**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW**

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 large documents of text. 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.

A summary is a text that is produced from one or more texts, that conveys important information in the original text, and it is of a shorter form. The goal of automatic text summarization is presenting the source text into a shorter version with semantics. The most important advantage of using a summary is, it reduces the reading time.

**1.2 MOTIVATION:**

Choosing to read an article mainly depends on the size of it and the time to be spent on reading it. If the article contains less critical information and contains large amount of textual data, people tend to skip it due to its less important information and large amount of time required to consume it.

Here, the articles with less critical information contain repetitive contents, which could be shrinken. But it requires deep understanding of the semantics present in the document in order to extract the most informative piece of data which can reduce the time that being spent on reading an article. So that many different information can be acknowledged on the time that being spent on reading a single article.

**1.3 PROBLEM STATEMENT:**

To produce an automatic text summarization by preserving the meaning and the key content of the original document. QUICK GLANCE generates a summary by extracting proper set of sentences from a document or multiple documents. It aims to produce an essence of the given documents.

* 1. **PROPOSED SYSTEM:**

The proposed system is the Quick Glance, a novel learning-free integer programming summarizer. It is a strong advantage that the approach is a training-free model. Therefore, neither labeled training data nor a large number of computational resources are required.

To achieve an efficient system, it is vital make the model understand natural language and select a few representative sentences from a given document. Therefore, the system is leveraged with publicly available DNN-based pre-trained sentence embedding vectors, which act as the aforementioned basic language understanding of humans.

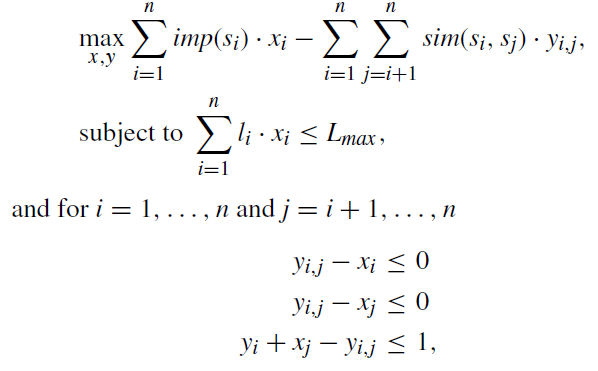
Employing pre-trained sentence vectors is of benefits to reducing resource consumption because they could be trained by either unsupervised methods or supervised classification tasks, which requires significantly fewer time and fortune to obtain labelled data than summarization task.

Next, in order to select representative sentences, ILP, which is a widely used method in industrial engineering, has been applied to extractive summarization. Along with this, PCA is used to automatically determine the most appropriate number of summary sentences. Hence, the proposed approach dynamically determines the number of summary sentences in terms of intrinsic information preservation.

# CHAPTER 2

# LITERATURE SURVEY

McDonald *et.al* [2] proposed the first ILP method for extractive summarization. It generates summaries by maximizing relevance (i.e., importance) of the selected sentences and minimizing there redundancy (i.e., similarity).



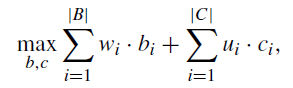
Where imp(si) is the importance score of sentence si, n is the number of sentences in the source document, li is the length of si, sim(si, sj) is the similarity between si and sj, and Lmax is the maximum length of summary sentences. xi is a binary variable indicating whether the sentence si is selected in the summary.

Yi;j denotes a binary variable indicating whether both si and sj are included in the summary. McDonald represented each sentence as a bag-of-words vector with TF-IDF values.

The importance scores are computed by using the positional information of the sentences and the similarity between each sentence vector and the document vector. The cosine similarity is used to compute the similarity between sentence vectors.

Berg-Kirkpatrick [3]constructed an ILP summarization model based on the notion of *concept*, which is actually a set of bi-grams. The distinctive characteristic of this model is that it extracts and compresses sentences simultaneously.

The model not only selects bi-grams with high importance but also chooses whether to cut (delete) individual sub trees from each sentence's parsing tree. The objective function of this model is the following:



Where *bi* and *ci* are binary variables that indicate the selection of the *ith* bi-gram as a summary and its deletion from the parsing tree. *w*iand *u*iindicate the weights of bi-grams and possible subtree cuts, respectively. Additionally, the model has a constraint of maximum allowed summary length, which is determined by the user. The weights are estimated by soft margin support vector machine optimization with bi-gram recall loss function. Therefore, the model is trained in a supervised manner, which requires gold-standard summaries.

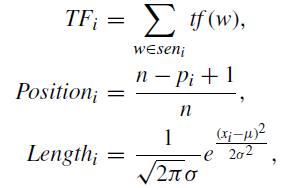
Galanis [4] also presented a supervised extractive summarization model that extracts sentences and concepts by maximizing sentence importance and diversity (i.e., minimizing redundancy). To represent sentences in a structured form, they leveraged various features, such as sentence position, named entities, word overlap, content word frequency, and document frequency. The model has a constraint of user-defined maximum summary length. Furthermore, support vector regression (SVR) was used to estimate sentence importance.

Boudin[5]proposed a purely concept-based extractive summarization model. The objective function of this model is the following:

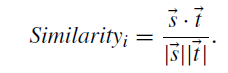


where *w*iis the weight of a concept, *f*kis the frequency of the non-stop word *k* in the document set, *ci* is a binary variable indicating whether the concept *i* is selected, and *tk* is a binary variable indicating the presence of the non-stop word *k* in the summary. This variable was introduced because the frequency of a non-stop word is a good predictor of a word appearance in a human-generated summary. To obtain the concept weight, heuristic counting (such as the document frequency of each concept) was used, or the supervised model was trained. The model also has a user-determined maximum summary length. It differs from others by applied sentence pruning. Sentences with fewer than 10 words were removed to improve computational efficiency.

