

CS585: NLP Project- Final Report

Title:

Text Summarization of Movie Reviews Using Graph Based Algorithm

Authors:

- Karthik Shivaram
- Nikhil Birur
- Zinou Li

Abstract:

Automatic Text Summarization involves condensing a document or a document set to produce a human comprehensible summary. The main idea of summarization is to find a representative subset of the data, which contains the information of the entire set. Summarization technologies are used in many sectors in industry today. An example of the use of summarization technology is "Google", a widely popular search Engine. In this project, we aim at presenting an unsupervised method for automatic sentence extraction using a graph- based ranking algorithm like **Page Rank**. Our aim is to generate a set of sentences from a huge document that sum up the content of the document. We start by scraping "critics reviews" of a movie from a popular movie database called RottenTomatoes.com along with the human picked summary of the movie. Once we have the required data, we make use of several similarity measures along with the page rank algorithm to rank every sentence in the given critic review. We then use the highest ranked sentence and evaluate it against the website generated 1-line summary of the movie. We then compare the system generated summary against our baseline which is the first and the last sentences of the summary. Once we have the two sentences, the system generated and the website generated summary lines, we make use of ROUGE (Recall-Oriented Understudy for Gisting Evaluation) for evaluating the closeness between the two sentences. We finally determine the quality of the generated summary against the baselines.

Assumptions Made:

The system generated, highest ranked, first sentence is the most descriptive sentence of the entire document.

Introduction:

Information/Data is one of the most important aspect of one's life. People rely on a wide variety of sources, ranging from news stories to media posts to search results, to stay informed. Being able to develop Machine Learning models that can automatically deliver accurate summaries of longer text can be useful for digesting such copious amounts of information in a compressed form. There's tremendous information overload that comes into effect as search engines retrieve thousands of search results in the form of links to the document. Though, the time taken by search results to display a list of these documents is very less, without summarization of these documents, the user should literally go through every document before he/she finds the relevant document. With tremendous amount of data available on the worldwide web, usage of text summarization not only saves the search time but also helps a user have a cut short understanding of information available.

There are two widely used approaches to summarize a text document:

- Extractive Summarization
- Abstractive Summarization

The two methods differ in the summary produced. Abstractive Summarization generates a summary that might contain words that are not present in the to-be summarized document. It is a summary that is closer to a summary generated by humans. On the other hand, Extractive Summarization is a summary produced by selecting a subset of existing words, phrases, or sentences in the original text to form the summary. In this project, we concentrate on the Extractive Summarization task using graph based algorithm. There are also various challenges for an Automatic Text Summarization System. One of the most important challenges is that summarization quality depends on the eye of the beholder – one person's interest in a body of text is different than another's; one person's view of the quality of a summary differs from

another's. Another noteworthy problem of text summarization is to search the document for sentences that use language that normally occurs in statements of broad description, judgement, or reflection. The goal of this project is to build a movie review summarization system that can detect the most important sentence(s) of a review to summarize the entire review. We attempt to do this in a machine learning context. Though Text summarization is complex and interesting, it proved to be much more complex in the case of movie review summarization. A technical document usually has a structure in which meaningful sentences that are related to one another are grouped in a single paragraph. There are certain keywords in such documents which make the idea of summarization a little easier. For instance, the last or the first paragraph of the document might summarize the entire document or the presence of keyword like "Therefore" in a paragraph makes it easier to summarize. In the case of movie reviews, very few reviews are carefully documented and have a structure. There's no fixed structure that each of these reviews follow and it's up to the reviewer to come up with a structure of his/her own. A review may begin with a description of the reviewer's expectations, only to later go on to describe how the movie did not live up to them. These factors combine to make movie review summarization a challenging task.

