	57.90	10.68	22.61	14.32
3	55.37	57.79	55.23	6.74	14.04	8.98

Inferences

- We see that Bipartite HITS seems to give better results than the other conventional methods
- Moreover, both TextRank and maximum occurring words as keywords perform well, with the latter performing slightly better.

SUMMARY OF BIPARTITE HITS

Experiment	**Algorithm**	**Intellexer**			**DUC**		
		FMEASURE			**FMEASURE**		
		N = 1	**N = 2**	**N = 3**	**N = 1**	**N = 2**	**N = 3**
Baseline	Bipartite Hit using Random keywords with stop words	64.05	52.91	50.21	32.39	11.90	7.13
	Bipartite Hit using Random keywords without stop words	62.37	51.07	48.42	31.39	11.32	6.81
Experiment 1	Bipartite Hit using Top Occurring words without stop words	67.42	57.24	54.58	35.57	15.36	9.82
Experiment 2	Bipartite Hit using Keywords from TextRank	66.89	56.46	53.78	34.95	14.71	9.31
Experiment 3	Bipartite Hit using Top Occurring words with stop words	68.21	57.89	55.22	34.13	14.31	8.97
Experiment 4	Bipartite Hit using Top Occurring words with stop words and 2 weight classes	68.19	57.84	55.17	34.16	14.34	9.00
Experiment 5	Bipartite Hit using Top Occurring words with stop words and 3 weight classes	68.23	57.90	55.23	34.13	14.32	8.98