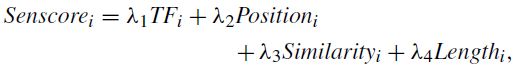
Liu[6] proposed a simple weighting-based unsupervised extractive summarization method. For the *ith* sentence in a document (*Si*), they calculated sentence scores by leveraging term frequency, similarity, positional information, and sentence length. In particular, term frequency, position score, and sentence length score are calculated as follows:



Where *tf* (*w*) denotes the term frequency of the word *w*, *n* represents the total number of sentences, *pi* refers to the position of *Si*, *xi* is the length of *Si*, and µ and σ denote the mean and standard deviation of the total sentence length, respectively. For a given sentence vector E*s* and title vector E*t*, similarity score is defined as the cosine similarity of two vectors:



The vectors of each sentence and article title are built based on the term frequency weighting schema. Therefore, the final sentence score is calculated as follows:



Where λ1, λ2, λ3, and λ4 are user-defined hyper-parameters. Liu *et al.*  Generated a summary by selecting sentences with the highest score until the total summary did not exceed the user-defined summary length.

|  |  |  |
| --- | --- | --- |
| **REFERENCE NO.** | **TECHNIQUES** | **REMARKS** |
| [2] | Integer Linear Programming | * First ILP model * Maximizing relevance and minimizing redundancy |
| [3] | Integer Linear Programming | * User-defined maximum summary length * Selects bi-grams with high importance |
| [4] | Support Vector Regression | * Sentence position * Sentence importance |
| [5] | Concept-based model | * Sentence pruning * Computational efficiency |
| [6] | Weighting-based model | * Similarity, position score and sentence length |

**CHAPTER 3**

**PROPOSED METHOD**

Sentences

Sentence vector

Sequential information

Document

Principal sentences

Summary

1. Pre-processing 2. Pre-trained Sentence model

3. Positional Encoding 4. PCA 5. ILP

**Figure 3.1 Block Diagram**

The QUICK GLANCE model consists of a two-step procedure: document representation and representative sentence selection. To represent a document as a continuous vector, we use distributed vectors pre-trained by deep learning-based sentence embedding models. Next, ILP and PCA are used to evaluate the sentence importance score and select the representative sentences for the summary.

**3.1 DEEP REPRESENTATION OF A DOCUMENT**

**3.1.1 PRE-PROCESSING**

The input document undergoes a pre-processing step which involves tokenizing the white space, a lower-case conversion, removing the punctuations, numbers and empty strings from each token.

**3.1.2 PRE-TRAINED SENTENCE MODEL**

Since, the document consists of n sentences having the sequential information, representing the document as a sequence of sentences as follows:

**D = [s1,s2, . . . ,sn]**

where

D denotes the document and

sk refers to its k-th sentence.

Let a sentence be represented as a column vector.

Thus, Dbasic, the basic representation of D, becomes the following

matrix:

**Dbasic = [sv1,sv2, . . . ,svn]**

Where svk denotes the pre-trained sentence vector of sk . Dbasic is a d × n matrix, where d is the embedding dimension. It is possible to use any sort of sentence embedding method to generate svk. Here, the pre-trained sentence embedding model BERT (Bidirectional Encoding Representations from Transformers) [8] is used, since it has a good memory to know the positional sequence. It has pre-defined vectors for each word in a sentence.

It also has the vector for sub-word if the word doesn’t exist. BERT generates contextualized represents of sentence as a T × dh matrix, where T is the number of tokens and dh is the dimension of the hidden states. As the representation of a sentence should be a vector, the average of the BERT representations is considered to obtain the dimension dh.

**3.1.3 POSITIONAL ENCODING**

Although the matrix Dbasic contains the intrinsic meaning of each sentence, it lacks positional information, which plays a critical role in natural language tasks.

Therefore, positional encoding is used to effectively reflect sequential information, which employs cosine and sine functions. In particular, the positional encoding matrix PE is calculated as follows:

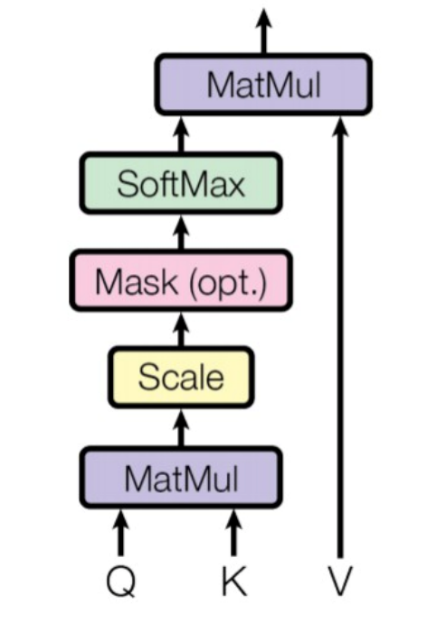
PE(pos,2i) = sin(pos/100002i/d )

PE(pos,2i+1) = cos(pos/100002i/d )

where pos is the position, and i is the dimension. Then, the final input embedding matrix Demb is calculated as the addition of Dbasic and PE.

Demb = Dbasic + PE

**3.1.4 SCALED DOT-PRODUCT ATTENTION**



**Figure 3.2 Attention Mechanism**

To obtain a deeper representation of the document scaled dot product attention, wherein the attention weights can be calculated without training parameters.

The matrix **Q** is regarded as a set of queries. Likewise, the matrices **K** and **V** are the sets of keys and values, respectively. Then, the result of the scaled dot product attention is as follows:

Attention (Q, K, V) = softmax( Q .KT / √ dk )V

Where dk is the dimension of queries and keys. Since, the self-attention method, the matrices Q, K, and V are Demb. Therefore, a deep representation Ddeep of the document is calculated as

Ddeep = Attention (Demb, Demb, Demb)

Where Ddeep has dimension d ×n, the same dimension as that of Demb. Therefore, the matrix Ddeep can be considered the sequence of the deep representation vectors of the sentences constituting the document.

Ddeep = [svdeep,1,svdeep,2, . . . ,svdeep,n].

**3.2 PRINCIPAL COMPONENT ANALYSIS:**

The proposed QUICK GLANCE automatically determines the number of appropriate summary sentences for each document by PCA, and thus the importance of each sentence can be quantitatively evaluated. PCA is a method for reducing high-dimensional data to lower dimensions. The principal components (PCs) extracted by PCA are composed of a linear combination of the original variables.