Background / References:

Here is the list of Summaries of the Scientific Papers we have referred:

Extractive Summarization Using Supervised and Semi-Supervised Learning:

The author proposed a learning-based approach to combine various sentence features categorized as surface, content, relevance and event. Each sentence feature has its unique contribution. First, deciding which sentence is most important in the text. Second, employing supervised learning classifier to examine features. Finally, re-ranking candidate sentences and extracting the top sentences to summarize the whole text. Also, the author investigated co-training by combining labeled and unlabeled data to perform semi-supervised learning.

Project Similarity/Dissimilarity:

This was the first paper we read to get an idea on how text summarization is done. Only few ideas on the approach are taken by us here no similarity exists between the paper and our project.

Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization:

Graph-based ranking algorithm is a way of deciding on the importance of a vertex within a graph, by considering global information recursively computed from the entire graph, rather than relying only on local vertex-specific information. In this paper, the author investigated several graph-based ranking algorithms and evaluated their application to unsupervised sentence extraction in a context.

Project Similarity/Dissimilarity:

This gave us the idea of using graphs to represent sentences as nodes in the graph and how we could rank them. Here we take the idea of graph based Ranking algorithms from this paper.

The Evaluation of Sentence Similarity Measures:

In this paper, the author tried fourteen similarity measures which could be concluded in three different classes: word overlap, TF-IDF, and linguistic measures, used in the evaluation. Two sentences are similar if they are a paraphrase of each other, which means they talk about the same event or idea judging from the common principle actors and actions, or if one sentence is a subset of the other.

Project Similarity/Dissimilarity:

Here we learn about different sentence similarity measures so we can utilize them in our system. From this paper, we take the different methods of calculating similarities between sentences.

TextRank: Bringing Order into Texts:

The author introduced the TextRank graph-based ranking model for graphs extracted from natural language texts. They investigated and evaluated the application of TextRank to two language processing tasks consisting of unsupervised keyword and sentence extraction, and showed that the results are competitive with other algorithms.

Project Similarity/Dissimilarity:

This is the main paper we take our text summarizer's algorithm from. The main algorithm discussed in this paper for sentence ranking is used in our system.

LexRank: Graph-based Lexical Centrality as Salience 5 in Text Summarization:

Instead of TextRank, the author considered an innovative approach, LexRank, for computing sentence importance based on the concept of eigenvector centrality in a graph representation of sentences. This paper assesses the centrality of each sentence in a cluster and extract the most important ones to include in the summary.

Project Similarity/Dissimilarity:

This paper gives us another insight on how to represent edges between our nodes (i.e. the relationship between 2 given sentences). We also implement the same algorithm given here. The paper given above this and this paper are quite similar on how to rank the sentences only the scoring metric for the edges between nodes is different.

Approach:


1. Data:

Rotten Tomatoes:

Rotten Tomatoes is a very popular, American review aggregator website for film and television. Rotten Tomatoes maintains a page with a variety of information and links related to any movie that has been released on a worldwide platform. Rotten Tomatoes gives a rating (percentage) to each movie based on user reviews as well as critic reviews. However, this rating does not summarize the reviews of the critics or users and is only an indication of how good or bad a movie is. In this project, we concentrated mainly on critic reviews as these reviews are more descriptive than a regular user review and we hoped to find more information in such reviews. The critic review section on Rotten Tomatoes for one of the movie looks like the image below Rotten Tomatoes divides the reviews into three categories. These are “Top Critics”, “Fresh” and “Rotten”. There’s also a 4th category, “All Critics”, which gives us a list of all the reviews. As seen above, Rotten Tomatoes gives us a 1-line summary of the review and a hyper link, “Full Review”. Clicking on this hyperlink will take us to a webpage which displays the complete review of a critic. The central focus of our project is to extract these 1-line summaries and compare them with the system generated summary of the whole review.


CRITIC REVIEWS FOR THE BOSS BABY


All Critics (137) | Top Critics (29) | Fresh (73) | Rotten (64)




Its snappy, pop-culture-referencing script feels workshopped to death.

