TEXT SUMMARIZATION AN OVERVIEW

A Major Project Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Kurre Lokanadh Reddy	**121710301027**
Dudyala Thulaseeswara Reddy	**121710301015**
Gumpina Arun	**121710301020**
Peketi Varun	**121710301040**

Under the esteemed guidance of

Mrs. D. Suneetha

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM (Deemed to be University)

VISAKHAPATNAM

DECEMBER 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM
(Deemed to be University)

VISAKHAPATNAM

DECLARATION

We, hereby declare that the Major Project report entitled " **Text Summarization an Overview** " is an original work done in the Department of Computer Science and Engineering, GITAM Institute Of Technology, GITAM (Deemed To Be University, Visakhapatnam) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering . The work has not been submitted to any other college or University for the award of any degree or diploma.

Name(s)	Registration Number (s)
Kurre Lokanadh Reddy	**121710301027**
Dudyala Thulaseeswara Reddy	**121710301015**
Gumpina Arun	**121710301020**
Peketi Varun	**121710301040**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM
(Deemed to be University)

VISAKHAPATNAM

CERTIFICATE

This is to certify that the Major Project report entitled " **Text Summarization an Overview"** is a Bonafide record of work carried out by KURRE LOKANADH REDDY (121710301027),DUDYALA THULASEESWARA REDDY(121710301014), GUMPINA ARUN(121710301020), PEKETI VARUN (121710301040) submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

Project Guide:
Mrs. D. Suneetha
Assistant Professor
Department of CSE
GITAM Institute Of Technology

Head of the Department:
Dr. Konala Thammi Reddy
Professor
Department of CSE
GITAM Institute Of Technology

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to our Project Guide **Mrs. D. Suneetha**, Assistant
Professor, Department of CSE who took time to hear us out, guide and keep us on the righteous path so as to carry out our work at their esteemed organization. We had a great chance of learning and professional development. It has been a great privilege to work under his continuous assistance and support.

We would also like to thank **Y. Srinivas,** Professor and **Dr. Srinivas L. Chakravarthy**, Department of CSE for their part in useful decisions, giving necessary suggestions and guidance for the completion of our project.

We express our deepest thanks to **Prof. K. Thammi Reddy**, Head of the Department, Computer Science and Engineering, GITAM (Deemed to be University) for his valuable suggestions and constant motivation that helped us to complete the project. We choose this moment to acknowledge his assistance gratefully.

We perceive this opportunity as a big milestone in our career development. We assure to continue the work on their improvement, in order to attain desired career objectives. Finally, it has been a great pleasure to use gained skills and knowledge and thanks to everyone who helped us directly and indirectly throughout this project.

TABLE OF CONTENTS

ABSTRACT

In this new era, where tremendous information is available on the Internet, it is most important to provide the improved mechanism to extract the information quickly and most efficiently. It is very difficult for human beings to manually extract the summary of a large documents of text. There are plenty of text material available on the Internet. So, there is a problem of searching for relevant documents from the number of documents available, and absorbing relevant information from it. In order to solve the above two problems, the automatic text summarization is very much necessary. Text summarization is the process of identifying the most important meaningful information in a document or set of related documents and compressing them into a shorter version preserving its overall meanings. Though there are many advanced models to achieve the best summary its also important that the model is cost effective and easy to maintain, these neceisitie give the importance to short algorathamic models. The project covers some of the well known models with their process and implementations from scratch.

INTRODUCTION

Text summarization is a process of extracting or collecting important information from original text and presents that information in the form of summary. In recent years, need for summarization can be seen in various purpose and in many domains such as news articles summary, email summary, short message of news on mobile, and information summary for businessman, government officials, researchers online search through search engine to receive the summary of relevant pages found, medical field for tracking patient's medical history for further treatment. On the internet, many such examples are available like, news article summarizer such as Microsoft News2, Google1 or Columbia Newsblaster3. BaseLine, FreqDist, SumBasic, MEAD, AutoSummarize & SWESUM is few popular biomedical summarization tools. Text Compacter, Sumplify, Tools4Noobs, FreeSummarizer, WikiSummarizer & SummarizeTool are online summarization tools. Open Text summarizer, Classifier4J, NClassifier, CNGLSummarizer are few widely used open source summarization tools. The need of having information in abstract form on a click has increased as, the need for automatic text summarization has also increased in many areas namely, news articles summary, email summary, short message news on mobile and information summary for business, government officials research, online research engines to receive summary. In late 1950, the first system came in term; The automatic summarizer in general selects important sentences from the document and groups them together, it consumes less time or time saving to understand the content within the large document. The aim of automatic text summarization is to convert large document into shorter one and store important content. The automatic summarization of text is a well- known task in the field of natural language processing (NLP). Significant achievements in text summarization have been obtained using sentence extraction and statistical analysis. Text summarization approaches can be broadly divided into two groups: extractive summarization and abstractive summarization. Extractive summarizations extract important sentences or phrases from the original documents and group them to produce a summary without changing the original text. An extractive text summarization system is proposed based on POS tagging by considering Hidden Markov Model using corpus to extract important phrases to build as a summary. Abstractive summarization consists of understanding the source text by using linguistic method to interpret and examine the text. Abstractive methods need a deeper analysis of the text. These methods have the ability to generate new sentences, which improves the focus of a summary, reduce its redundancy and keeps a good compression rate.

In the 2014 book on the subject titled "Automatic Text Summarization," the authors provide 6 reasons why we need automatic text summarization tools.