Hence, PCA has been widely used for dimensionality reduction because it is possible to explain the entire dataset through a few PCs. The purpose of PCA is to maximize the variance of y = Xw, which is the projection of the original data X, where X is an n × p matrix and w is a vector of size p × 1. n is the number of observations and p is the number of variables.

For a centered dataset X, PCA is performed by the following optimization:

Max Var(Y) = w T ∑w

s.t ||w|| = 1

where ∑ is the covariance matrix of X. Solving the above equation for w yields the eigenvectors e of the covariance matrix. Therefore, when the i-th largest eigenvalue λi and its corresponding eigenvector ei are given, the i-th PC is calculated as follows:

yi = ei1X1 + ei2X2 + . . . + eipXp

Furthermore, by the following equation, the variance of yi is λi .

Var(yi) = e T i ∑ei = λi

As the total population variance is ∑p i=1 λi , the variance preservation ratio of yi is

αi = λi / ∑p i=1 λi

where αi is the variance preservation ratio of Yi . This implies that a PC yi preserves (100 × αi)% of the total variance. As Ddeep has sentence vectors as column vectors, each sentence vector is considered as a variable and the dimension of the vector, d, as the number of observations.

Therefore, PCA reduces the number of sentences in a document by extracting the PCs. Each PC (i.e.,) principal sentence, could be considered a vector of condensed sentences that contains as much information of the original document as possible.

Subsequently, a user-defined specific hyper parameter β - the variance preservation ratio that automatically decide the appropriate number of summary sentences N.

Therefore, it could be considered that at least β% of the total information is preserved in N selected sentences. Then the sentence importance score is calculated based on the correlation between the sentence vectors of Ddeep and the PS’s.

As a PS is a high-level vector that efficiently condenses the intrinsic information of the document, greater similarity of a sentence vector to a PS implies higher importance of the sentence. Therefore, sentence importance score is defined by:

imp (si) = ∑ k≠i cos(svdeep,i, PSk )

**3.3 INTEGER LINEAR PROGRAMMING**

**3.3.1 OPTIMIZATION**

The optimization problem is formulated as in McDonald [1], because the minimum extraction unit is a sentence, not a concept. The requirements for the formulation are sentence importance scores and similarity scores between sentences.

The similarity between the sentences si and sj is defined as the cosine similarity of their deep sentence representation vectors as follows:

sim(si,sj) = cos(svdeep,i,svdeep,j)

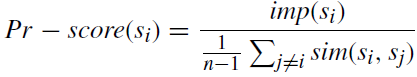
Then, the appropriate number of summary sentences (the length of summary sentences) for each document is automatically determined through PCA as follows:

∑n i=1 xi = N

Where N is the appropriate number of summary sentences.

**3.3.2 SENTENCE PRUNING**

In order to reduce the time complexity of QUICK-GLANCE from O(n2 ), sentence pruning is performed to remove unimportant sentences. The sentences to be extracted are those which should have high sentence importance scores and low redundancy scores. Therefore, the pruning score of a sentence si is defined by



Where imp(si) is the sentence importance score of sentence si, sim(si,sj) is the similarity between the sentences si,sj.

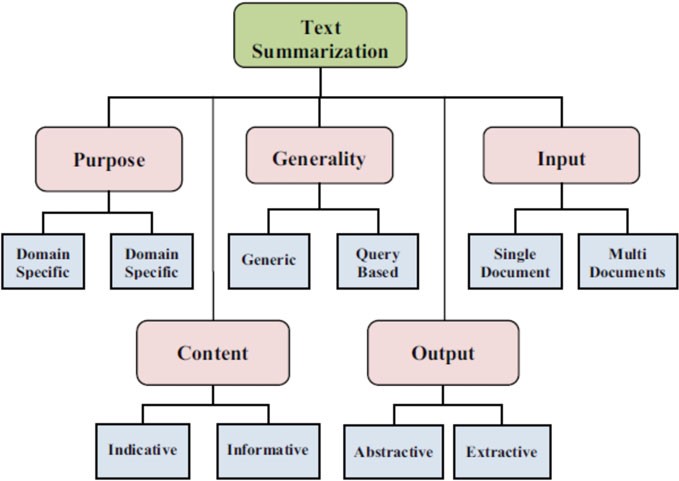
**3.3.3 EVALUATING THE SUMMARIZATION SYSTEMS**

Summary evaluation is a very important aspect for text summarization. Generally, summaries can be evaluated using intrinsic or extrinsic measures. While intrinsic methods attempt to measure summary quality using human evaluation and extrinsic methods measure.

The same through a task-based performance measure such the information retrieval-oriented task. Evaluation methods are useful in evaluating the usefulness and trustfulness of the summary.

Evaluating the qualities like comprehensibility, coherence, and readability is really difficult. System evaluation might be performed manually (gold standard) by experts to measure the quality of summary, the manually expert system is used. The qualitative evaluation is done by counting the numbers of sentences selected by the system that match with the human gold standard. To measure the quantitative assessment of the summary the ROUGE evaluator tool is used which consist of precision, recall and F-measure.