April 3, 2017 | Rating: 2/5 | [Full Review...](#)





Tom Huddleston
Time Out
 Top Critic




I could have done without the kewpie-doll faces and oversized eyes, but for the most part and where it counts, The Boss Baby gets its kids just right.

March 31, 2017 | Rating: 3/5 | [Full Review...](#)





Matthew Lickona
San Diego Reader
 Top Critic




It's the rare cartoon that actually feels like a cartoon, propelled by its goofiness and sheer energy and rarely bogged down by boring, polemical lesson-learning.

March 31, 2017 | Rating: 3/4 | [Full Review...](#)





John Semley
Globe and Mail
 Top Critic



It's not easy to make an entire movie around a two-word premise, and "Boss Baby" shows why.

March 31, 2017 | Rating: C | [Full Review...](#)



Adam Graham
Detroit News
 Top Critic

One of the assumptions in our project is that the 1-line summary describes the entire movie as closely as possible.

We start by choosing "n" different movies and run a web scraper built by us, called RottenTomatoesScraper.py that does the following:

- Visits the homepage of each movie.
- Goes to the critic's review section.
- Extracts the one line summary.
- Visits the full review and extracts the complete review.
- Writes the data collected to a pickle file. Also, each critic review is written to a separate file under the folder named after the movie.

At the end of the script we had a pickle file with information structured into movie id, movie name, one line summary and the complete movie review. We also had the reviews saved as separate files in the folder named after the movie.

2. Data Processing:

We removed a lot of unwanted data from the collected data and restricted our attention to only those summaries which had greater than 300 characters. Each review was broken into individual sentences using a very popular natural language processing library called "PunktSentenceTokenizer" from "nltk". These sentences were further tokenized into individual words and we made sure to remove the stop words. We also made sure to remove numbers, punctuation and any non-Unicode characters

3. Similarity Measurement:

Once we had the individual tokens, we calculated the similarities between two sentences using the following measures:

Tf-idf cosine similarity:

Term Frequency, also known as TF measures the number of times a term occurs in a document. In real world, a large document may contain more words so that the frequency of each terms will be much higher than the smaller ones. Hence a normalization based on its size should be done. One easy approach is to divide the term frequency by the total number of terms. Inverse Document Frequency (IDF) is the method to weigh down the effects of too frequently occurring terms and weigh up the effects of less frequently occurring terms. Logarithm can be used to solve this problem. TF-IDF is a multiplication of both TF value and IDF value. The set of documents in a collection is viewed as a set of vectors in a vector space. We used the formula below to compute the similarity of both documents.

The tf-idf scoring is done as follows:

1) Term Frequency:

$$TF(t) = (\text{Number of times term } t \text{ appears in a document}) / (\text{Total number of terms in the document})$$

2) Inverse Document Frequency:

$$IDF(t) = \text{LOG}_E (\text{Total number of documents} / \text{Number of documents with term } t \text{ in it}).$$

3) Overall Cosine Similarity is found as follows:

$$\text{sim}(d_1, d_2) = \frac{\vec{v}(d_1) \cdot \vec{v}(d_2)}{|\vec{v}(d_1)| |\vec{v}(d_2)|}$$

Ngram cosine similarity:

This is a method to first create an N-gram word model's frequency vectors, and compute their cosine similarity. In our case, we defined a fixed and common order among all possible Ngrams. For each sentence, we obtain a vector of dimensionality N. If we can't find such Ngram, just set the corresponding value in the vector to zero.

Jaccard similarity:

Jaccard similarity coefficient is a strategy used for comparing the similarity and diversity of sample sets. It is defined as the size of the intersection divided by the size of the union of the sample sets.

$$J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

Okapi bm25 similarity:

BM25 is a bag-of-words retrieval function that ranks a set of documents based on the query terms appearing in each document, regardless of the inter-relationship between the query terms within a document such as relative proximity.