1. Summaries reduce reading time.
2. When researching documents, summaries make the selection process easier.
3. Automatic summarization improves the effectiveness of indexing.
4. Automatic summarization algorithms are less biased than human summarizers.
5. Personalized summaries are useful in question-answering systems as they provide personalized information.
6. Using automatic or semi-automatic summarization systems enables commercial abstract services to increase the number of texts they are able to process.

There are many reasons and uses for a summary of a larger document.

One example that might come readily to mind is to create a concise summary of a long news article, but there are many more cases of text summaries that we may come across every day.

In the 1999 book on the topic titled “Advances in Automatic Text Summarization,” the authors provide a useful list of every-day examples of text summarization.

- headlines (from around the world)
- outlines (notes for students)
- minutes (of a meeting)
- previews (of movies)
- synopses (soap opera listings)
- reviews (of a book, CD, movie, etc.)
- digests (TV guide)
- biography (resumes, obituaries)
- abridgments (Shakespeare for children)
- bulletins (weather forecasts/stock market reports)
- sound bites (politicians on a current issue)
- histories (chronologies of salient events)

This project is covers some of the highlights of the evolution in text summarization with mostly using extractive approaches and few hybrid and deep learning based approaches. However, results of deep learning methods are not yet state-of-the-art compared to extractive methods, yet impressive results have been achieved on constrained problems such as generating headlines for news articles that rival or out-perform other abstractive methods.

PROBLEM STATEMENT

The increasing availability of large and unstructured text data, which contains huge amount of information hidden in it's core. This data can be used in various way to enhance the business and also take the machine understandings to a new level. Though there are very successful models like Google's PEGUSUS , are there, the importance and need is still distributed equally among various small algorthimic extractive models in the same way there is on the abstractive Neural Network models.

The situations, range, affordable cost, time all these factors play a key role in choosing the model that suits well. The project covers such different models with their implementations and deployment on a web platform, Covering all the hidden difficulties in the process of implementing the code from scratch.

UNDERSATANDING

What is text summarization

Automatic text summarization, or just text summarization, is the process of creating a short and coherent version of a longer document. *Text summarization is the process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user (or users) and task (or tasks).* Or in other words, *Automatic text summarization is the task of producing a concise and fluent summary while preserving key information content and overall meaning.*

We (humans) are generally good at this type of task as it involves first understanding the meaning of the source document and then distilling the meaning and capturing salient details in the new description. As such, the goal of automatically creating summaries of text is to have the resulting summaries as good as those written by humans, which is referred to as gold standards.

The machine needs to understand the source document first then the process of summarization begins. There is case where this understanding is difficult to achieve given different factors against it, like wise for humans when there is no time left do all that process. Then we go for writing some of sentences as it is from the source text. The machines also made to do in such cases.

There are two main approaches to summarizing text documents; they are:

1. Extractive Methods.

2. Abstractive Methods.

Extractive text summarization involves the selection of phrases and sentences from the source document to make up the new summary. Techniques involve ranking the relevance of phrases in order to choose only those most relevant to the meaning of the source.

Abstractive text summarization involves generating entirely new phrases and sentences to capture the meaning of the source document. This is a more challenging approach, but is also the approach ultimately used by humans. Classical methods operate by selecting and compressing content from the source document.

Classically, most successful text summarization methods are extractive because it is an easier approach, but abstractive approaches hold the hope of more general solutions to the problem.

But in-depth there are more than 2 ways to do this considering all the process involved at different levels.

Different ways of summarization

There are no fixed guidelines for categorization of the techniques that we use for it. Although for performing tasks in an organized way they are generally be divided into these following types:

1. Short Tail Summarization: In this type of summary the input content is very short and precise. Even after having a short length it needs to be summarized in such a way that it could be bounded further without any change in its meaning.

2. Long Tail Summarization: As you might have already grasped by the name. The content here could be too long to be handled by a human being alone. It could contain text data from thousands of pages and books at once.

3. Single Entity: When the input usually contains elements from just one source.

4. Multiple Entities: When the input contains elements from different document sources. This is one of the most useful applications of this technique.

5. General Purpose: In this type of Text Summarization Python has no attribute for the type of input is provided. The algorithm does not have a sense of the domain in which the text deals. It is performed after multiple training of algorithms on various types of textual content.

6. Domain-Specific: They are performed under a specific domain each time. For example, a Text Summarization algorithm that summarizes the Food recipe in just a few words. So it does have a domain of Food recipes and knows the context behind the data.

7. Informative Summarization: This summary keeps all the information related to the actual content. No change in meaning is seen in the output results.

8. Headlines Generation: News channels and applications use this type of summary. Headlines are generated according to the text content of an article.

9. Keyword Extraction: Only the most important keywords and phrases are extracted from the whole data. For example, extracting the phrases where some verbal conversation is going on and leaving all the narrations behind.

The alternate way in python

As said above the code for this project is written from scratch in order to help the understandings of the processes and sub-process involved. But there are many python packages that help us in doing this, which are developed for the sole purpose of summarization and its related applications. Here is short note to know them to some extent.

Current well known models

DATA SET

CNN daily data set

The CNN / DailyMail Dataset is an English-language dataset containing just over 300k unique news aticles as written by journalists at CNN and the Daily Mail. The current version supports both extractive and abstractive summarization, though the original version was created for machine reading and comprehension and abstractive question answering.

Language: The CNN / DailyMail Dataset is an English-language dataset containing just over 300k unique news aticles as written by journalists at CNN and the Daily Mail. The current version supports both extractive and abstractive summarization, though the original version was created for machine reading and comprehension and abstractive question answering.