**3.4 TEXT SUMMARIZATION TECHNIQUES**

****

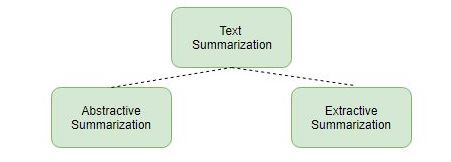
**Figure 3.4 Text Summarization Techniques**

**3.5 TEXT SUMMARIZATION IN NLP**

Let’s first understand what text summarization is before we look at how it works. Here is a succinct definition to get us started: “Automatic text summarization is the task of producing a concise and fluent summary while preserving key information content and overall meaning”

There are broadly two different approaches that are used for text summarization:

* + - * **Extractive Summarization**
      * **Abstractive Summarization**

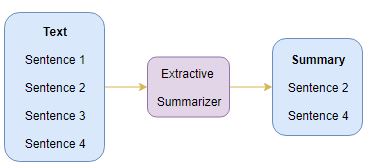


**Figure 3.5 Text Summarization in NLP**

**3.5.1 EXTRACTIVE SUMMARIZATION**

In Extractive Summarization, we are identifying important phrases or sentences from the original text and extract only these phrases from the text. These extracted sentences would be the summary

**We identify the important sentences or phrases from the original text and extract only those from the text.** Those extracted sentences would be our summary. The below diagram illustrates extractive summarization:

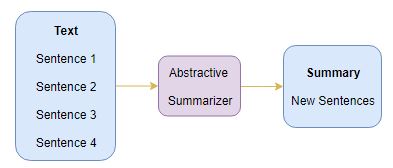


**Figure** **3.5.1 Extractive Summarization**

**3.5.2 ABSTRACTIVE SUMMARIZATION**

This is a very interesting approach. Here, we generate new sentences from the original text. This is in contrast to the extractive approach we saw earlier where we used only the sentences those were present. The sentences generated through abstractive summarization might not be present in the original text:

In the Abstractive Summarization approach, we work on generating new sentences from the original text. The abstractive method is in contrast to the approach that was described above. The sentences generated through this approach might not even be present in the original text.



## Figure 3.5.2 Abstractive Summarization

## 3.6 INTRODUCTION TO SEQUENCE-TO-SEQUENCE (SEQ2SEQ) MODELING

We can build a Seq2Seq model on any problem which involves sequential information. This includes Sentiment classification, Neural Machine Translation, and Named Entity Recognition – some very common applications of sequential information.

In the case of Neural Machine Translation, the input is a text in one language and the output is also a text in another language:

nmt

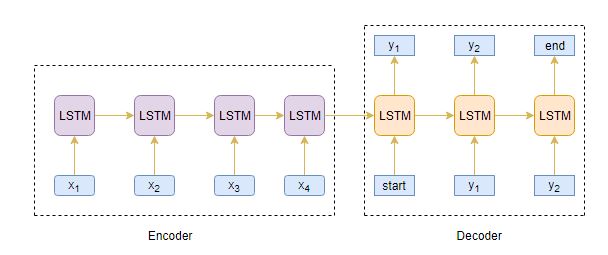
In the Named Entity Recognition, the input is a sequence of words and the output is a sequence of tags for every word in the input sequence:

ner

Our objective is to build a text summarizer where the input is a long sequence of words (in a text body), and the output is a short summary (which is a sequence as well).

So, **we can model this as a Many-to-Many Seq2Seq problem.**

Below is a typical Seq2Seq model architecture:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/05/final.jpg)

## Figure 3.6 introduction to sequence-to-sequence (seq2seq) modeling

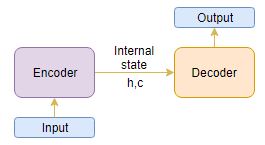
There are two major components of a Seq2Seq model:

* **Encoder**
* **Decoder**

## 3.7 UNDERSTANDING THE ENCODER-DECODER ARCHITECTURE

The Encoder-Decoder architecture is mainly used to solve the sequence-to-sequence (Seq2Seq) problems where the input and output sequences are of different lengths.

Let’s understand this from the perspective of text summarization. The input is a long sequence of words and the output will be a short version of the input sequence.



## Figure 3.7 understanding the encoder-decoder architecture

Generally, variants of Recurrent Neural Networks (RNNs), i.e. Gated Recurrent Neural Network (GRU) or Long Short Term Memory (LSTM), are preferred as the encoder and decoder components. This is because they are capable of capturing long term dependencies by overcoming the problem of vanishing gradient.

We can set up the Encoder-Decoder in 2 phases:

* **Training phase**
* **Inference phase**

Let’s understand these concepts through the lens of an LSTM model.

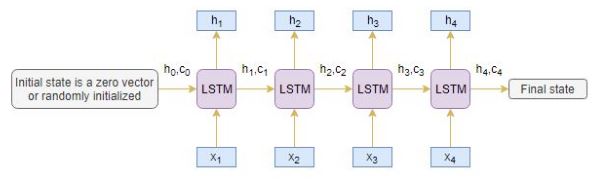
## 3.7.1 TRAINING PHASE

In the training phase, we will first set up the encoder and decoder. We will then train the model to predict the target sequence offset by one timestep. Let us see in detail on how to set up the encoder and decoder.

## 3.7.2 ENCODER

An Encoder Long Short Term Memory model (LSTM) reads the entire input sequence wherein, at each timestep, one word is fed into the encoder. It then processes the information at every timestep and captures the contextual information present in the input sequence.

I’ve put together the below diagram which illustrates this process:

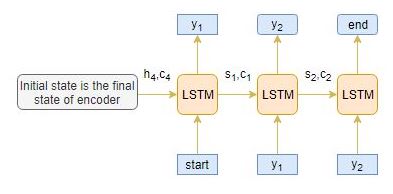
[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/05/61.jpg)

## Figure 3.7.2 Encoder

The hidden state (hi) and cell state (ci) of the last time step are used to initialize the decoder. Remember, this is because the encoder and decoder are two different sets of the LSTM architecture.

**3.7.3 DECODER**

The decoder is also an LSTM network which reads the entire target sequence word-by-word and predicts the same sequence offset by one timestep. **The decoder is trained to predict the next word in the sequence given the previous word.**

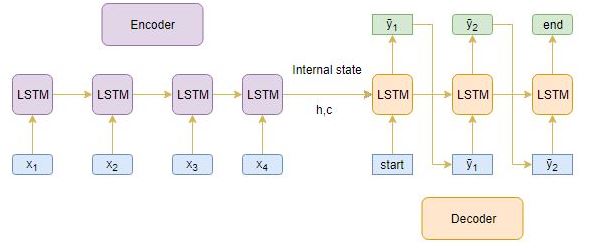
[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/05/71.jpg)

**Figure 3.7.3 Decoder**

**<Start>** and <**end>**are the special tokens which are added to the target sequence before feeding it into the decoder. The target sequence is unknown while decoding the test sequence. So, we start predicting the target sequence by passing the first word into the decoder which would be always the <**start>**token. And the <**end>**token signals the end of the sentence.