The formula is defined as: Given a query Q, containing keywords q1, q2, ..., qn, the BM25 score of Document D is:

$$\text{score}(D, Q) = \sum_{i=1}^n \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)},$$

Where

- $f(q_i, D)$ is q_i 's term frequency in the document
- $|D|$ is the length of the document
- avgdl is the average length of document in the text collection.

Here D and Q are both sentences.

Word2Vec for cosine similarity:

Word2vec is a method to represent words as a mathematical vector. First, we need to train the model on a large corpus, given the vector size you want, window size, and some other features you want to add, it will generate a large keyed-vectors which contains each word with their distinguish vector representation. With that information, we can easily calculate the cosine similarity of different sentences, by averaging out the word2vec scores for each sentence.

Here we created our Model Using existing IMDB Movie Review Dataset from Kaggle's Bag of words to Bag of Popcorns challenge.

Original Similarity Measure Used in Text Rank:

Text Rank is an unsupervised algorithm for the automated summarization of texts that can extract the most important word in a document. This algorithm models any documents as a graph using sentences as nodes. Between those nodes, a similarity function is needed to build edges. The higher the similarity between sentences, the more important the edge between them will be in the graph.

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

S_i, S_j are two sentences we need to compute the similarity. The result of this process is a dense graph representing the document.

4. Graph Creation:

These sentences were transformed into a graph of related sentences. Based on how similar two sentences in a review are, we built a graph such that each node in the graph represented a sentence and each edge represented the similarity measure between two nodes (sentences). However, to create the graph, we set a threshold for the similarity, above which we accept that any two sentences are “similar”.

5. Page Rank for Text Summarization:

Given a set of nodes and the relationships between these nodes, PageRank provides us with a means of identifying which amongst these nodes is the most important. Because of the existence of an edge between similar sentences, running PageRank on this and choosing the top “m” sentences will give us the summary. Though there are sophisticated NLP algorithms that can alter the structure of a text document and produce a summary of the document much like a human generated one, we made use of PageRank because it’s an unsupervised method of summarizing a document without needing a lot of data. It requires no additional information about the article to perform summarization. We made use of the “networkx.pagerank” library to implement the PageRank algorithm.

6. Evaluation:

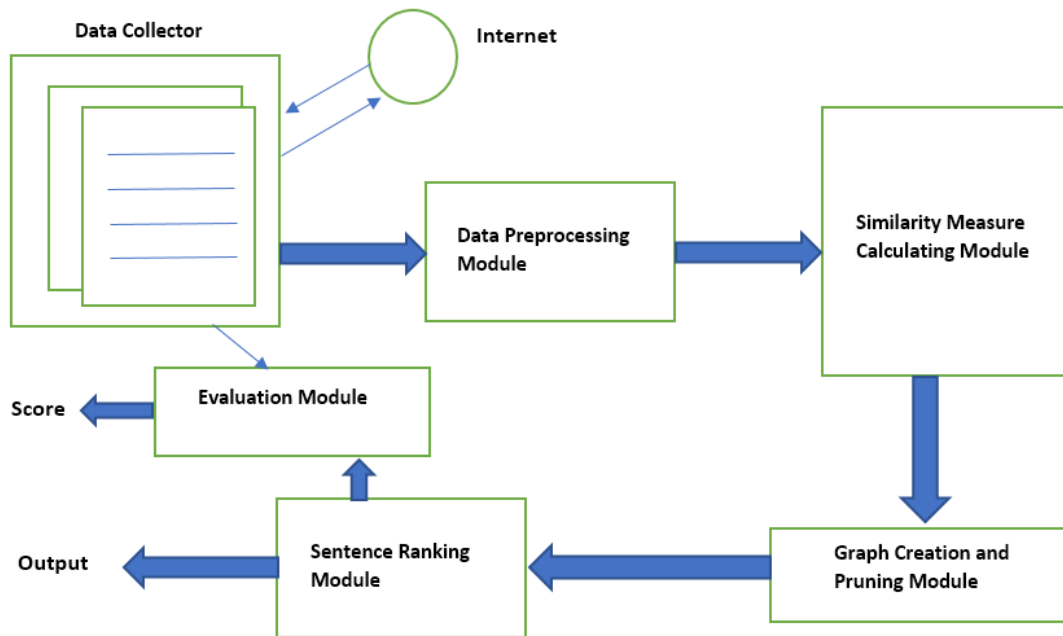
Running the page rank algorithm on our dataset generated a tuple consisting the rank of the sentence in the review and the sentence itself. Here we considered the highest ranked sentence to be the summary of the entire movie review. We can also consider top “m” ranked sentences to be the summary of the movie review. Once we had the system generated review, we compared the 1-line summary extracted from Rotten Tomatoes with this review.