Dataset Structure

Data Instances: For each instance, there is a string for the article, a string for the highlights, and a string for the id. See the CNN / Daily Mail dataset viewer to explore more examples.

{'id': '0054d6d30dbcad772e20b22771153a2a9cbeaf62',

'article': '(CNN) -- An American woman died aboard a cruise ship that docked at Rio de Janeiro on Tuesday, the same ship on which 86 passengers previously fell ill, according to the state-run Brazilian news agency, Agencia Brasil. The American tourist died aboard the MS Veendam, owned by cruise operator Holland America. Federal Police told Agencia Brasil that forensic doctors were investigating her death. The ship's doctors told police that the woman was elderly and suffered from diabetes and hypertension, according the agency. The other passengers came down with diarrhea prior to her death during an earlier part of the trip, the ship's doctors said. The Veendam left New York 36 days ago for a South America tour.'

'highlights': 'The elderly woman suffered from diabetes and hypertension, ship's doctors say .\nPreviously, 86 passengers had fallen ill on the ship, Agencia Brasil says .'}

The average token count for the articles and the highlights are provided below:

Feature	Mean Token Count
Article	781
Highlights	56

Data Fields

id: a string containing the heximal formated SHA1 hash of the url where the story was retrieved from

article: a string containing the body of the news article

highlights: a string containing the highlight of the article as written by the article author

Data Splits

The CNN/DailyMail dataset has 3 splits: train, validation, and test. Below are the statistics for Version 3.0.0 of the dataset.

Dataset Split	Number of Instances in Split
Train	287,113
Validation	13,368
Test	11,490

However, we mostly use this dataset for evaluation of all the developed models, and for the training of Seq2Seq model.

GloVe: Global Vectors for Word Representation

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

Training: The GloVe model is trained on the non-zero entries of a global word-word co-occurrence matrix, which tabulates how frequently words co-occur with one another in a given corpus. Populating this matrix requires a single pass through the entire corpus to collect the statistics. For large corpora, this pass can be computationally expensive, but it is a one-time up-front cost. Subsequent training iterations are much faster because the number of non-zero matrix entries is typically much smaller than the total number of words in the corpus.

The tools provided in this package automate the collection and preparation of co-occurrence statistics for input into the model. The core training code is separated from these preprocessing steps and can be executed independently.

Once the training is done it gives a one dimensional vector for every word that occurred. These vectors are stored and can be used in the conversion words to numerical vectors. The same is done in this project when needed using transfer learning using the Word2Vec layer.

ENVIRONMENT

Python

The entire code is developed in python language from scratch. The availability of the packages such as NLTK, Numpy, SkLearn made python the first choice for implementation. The entire models are developed around Numpy, Sklearn and NLTK. With NLTK taking care of the pre-processing and conversion of human language into numerical vectors and, Numpy and SkLearn taking care of the calcualations and fuzzy or higher level caleculations respectively.

The entire code except the feature based and Seq2Seq is written in a single file to make good use of the sub-modules. The fuzzy code and Seq2Seq code are written in separate files to reduce to complexeties. And are stored as sibiling directories for the main python file.

Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions

The main python file is the one that implements the Flask code, which is used to integrate the web Languages(HTML,CSS,JS) with the python code. The code excecution starts from here once the deployment is done. The flask settings take the default path for the HTML templates as templates and other files under static. The deployment in the local environment is easy, less complex and uses a single dino. So the python compiler handles this on its own with the help of Flask.

Heroku

Heroku is a cloud platform as a service supporting several programming languages. One of the first cloud platforms, Heroku has been in development since June 2007, when it supported only the Ruby programming language, but now supports Java, Node.js, Scala, Clojure, Python, PHP, and Go.

The entire code is uploaded to github and is deployed on Heroku using gunicorn python package. The Gunicorn "Green Unicorn" is a Python Web Server Gateway Interface HTTP server. It is a pre-fork worker model, ported from Ruby's Unicorn project. The Gunicorn server is broadly compatible with a number of web frameworks, simply implemented, light on server resources and fairly fast.

METHODOLOGY

PRE-PROCESSING

In any machine learning task, cleaning or pre-processing the data is as important as model building if not more. And when it comes to unstructured data like text, this process is even more important.

Objective of this kernel is to understand the various text preprocessing steps with code examples.

Some of the common and basic text preprocessing / cleaning steps are:

- Lower casing
- Removal of Punctuations
- Removal of Stopwords
- Removal of Frequent words
- Removal of Rare words
- Stemming
- Lemmatization

So these are the different types of text preprocessing steps which we can do on text data. But we need not do all of these all the times. We need to carefully choose the preprocessing steps based on our use case since that also play an important role.

For example, in sentiment analysis use case, we need not remove the emojis or emoticons as it will convey some important information about the sentiment. Similarly we need to decide based on our use cases. And in case of text summarization with transfer learning the model learns to create sentences we should not remove any stopwords, with out which the sentence in never complete. On the other hand we need to elemenate the word redundancy in its verb forms by using lemmatization or stemming with out which the word corpra for that model will be large and might lead to long execution times and inaccurate results.

So it is always important to do the pre-processing of the data in the best way possible for that particular model. As we are implementing different models in this project ,we have 3 different pre-processing methods. All taking the same input and giving the output in same format as words list for each sentences.