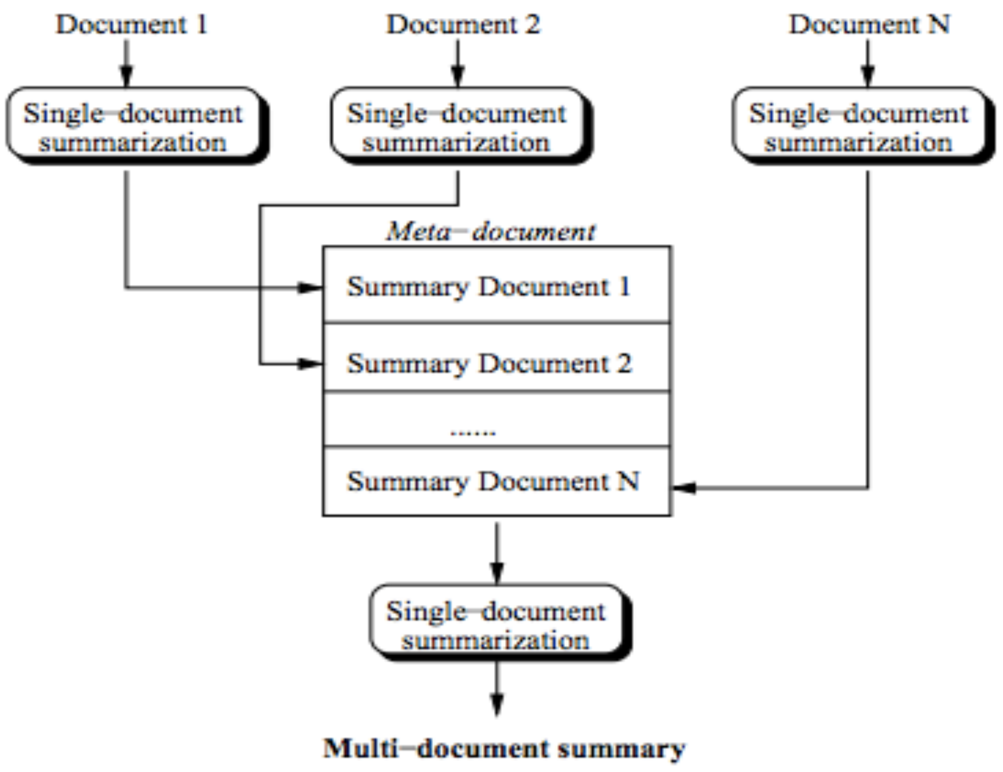
FINAL SUMMARY OF OUR RESULTS

Experiment	**Algorithm**	**Intellexer**			**DUC**		
		FMEASURE			**FMEASURE**		
		N = 1	**N = 2**	**N = 3**	**N = 1**	**N = 2**	**N = 3**
Baseline	Bipartite Hit using Random keywords with stop words	64.05	52.91	50.21	32.39	11.9	7.13
	Bipartite Hit using Random keywords without stop words	62.37	51.07	48.42	31.39	11.32	6.81
Experiment 1	Bipartite Hit using Top Occurring words without stop words	67.42	57.24	54.58	**35.57**	**15.36**	**9.82**
Experiment 2	Bipartite Hit using Keywords from TextRank	66.89	56.46	53.78	34.95	14.71	9.31
Experiment 3	Bipartite Hit using Top Occurring words with stop words	68.21	57.89	55.22	34.13	14.31	8.97
Experiment 4	Bipartite Hit using Top Occurring words with stop words and 2 weight classes	68.19	57.84	55.17	34.16	14.34	9
Experiment 5	Bipartite Hit using Top Occurring words with stop words and 3 weight classes	**68.23**	**57.9**	**55.23**	34.13	14.32	8.98
Experiment 1	HITS: Hub Scores with Unsymmetrical Similarity as edge weight	52.19	42.88	40.41	31.05	13.54	8.45
Experiment 2	HITS: Authority Scores with Unsymmetrical Similarity as edge weight	57.38	48.81	46.41	30.76	14.38	9.26
Baseline	All Words + Cosine Similarity + TextRank	50.67	40.49	37.96	30.87	11.65	7.1
Experiment 1	TextRank with Dice Similarity	55.32	46.1	43.57	31.12	13.64	8.54
Experiment 2	TextRank with Jaccard Similarity	54.93	45.65	43.12	**31.26**	**13.72**	**8.59**
Experiment 3	TextRank with Rada Mihalcea's Similarity	56.67	47.83	45.37	30.79	14	8.85
Experiment 4	TextRank with Cosine Similarity	**57.29**	**48.85**	**46.43**	30.63	13.49	8.41

Inference

- Hence, the Bipartite Hit using Top Occurring words without stop words is the one which gives the best results for human generated summaries
- Also, the Bipartite HITS performs much better than TextRank and conventional HITS algorithm

7. Multi-Document Summarization [7]



N	Precision	Recall	FMeasure
1	38.13	38.33	36.03
2	8.41	8.39	7.89
3	2.57	2.47	2.36

Data

The data for multi document summarization was taken from 60 documents containing 10 documents each and were human generated.

Inference

- We observed that Cosine similarity function gave the best results for single document summarization
- Hence, we have used the same with Nouns and Adjectives as syntactic filters

Future Work

To implement a research paper for summarization, we will do the following

- Get the data used for other papers (Rada Mihalcea's TextRank) which we have used as the baseline – DUC 2002
- Evaluate all the summaries using the Rouge software and evaluate the results obtained by our TextRank and HITS with the results obtained in other papers
- Literature survey to find out another source of human generated summaries apart from DUC
- Implement 2 new Baselines
    - Take the top 1/3rd sentences generated by the algorithms and do ngram analysis
    - Take the last 1/3rd sentences and do the analysis

References

[1] - http://www-nlpir.nist.gov/projects/duc/
[2] - https://github.com/turian/textrank#readme
[3] - http://www.cse.unt.edu/~rada/papers/mihalcea.emnlp04.pdf
[4] - http://en.wikipedia.org/wiki/Document_summarization
[5] - http://www.cs.cmu.edu/~nasmith/LS2/das-martins.07.pdf
[6] - http://www.aicit.org/jcit/ppl/03-JCIT1-785287JE.pdf
[7] - http://acl.ldc.upenn.edu/W/W00/W00-0405.pdf
[8] - https://github.com/turian/textrank