To evaluate how close the system generated review and the 1-line summary are we make use of the following evolution measures:

- ROUGE - N
- ROUGE – L

ROUGE-N and ROUGE-L can be thought of as the granularity of texts being compared between the system summaries and reference summaries. For example, ROUGE-1 refers to overlap of unigrams between the system summary and reference summary. ROUGE-2 refers to the overlap of bigrams between the system and reference summaries. From these evaluation methods, we calculated F-1 scores and the longest common subsequence length of the two summaries.

The entire above approach can be summarized in the following picture:



Experiment:

Dataset:

As mentioned above, the dataset is a set of reviews scraped from RottenTomatoes and the linked critic personal review websites. The data is stored in a pickle file with relevant information such as the movie id, movie name, the RottenTomato 1-line summary, the entire critic review.

Baseline Methods:

The baselines used here are the first and the last sentences of the summary. This was done due to one of the assumptions of the project, that the first line and the last line of the review are most descriptive of the entire review.

Evaluation:

The evaluation methods used in this project are ROUGE - N and ROUGE - L. This assumes that 1-line summary extracted from RottenTomatoes are human picked sentences and hence we use that as our human generated summary in our ROUGE scores. ROUGE-n calculates the number of overlapping n-grams between RottenTomatoes 1-line summary and the system generated 1-line summary. It calculates precision as a fraction of the number of n-grams in the RottenTomatoes Summary and recall as a fraction of the number of n-grams in the system generated Summary. Finally, a F-1 score is calculated from precision and recall. ROUGE - L is another evolution measure which considers the longest common subsequence of RottenTomatoes 1-line summary and the system generated 1-line summary. Once we estimate the ROUGE-n and ROUGE-L scores, we store in a panda's data frame and store it in a pickle file.

For every similarity measure (tf-idf, bm25plus, jaccard, word2vec, n-gram and text rank), we repeat the above process and compute the average ROUGE-N and average ROUGE -L scores for each of them.

Results and Observations:

The task of selecting the best summary sentence out of a review is inherently somewhat ill posed. We based our model on the assumption that the “best” sentence was exactly the 1-line summary that RottenTomatoes picked. However, there may be several very good summary sentences, and RT’s choice may be somewhat arbitrarily. We manually went through 50 movie reviews to determine a baseline for how difficult it was to guess the same sentence that RottenTomatoes.com did. Each of us picked one line from the review and considered it to be the one line summary of the entire review. We then compared this with the RT on-line summary. The results were surprising. We got about 21 of the 50 one-line summaries to be similar. Our rough estimation for human performance of this task was therefore around 42% for locating the sentence that RottenTomatoes picked out.