1. Simple Tokenization:

This method takes the text as input and returns the word list back. The model first removes all the spaces and punctuations following with sentence tokenization. After the sentences are tokenzeed to words and the stopwords are removed from every sentence.

This method is used in processes like LSA ,Luhn where it is no compulsion to the amount of words and no need for the words to be in the English word corpra.

Lower Casing

Lower casing is a common text preprocessing technique. The idea is to convert the input text into same casing format so that 'text', 'Text' and 'TEXT' are treated the same way. This

is more helpful for text featurization techniques like frequency, tfidf as it helps to combine the same words together thereby reducing the duplication and get correct counts / tfidf values. This may not be helpful when we do tasks like Part of Speech tagging (where proper casing gives some information about Nouns and so on) and Sentiment Analysis (where upper casing refers to anger and so on).

By default, lower casing is done my most of the modern day vecotirzers and tokenizers like sklearn TfidfVectorizer and Keras Tokenizer. So we need to set them to false as needed depending on our use case.

Removal of Punctuations

One another common text preprocessing technique is to remove the punctuations from the text data. This is again a text standardization process that will help to treat 'hurray' and 'hurray!' in the same way. We also need to carefully choose the list of punctuations to exclude depending on the use case. For example, the string.punctuation in python contains the following punctuation symbols

!"#$%&\'()+,-./:;<=>?@[\\]^_{|}~ `*

We can add or remove more punctuations as per our need. In this case we don't want to remove the "." With out which we can not identify the sentences in the text.

Removal of stopwords

Stopwords are commonly occuring words in a language like 'the', 'a' and so on. They can be removed from the text most of the times, as they don't provide valuable information for downstream analysis. In cases like Part of Speech tagging, we should not remove them as provide very valuable information about the POS. These stopword lists are already compiled for different languages and we can safely use them.

2. Stemming on Verbs:

This process is an extension to the normal method with stopwords being removed or kept according to the need and situation.

Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form. For example, if there are two words in the corpus walks and walking, then stemming will stem the suffix to make them walk. But say in another example, we have two words console and consoling, the stemmer will remove the suffix and make them console which is not a proper English word. There are several type of stemming algorithms available and one of the famous one is porter stemmer which is widely used.

We can see that words like private and propose have their e at the end chopped off due to stemming. These will lead to unintended results ,for example we are getting the Embedding vectors and most of the words got chopped off then by default we assign them as unknown. This is not intended. What can we do fort hat? We can use Lemmatization in such cases.

3. Lemmataization :

Lemmatization is similar to stemming in reducing inflected words to their word stem but differs in the way that it makes sure the root word (also called as lemma) belongs to the language.

For finding the proper lemma before rooting it takes scans up and down the wordNet. As a result, this one is generally slower than stemming process. So depending on the speed requirement, we can choose to use either stemming or lemmatization.

This is default for most of the cases of Embedding and Neural Network related models.

ALGORITHEMS USED

Tf-idf (Term Frequency-Inverse Document Frequency): the cell is filled in with the tf-idf value of the word.
A higher tf-idf value means that the word is more frequent in the sentence but less frequent in the whole
document. A higher value also indicates that the word is much more representative for that sentence than
others.
It is a function that assigns score to words based on their frequency in the collection of tweets. This score reflects importance of a word, w, in a sentence,s.

$$\textbf{TF-IDF(s,w) = TF (s,w) * IDF (w)}$$

Term frequency TF(s,w) : number of times the word, w occurs in the sentence, s.
Inverse document frequency (IDF(w)):Log (number of tweets containing the word,w)/(total number of tweets)

Singular Value Decomposition: SVD is an algebraic method that can model relationships among words/phrases and sentences. In this method, the given input matrix A is decomposed into three new matrices as follows:
A = U P VT
where A is the input matrix (m × n); U is words × extracted concepts (m × n); P represents scaling values, diagonal descending matrix (n × n); and V is sentences × extracted concepts (n × n).

Modified Tf-idf: this approach is proposed in Ozsoy et al. [3], in order to eliminate noise from the input matrix.
The cell values are set to tf-idf scores first, and then the words that have scores less than or equal to the average of the row are set to 0.

Cross method. The cross method is an extension to the approach of Steinberger Jezek [5], proposed in Ozsoy et al.[3]. In this approach, input matrix creation and SVD calculation steps are executed as in the other approaches and the VT matrix is used for sentence selection purposes. Between the SVD calculation step and the sentence selection step, there exists a pre-processing step. The aim of the pre-processing step is to remove the overall effect of sentences that are related to the concept somehow, but not the core sentence for that concept. For each concept, which is represented by the rows of the VT matrix, the average sentence score is calculated. Then the cell values which are less than or equal to the average score are set to zero. This process of setting the cell values to zero whose score is less than average removes less related sentences while keeping more related ones for that concept.

V^T matrix ($k = 2$)				
	Sent0	Sent1	Sent2	Avg.
Con0	~~0.457~~ 0	0.728	~~0.510~~ 0	0.565
Con1	~~-0.770~~ 0	0.037	0.637	-0.021
Length	0	0.765	0.637	

After pre-processing, the steps of Steinberger and Jezek approach are followed with a modification. In our cross approach, the total length of each sentence vector, which is represented by a column of the VT matrix, is calculated. While calculating the length score, the parameter of number of concepts to be used is given by the user; if it is not given, all of the extracted concepts are used. Then, the longest sentence vectors are collected as a part of the resulting summary.
In Figure 5, an example VT matrix is given after the pre-processing is executed. For the pre-processing step, first the average score for each concept is calculated, and then the cell values less than this average are set to zero. Finally, length scores are calculated by adding up the concept scores with values after the pre-processing step. In this example matrix, sen1 has the highest length score, so it has been chosen to be part of the summary.