**3.7.4 INFERENCE PHASE**

After training, the model is tested on new source sequences for which the target sequence is unknown. So, we need to set up the inference architecture to decode a test sequence:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/05/82.jpg)

**Figure 3.7.4 Inference phase**

**CHAPTER 4**

**IMPLEMENTATION**

Implementation

SUMMARY

GUI INPUT

Figure 4 Implementation in GUI

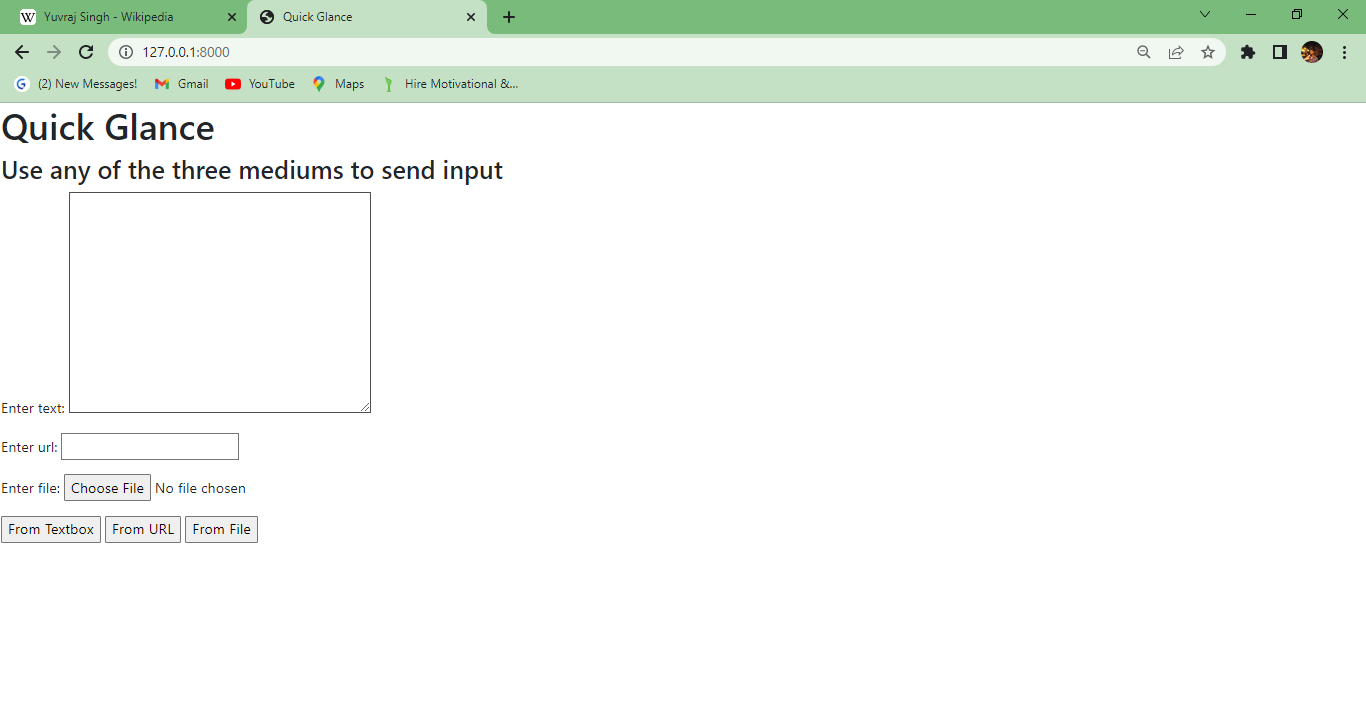
**4.1 GUI INTEGRATION:**

The project is given a Graphical User Interface on the Web using Django Web Framework. The textual data can be given in 3 different ways.

* Direct text using Textbox
* Parsed text from any given URL
* Textual data from a .txt file

The data from the textbox and .txt file can be retrieved using simple forms and read () method from Django. Whereas the parsed text from the URL has to be retrieved using Web scraping through Beautiful Soup.