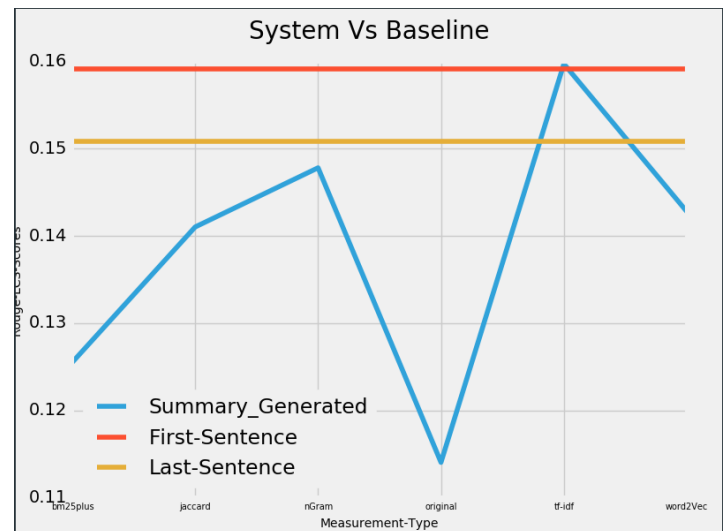
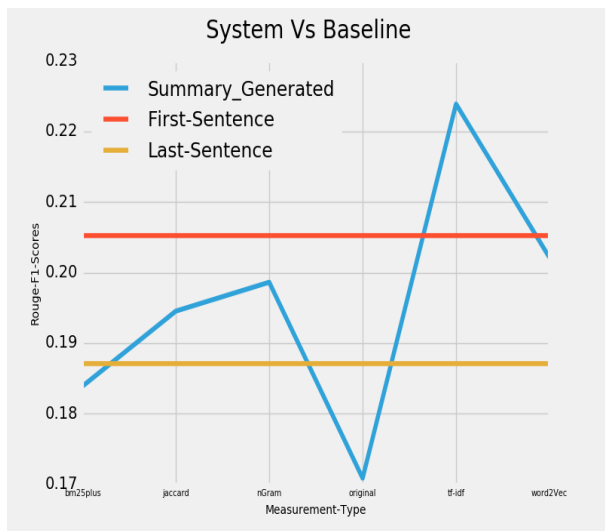
1. Using Threshold = 0.5

Rouge-N Metric for Average F1’s

	Similarity measure	Our Summary	First Sentence	Last Sentence
0	bm25plus	0.183862	0.205222	0.187022
1	jaccard	0.194523	0.205222	0.187022
2	nGram	0.198620	0.205222	0.187022
3	original	0.170838	0.205222	0.187022
4	tf-idf	0.223887	0.205222	0.187022
5	word2Vec	0.202127	0.205222	0.187022

Rouge-L Metric for Average F1’s

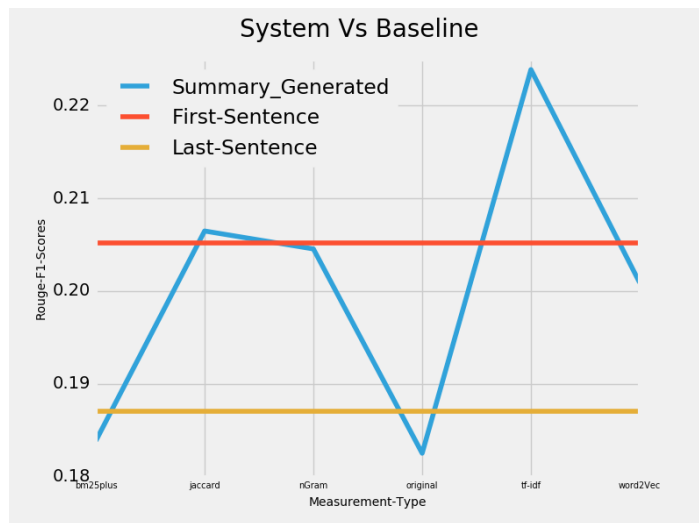
	Similarity measure	Our Summary	First Sentence	Last Sentence
0	bm25plus	0.125506	0.159107	0.150852
1	jaccard	0.141023	0.159107	0.150852
2	nGram	0.147806	0.159107	0.150852
3	original	0.114035	0.159107	0.150852
4	tf-idf	0.159723	0.159107	0.150852
5	word2Vec	0.142790	0.159107	0.150852