Topic Clustering
K-means clustering
The k-means clustering algorithm is an unsupervised clustering algorithm which determines the optimal number of clusters using the elbow method. It works iteratively by selecting a random coordinate of the cluster center and assign the data points to a cluster. It then calculates the Euclidean distance of each data point from its centroid and based on this, it updates the datapoint positions as well as the cluster centers.

For minimizing the Within cluster sum of squares(WCSS) the following formula is used :

$$arg\ arg\ min_{\ S}\ \sum_{i=1}^{k} \quad \frac{1}{2|S_i|} \sum_{x,y \in S_i} \quad \|x - y\|^2$$

Where the mean points and x contains observations in a d-dimensional vector and k is the number of cluster centers.

For finding the optimum number of centroids, the ‘elbow method’ is used. In it, the SSE value(sum of squared error) is calculated for different values of k (i.e no. of clusters) by clustering the dataset following each value of k. The point on the graph where a ‘hinge’ occurs is considered to be the optimal value of k. Figure 3. shows the elbow method for k means algorithm.
Using the cluster numbers obtained from the elbow method, we use the k-means algorithm to predict the labels. These cluster values can be used in the feature generation or in sentence selection.

TEXT RANK METHOD

Graph-based ranking algorithms are essentially a way of deciding the importance of a vertex within a graph, based on global information recursively drawn from the entire graph. The basic idea implemented by a graph-based ranking model is that of “voting” or “recommendation”. When one vertex links to another one, it is basically casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex. Moreover, the importance of the vertex casting the vote determines how important the vote itself is, and this information is also taken into account by the ranking model. Hence, the score associated with a vertex is determined based on the votes that are cast for it, and the score of the vertices casting these vote.

Formally, let G= (V,E) be a directed graph with the set of vertices V and set of edges E ,where E is a subset of V x V . For a given vertex V_i , let $In(V_i)$ be the set of vertices that point to it (predecessors), and let $Out(V_i)$ be the set of vertices that vertex V_i points to (successors). The score of a vertex V_i is defined as follows (Brin and Page, 1998):

where d is a damping factor that can be set between 0 and 1, which has the role of integrating into the model the probability of jumping from a given vertex to another random vertex in the graph. In the context of Web surfing, this graph-based ranking algorithm implements the "random surfer model", where a user clicks on links at random with a probability d, and jumps to a completely new page with probability 1 – d. The factor d is usually set to 0.85 (Brin and Page, 1998), and this is the value we are also using in our implementation.

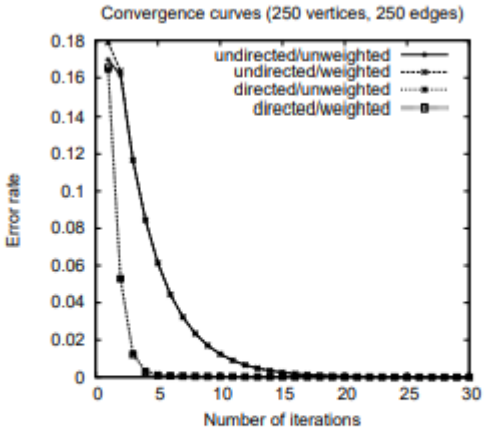


Figure 1: Convergence curves for graph-based ranking: directed/undirected, weighted/unweighted graph, 250 vertices, 250 edges.

Figure 1 plots the convergence curves for the same sample graph from section 2.1, with random weights in the interval 0–10 added to the edges. While the final vertex scores (and therefore rankings) differ significantly as compared to their unweighted alternatives, the number of iterations to convergence and the shape of the convergence curves is almost identical for weighted and unweighted graph.

Convergence is achieved when the error rate for any vertex in the graph falls below a given threshold. The error rate of a vertex V_i is defined as the difference between the "real" score of the vertex $S(V_i)$ and the score computed at iteration k, $S^k(V_i)$. Since the real score is not known apriori, this error rate is approximated with the difference between the scores computed at two successive iterations: $S^{k-1}(V_i) - S^k(V_i)$.

Starting from arbitrary values assigned to each node in the graph, the computation iterates until convergence below a given threshold is achieved. After running the algorithm, a score is associated with each vertex, which represents the "importance" of the vertex within the graph. Notice that the final values obtained after TextRank runs to completion are not affected by the choice of the initial value, only the number of iterations to convergence may be different.

It is important to notice that although the TextRank applications described in this paper rely on an algorithm derived from Google's PageRank (Brin and Page, 1998), other graph-based ranking algorithms such as e.g. HITS (Kleinberg, 1999) or Positional Function (Herings et al., 2001) can be easily integrated into the TextRank model.

Text as a Graph To enable the application of graph-based ranking algorithms to natural language texts, we have to build a graph that represents the text, and interconnects words or other text entities with meaningful relations. Depending on the application at hand, text units of various sizes and characteristics can be added as vertices in the graph, e.g. words, collocations, entire sentences, or others. Similarly, it is the application that dictates the type of relations that are used to draw connections between any two such vertices, e.g. lexical or semantic relations, contextual overlap, etc.

Regardless of the type and characteristics of the elements added to the graph, the application of graph-based ranking algorithms to natural language texts consists of the following main step:

1. Identify text units that best define the task at hand, and add them as vertices in the graph.

2. Identify relations that connect such text units, and use these relations to draw edges between vertices in the graph. Edges can be directed or undirected, weighted or unweighted.