**GUI Homepage**



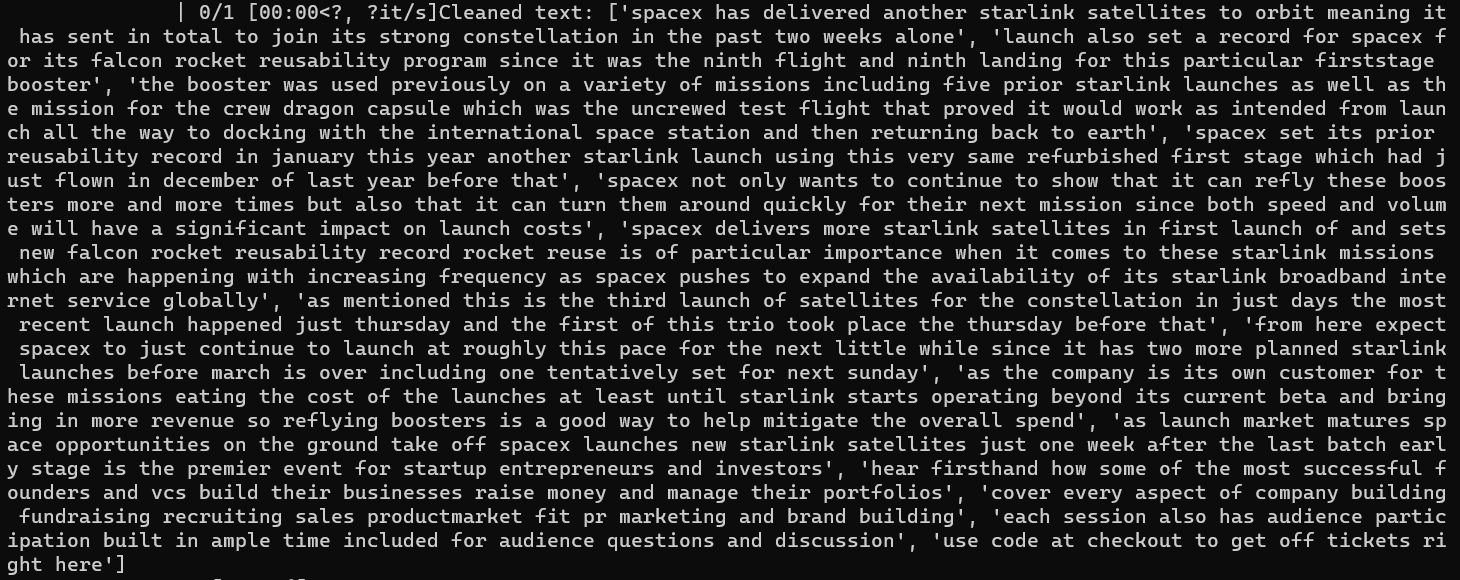
**Figure 4.1 GUI Homepage**

**4.2 IMPLEMENTATION**

**4.2.1 PARAGRAPH TO SENTENCES AND CLEANING:**

The given text will be paragraphs by default, so it has to be converted into sentences for text extraction. NLTK is used for converting the given text into sentences. Using a split() function is a good idea, but with a huge amount of text which may include several quotes and special characters, it’s will not always provide a good result. So, nltk.load() has been used.

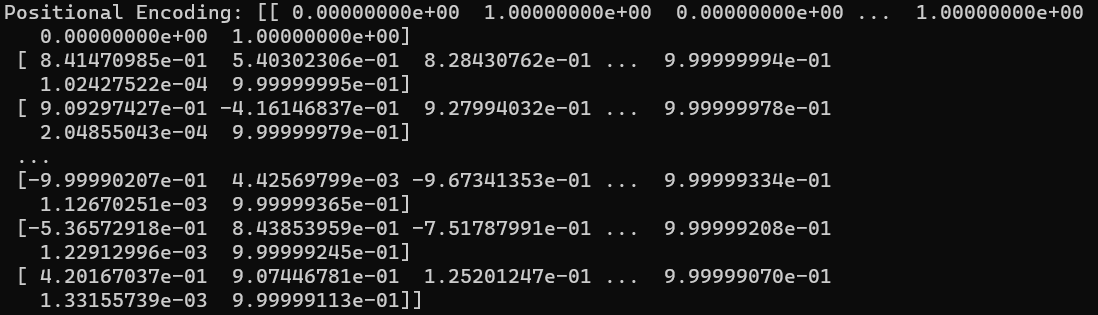
After the text has been converted to sentences, it has to be cleaned. It includes removing the punctuation, removing the needless spaces, converting all text to lower case, removing the number tokens. The processed text will be forwarded to further steps.



**Figure 4.2.1 Cleaned Text**

**4.2.2 POSITIONAL ENCODING:**

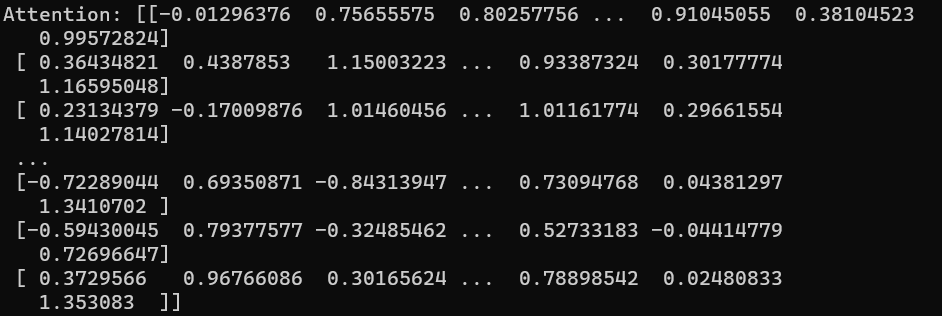
Positional encoding is used to effectively reflect sequential information. The positional encoding method employs cosine and sine functions to calculate the position of the word in the sentence



**Figure 4.2.2 Positional encoding**

**4.2.3 ATTENTION**

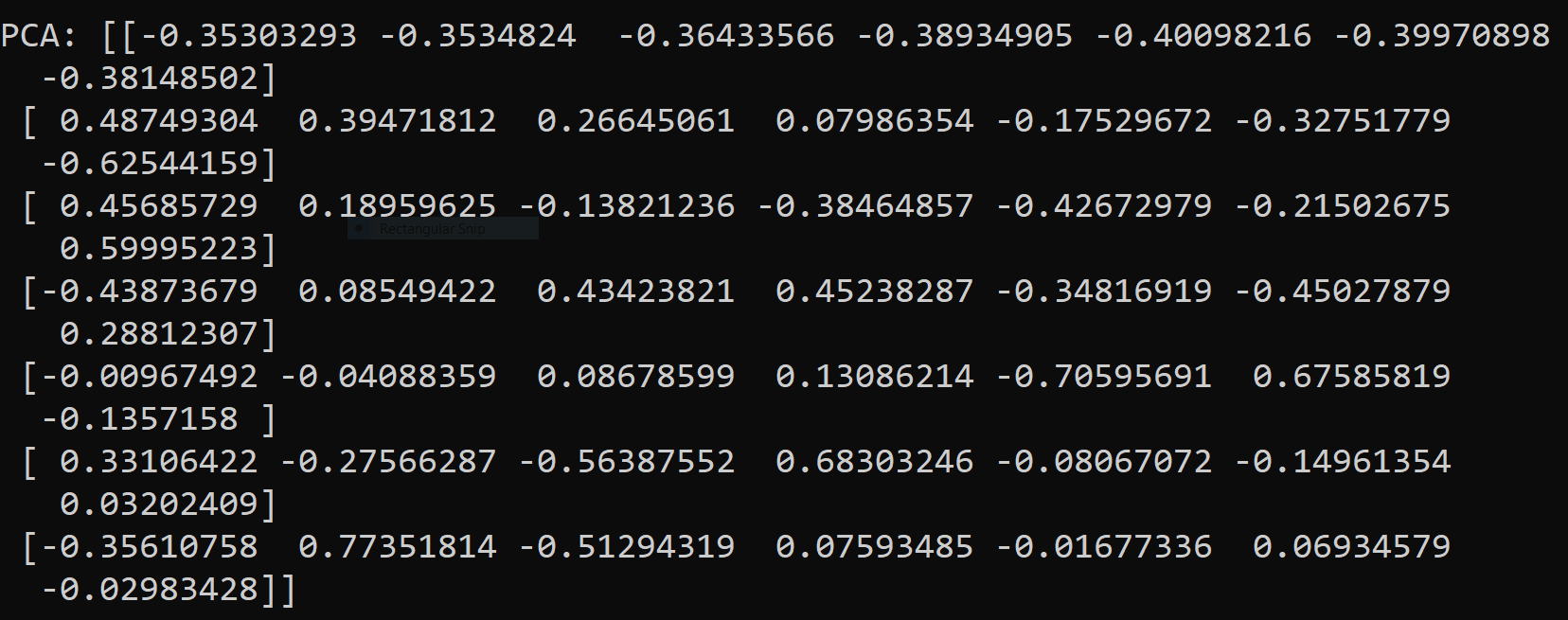
Attention has the ability to attend a semantically important concept or region of interest and also to find the relative strength of attention paid on multiple concepts. It is able to switch attention among concepts dynamically according to task status.