2. Using Threshold = 0.3

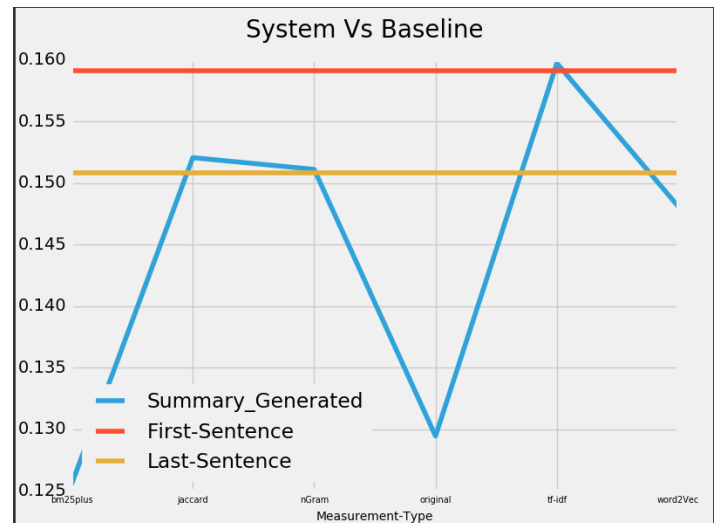
Rouge-N Metric for Average F1's

	Similarity measure	Our Summary	First Sentence	Last Sentence
0	bm25plus	0.183862	0.205222	0.187022
1	jaccard	0.206482	0.205222	0.187022
2	nGram	0.204551	0.205222	0.187022
3	original	0.182520	0.205222	0.187022
4	tf-idf	0.223887	0.205222	0.187022
5	word2Vec	0.200905	0.205222	0.187022



Rouge-L Metric for Average F1's

	Similarity measure	Our Summary	First Sentence	Last Sentence
0	bm25plus	0.125506	0.159107	0.150852
1	jaccard	0.152058	0.159107	0.150852
2	nGram	0.151106	0.159107	0.150852
3	original	0.129455	0.159107	0.150852
4	tf-idf	0.159723	0.159107	0.150852
5	word2Vec	0.148094	0.159107	0.150852



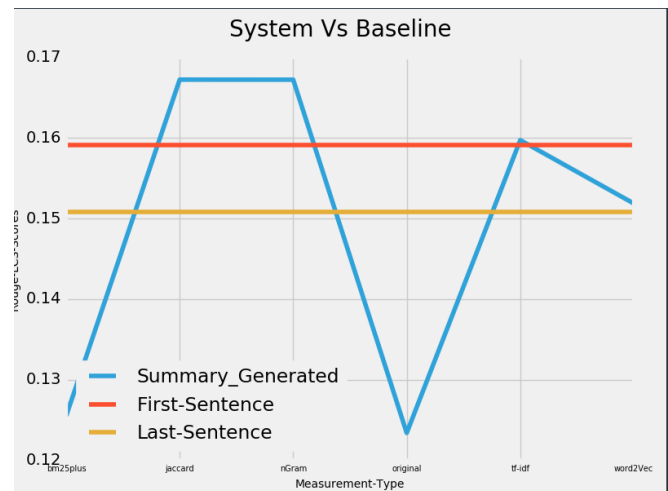
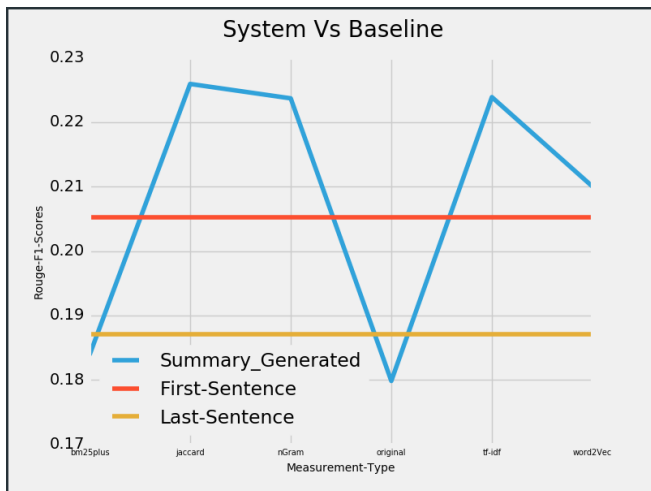
3. Using Threshold = 0.1