3. Iterate the graph-based ranking algorithm until convergence.

4. Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions.

LUHN'S METHOD

Measuring significance

To determine which sentences of an article may best serve as the auto-abstract, a measure is required by which the information content of all the sentences can be compared and graded. Since the suitability of each sentence is relative, a value can be assigned to each in accordance with the quality criterion of significance.

The "significance" factor of a sentence is derived from an analysis of its words. It is here proposed that the frequency of word occurrence in an article furnishes a useful measurement of word significance. It is further proposed that the relative position within a sentence of words having given values of significance furnishes a useful measurement for determining the significance of sentences. The significance factor of a sentence will therefore be based on a combination of these two measurements.

It should be emphasized that this system is based on the capabilities of machines, not of human beings. Therefore, regrettable as it might appear, the intellectual aspects of writing and of meaning cannot serve as elements of such machine systems. To a machine, words can be only so many physical things. It can find out whether or not certain such things are similar and how many of them there are. The machine carne member such findings and can perform arithmetic on those which can be counted. It can do all of this by means of suitable program instructions. The human intellect need be relied upon only to prepare the program.

Establishing a set of significant words

The justification of measuring word significance by use-frequency is based on the fact that a writer normally repeats certain words as he advances or varies his arguments and as he elaborates on an aspect of a subject. This means of emphasis is taken as an indicator of significance. The more often certain words are found in each other's company within a sentence, the more significance may be attributed to each of these words. Though certain other words must be present to serve the important function of tying these words together, the type of significance sought here does not reside in such words. If such common words can be segregated substantially by non-intellectual methods, they could then be excluded from consideration.

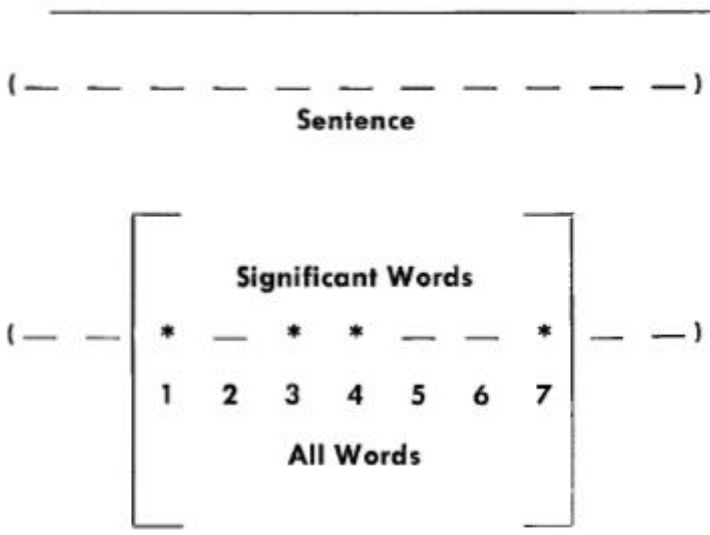
This rather unsophisticated argument on "significance" avoids such linguistic implications as grammar and syntax. In general, the method does not even propose to differentiate between word forms. Thus the variants differ, differentiate, different, differently,

difference and differential could ordinarily be considered identical notions and regarded as the same word. No attention is paid to the logical and semantic relationships the author has established. In other words, an inventory is taken and a word list compiled in descending order of frequency. So it is no information loss for the model even if you use steamer while pre-processing the text.

A word list compiled in accordance with the method outlined will generally take the form of the diagram in Fig. 1. The presence in the region of highest frequency of many of the words previously described as too common to have the type of significance being sought would constitute "noise" in the system. This noise can be materially reduced by an elimination technique in which text words are compared with a stored common-word list. A simpler way might be to determine a high-frequency cutoff through statistical methods to establish "confidence limits."

Establishing relative significance of sentences

As pointed out earlier, the method to be developed here is a probabilistic one based on the physical properties of written texts. No consideration is to be given to the meaning of words or the arguments expressed by word combinations. Instead it is here argued that, whatever the topic, the closer certain words are associated, the more specifically an aspect of the subject is being treated. Therefore, wherever the greatest number of frequently occurring different words are found in greatest physical proximity to each other, the probability is very high that he information being conveyed is most representative of the article.



Portion of sentence bracketed by and including significant words not more than four non-significant words apart. If eligible, the whole sentence is cited.

Figure 2 **Computation of significance factor.** *The square of the number of bracketed significant words (4) divided by the total number of bracketed words (7) = 2.3.*

A scheme for computing the significance factor is given by way of example in Fig. 2. It consists of ascertaining the extent of a cluster of words by bracketing, counting the number of significant words contained in the cluster, and dividing the square of this number by the total number of words within this cluster. The results based on this formula, as performed on abou5t 0 articles ranging from 300 to 4,500 words each, have been encouraging enough for further evaluation by a psychological experiment involving 100 people. This experiment will determine on an objective basis the effectiveness of the abstracts generated.

LATENT SYMANTIC ANALYSIS

Latent Semantic Analysis is an algebraic-statistical method that extracts hidden semantic structures of words and sentences. It is an unsupervised approach that does not need any training or external knowledge. LSA uses the context of the input document and extracts information such as which words are used together and which common words are seen in different sentences. A high number of common words among sentences indicates that the sentences are semantically related. The meaning of a sentence is decided using the words it contains, and meanings of words are decided using the sentences that contains the words. Singular Value Decomposition, an algebraic method, is used to find out the interrelations between sentences and words. Besides having the capability of modelling relationships among words and sentences, SVD has the capability of noise reduction, which helps to improve accuracy. In order to see how LSA can represent the meanings of words and sentences the following example is given.