**Figure 4.2.3 Attention Scores**

**4.2.4 PCA**

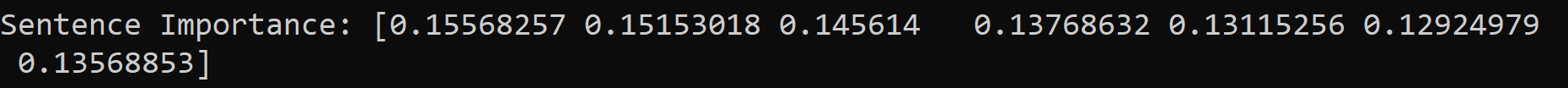
Method used for dimensionality reduction because it is possible to explain the entire dataset through a few PC’s (Principal Components).

****

**Figure 4.2.4 Principal components**

**4.2.5 ILP**

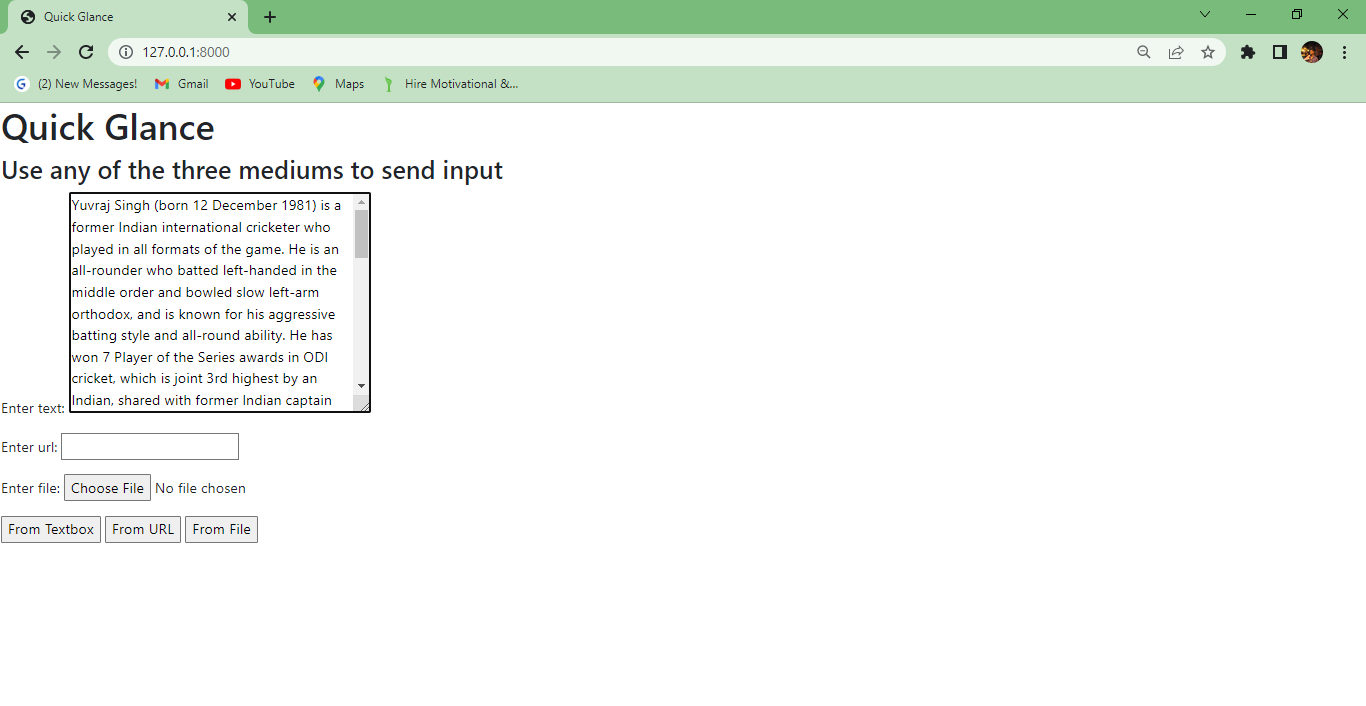
In ILP, sentence scores and similarity scores between sentences are calculated whereas sentence pruning helps to reduce unimportant sentences and time complexity.