Rouge-N Metric for Average F1's

	Similarity measure	Our Summary	First Sentence	Last Sentence
0	bm25plus	0.183862	0.205222	0.187022
1	jaccard	0.225936	0.205222	0.187022
2	nGram	0.223697	0.205222	0.187022
3	original	0.179830	0.205222	0.187022
4	tf-idf	0.223887	0.205222	0.187022
5	word2Vec	0.210009	0.205222	0.187022

Rouge-L Metric for Average F1's

	Similarity measure	Our Summary	First Sentence	Last Sentence
0	bm25plus	0.125506	0.159107	0.150852
1	jaccard	0.167234	0.159107	0.150852
2	nGram	0.167227	0.159107	0.150852
3	original	0.123384	0.159107	0.150852
4	tf-idf	0.159723	0.159107	0.150852
5	word2Vec	0.151866	0.159107	0.150852



Example:

- Highest Scoring Summary:** b'fairly one note in its humor and not as lively as you would assume it would be but with all around strong voice work and a predictably sweet message about sharing the love it\xe2\x80\x99s all as they say good enough for government work '

First Sentence: b'fairly one note in its humor and not as lively as you would assume it would be but with all around strong voice work and a predictably sweet message about sharing the love it\xe2\x80\x99s all as they say good enough for government work '

Last Sentence: b'add all around strong voice work and a predictably sweet message about sharing the love and it\xe2\x80\x99s all as they say good enough for government work '

RT Summary: b' fairly one note in its humor and not as lively as you would assume it would be but with all around strong voice work and a predictably sweet message about sharing the love it s all as they say good enough for government work '
- Highest Scoring Summary:** b'while dreamworks animation s latest movie starts well and ends sweetly the loud frenetic middle seems like an awfully good time to squeeze in a nap '

First Sentence: b' cnn the boss baby milks a fertile premise until it feels about as perfunctory as corporate drudgery '

Last Sentence: b'read more'

RT Summary: b' while dreamworks animation s latest movie starts well and ends sweetly the loud frenetic middle seems like an awfully good time to squeeze in a nap '

Conclusion:

From this project, we have learnt that the study of automated text summarization still has a long way to go before we can really claim to understand the nature of summaries. Evaluation of such a system is even harder since as mentioned earlier, a perfect summary depends on the person wanting the summary. There are several problems that hinder the development of summarization systems. One of the primary problems is not having good training data in different domains and for a variety of summary types. A good test summarization needs full text, human abstract and the corresponding extracted summaries. In the future we would like to try our hand at Abstract text summarization using sequence to sequence neural networks. For development of better text summarization in future, we would firstly need to build a huge dataset and use it for experiments and make our systems powerful. We would need to build a stronger understanding of the word similarities and estimate the similarities of two sentences.

References:

Links to the above- mentioned Scientific Papers:

<http://anthology.aclweb.org/C/C08/C08-1124.pdf>

<http://www.aclweb.org/anthology/P04-3020>

<http://www.cis.drexel.edu/faculty/thu/research-papers/dawak-547.pdf>

<https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf>

<https://www.jair.org/media/1523/live-1523-2354-jair.pdf>