Example 1: Three sentences are given as an input to LSA.

d0: ‘The man walked the dog’.

d1: ‘The man took the dog to the park’.

d2: ‘The dog went to the park’.

After performing the calculations we get the resulting figure, Figure 1. From Figure 1, we can see that d1 is more related to d2 than d0; and the word ‘walked’ is related to the word ‘man’ but not so much related to the word ‘park’. These kinds of analysis can be made by using LSA and input data, without any external knowledge. The summarization algorithms that are based on LSA method usually contain three main steps.

Input matrix creation: an input document needs to be represented in a way that enables a computer to understand and perform calculations on it. This representation is usually a matrix representation where columns are sentences and rows are words/phrases. The cells are used to represent the importance of words in sentences. Different approaches can be used for filling out the cell values. Since all words are not seen in all sentences, most of the time the created matrix is sparse.

The way in which an input matrix is created is very important for summarization, since it affects the resulting matrices calculated with SVD. As already mentioned, SVD is a complex algorithm and its complexity increases with the size of input matrix, which degrades the performance. In order to reduce the matrix size, rows of the matrix, i.e. the words, can be reduced by approaches like removing stop words, using the roots of words only, using phrases instead of words and so on. Also, cell values of matrix can change the results of SVD. There are different approaches to filling out the cell values. The approach used is Modified TF-IDF as discussed in the above section. Followed a SVD decomposition.

Sentence selection: Using the results of SVD, different algorithms are used to select important sentences. LSA has several limitations. The first one is that it does not use the information about word order, syntactic relations, and morphologies. This kind of information can be necessary for finding out the meaning of words and texts. The second limitation is that it uses no world knowledge, but just the information that exists in input document. The third limitation is related to the performance of the algorithm. With larger and more inhomogeneous data the performance decreases sharply. The decrease in performance is caused by SVD, which is a very complex algorithm. To handle these to some extent we use a Cross method for noise elimination and sentence selection. As described in the algorithms section above.

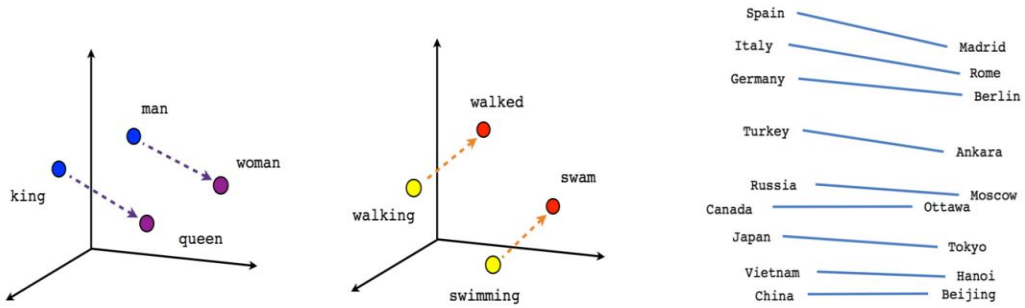
EMBEDDING METHOD

This is a hybrid cluster based method for text summarization where the we use the pre-trained embedding vectors for the creation a vectors for each and every sentence in the input text.

Input matrix creation:

Word Embeddings

A word embedding is a learned representation for text where words that have the same meaning have a similar representation. It is this approach to representing words and documents that may be considered one of the key breakthroughs of deep learning on challenging natural language processing problems.



Male-Female Verb tense Country-Capital

Word embeddings are in fact a class of techniques where individual words are represented as real-valued vectors in a predefined vector space. Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network, and hence the technique is often lumped into the field of deep learning. Key to the approach is the idea of using a dense distributed representation for each word.

Each word is represented by a real-valued vector, often tens or hundreds of dimensions. This is contrasted to the thousands or millions of dimensions required for sparse word representations, such as a one-hot encoding. These word embeddings are used to get the sentence embeddings, where the sentences with similar meaning or context get the similar representation. The input matrix is now ready and the next process is the sentence selection but before that we need to have a specific function to give the more suited abstract highest value according to the vector values obtained. We use topic clustering results to do this.

Sentence Selection:

Once the sentence vectors are done we pass them to the Topic Cluster method as described in the Algo Section above. The special dimension values are then used to get the metric/ measure required to select the sentences. The distance between the sentences position and the respective clusters center is calculated for all the sentences, which can be taken as that the sentence which is close to the topic is more related to the conversation and holds more valuable information needed for the abstract summery. We can further use the noise elimination similar to the one in cross method.

FEATURE BASED METHOD

One of the perspective towards text summarization, utilizes a combination of nine features to achieve feature scores of each sentence. Some of the features used to score the sentences are :

1. Word similarity among sentences
2. Word similarity among paragraphs
3. Iterative query score
4. Format based score
5. Numerical data score
6. Cue-phrases
7. Term weight
8. Thematic features
9. Title features.

The above approach for summarization gives better performance as compared to other summarization tools (MS - Word) as shown in Table I, II, II. Our aim is to compare the summary generated by this system to the summary generated by another system which uses only five features – four already used features such as Word similarity among sentences, Word similarity among paragraphs, Iterative query score, Cue-phrases and a new feature called Upper case feature. B. Implementation Input to the feature based text summarization system would consist of a well-structured source text document. The user may then provide the percent of summarization which he/she desires.

The implementation will be done in a series of steps as follows:

1) Preprocessing: Preprocessing involves analyzing the input text based on the following parameters:

a. Sentence count
b. Sentence segmentation
c. Word Stemming

2) Sentence scoring: All the sentences are scored based on various features such as its similarity to title, presence on numerical data in the sentence, its format (bold, italics), length of sentences, presence of certain phrases and its word similarity. Some of the features can be: a. Word similarity among sentences, b. Word similarity among paragraphs, c. Iterative query score, d. Cue-phrases, e. Upper Case.

Title feature: The number of title word in sentence, words in sentence that also occur in title gives high score [6]. This is determined by counting the number of matches between the content words in a sentence and the words in the title. We calculate the score for this

feature which is the ratio of the number of words in sentence that occur in the title over the number of word in title.

Score (Si) = No. Title word in Si /No. Word in Title

Sentence length: The number of word in sentence, this feature is useful to filtering out short sentences such as datelines and author names commonly found in news articles. The short sentences are not expected to belong to the summary [5]. We use normalized length of the sentence, which is the ratio of the number of words occurring in the sentence over the number of words occurring in the longest sentence of the document.

Score (Si) = No. Word occurring in Si / No. Word occurring in longest sentence

Term weight: Calculating the average of the TF-ISF (Term frequency, Inverse sentence frequency). The frequency of term occurrences within a document has often been used for calculating the importance of sentence [7].

Score (Si) = Sum of TF-ISF in Si / Max(Sum of TF-ISF)

Sentence position: Whether it is the first and last Sentence in the paragraph, sentence position in text gives the importance of the sentences. This feature can involve several items such as the position of a sentence in the document, section, paragraph, etc., [14] proposed first and last sentence highest ranking. The score for this feature: is maximum normalization with maximum value being the length of the sentences.

Score (Si) = 1 / position of the sentence Si

Sentence to sentence similarity: Similarity between sentences, for each sentence s, the similarity between s and each other sentence is computed by the cosine similarity measure. The score of this feature for a sentence s is obtained by computing the ratio of the summary of sentence similarity of sentence s with each other sentence over the maximum of summary.

Score (SJ = Sum of Sentence Similarity in Si / Max(Sum of Sentence Similarity)

Proper noun: The number of proper noun in sentence, sentence inclusion of name entity (proper noun). Usually the sentence that contains more proper nouns is an important one and it is most probably included in the document summary [17]. The score for this feature is calculated as the ratio of the number of proper nouns in sentence over the sentence length.

Score (Si) = No. Proper nouns in Si / Length (Si)

Thematic word: The number of thematic word in sentence, this feature is important because terms that occur frequently in a document are probably related to topic. The number of thematic words indicates the words with maximum possible relativity. We used the top 10 most frequent content word for consideration as thematic. The score for this feature is calculated as the ratio of the number of thematic words in sentence over the sentence length.

Score (Si) = No. Thematic word in Si / Length (Si)

Numerical data: The number of numerical data in sentence, sentence that contains numerical data is important and it is most probably included in the document summary. The score for this feature is calculated as the ratio of the number of numerical data in sentence over the sentence length.

Score (Si) = No. Numerical data in Si / Length (Si)

3) Fuzzy logic is then used to score the sentences. A fuzzy system consists of three phases –

a. Fuzzifier – Input to Fuzzifier consists of the feature scores of each sentence. This input is then converted from the numerical data into the linguistic values such as Very High, High, Medium, Low, and Very Low. This conversion is performed using the membership functions which describe how each feature score is converted into a membership value or degree of membership.

b. RuleBase & Inference Engine – Fuzzy rules are an important part of the fuzzy logic system. This phase comes after fuzzification of the candidate feature vectors and involves defining the fuzzy IF-THEN rules which form a major significant module in any fuzzy system. Fuzzy rules can be divided into two parts - Antecedent and consequent. Antecedent denotes the probable input feature values and consequent is the implication of the rule that decides whether the sentence is important, average or unimportant on the basis of the input.

c. Defuzzifier - The final desired output for each variable is generally a single number. The defuzzifier performs the task of converting the linguistic values obtained from the inference engine into crisp values. The main purpose of getting crisp value is to denote how close the sentence is to the given linguistic value. The output of Fuzzy logic system consists of crisp values assigned to every sentence. The selection of features plays an important role in determining the type of sentences that will be selected as part of the summary and, therefore, would influence the performance of the Fuzzy logic system.

4**) Sorting and selection:**

a. The sentences (S0, S1,...Sn-1) are then sorted in descending order of their crisp Fuzzy scores. These scores lie within the range 0 to 1.

b. Based on the percentage of summarization, a fixed number of the sorted sentences are selected.

c. After selecting, these sentences are displayed in their respective order of appearance in the original input text document.

DEPLOYMENT

RESULTS

The results of deep learning methods are not yet state-of-the-art compared to extractive methods, yet impressive results have been achieved on constrained problems such as generating headlines for news articles that rival or out-perform other abstractive methods. The promise of the approach is that the models can be trained end-to-end without specialized data preparation or sub models and that the models are entirely data-driven, without the preparation of specialized vocabulary or expertly pre-processed source documents.

CONCLUSION

REFERENCES

TextRank: Bringing Order into Texts Rada Mihalcea and Paul Tarau Department of Computer Science University of North Texas rada,tarau @cs.unt.ed .