****

**Figure 4.2.5 Sentence importance**

**4.3 SUMMARY**

The summary of the given document is achieved and is displayed in the GUI’s homepage



**Figure 4.3 Input before Textbox summarization**

**4.3.1 OUTPUT AFTER TEXT SUMMARIZATION**

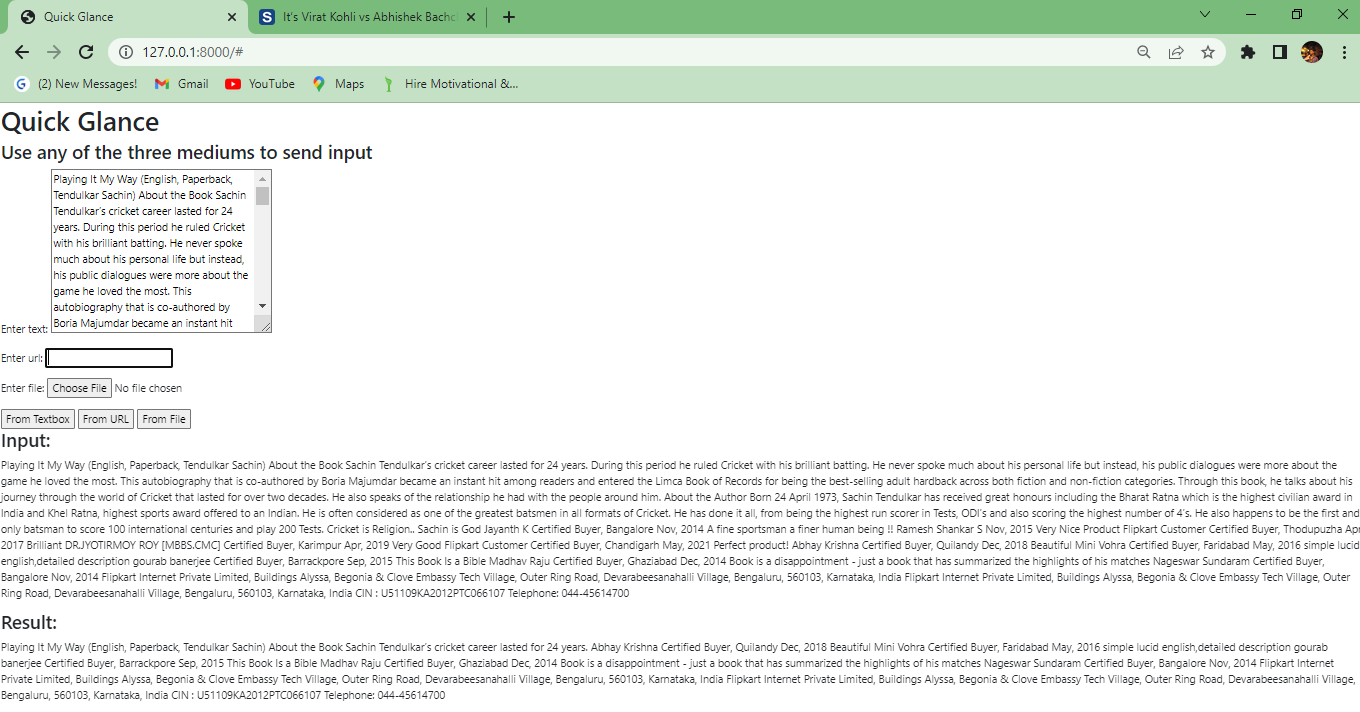
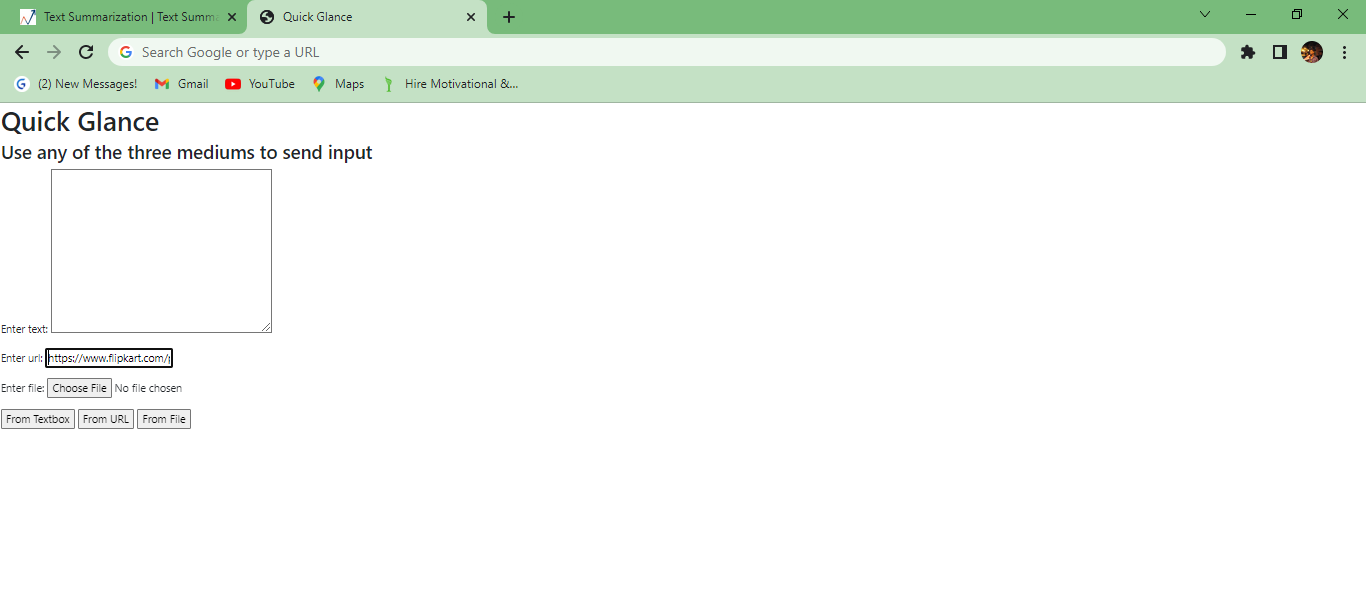
****

Figure 4.3.1 **Output after Text Summarization**

4.3.2 **INPUT BEFORE URL SUMMARIZATION**

**** Figure 4.3.2 **Input Before URL Summarization**

4.3.3 **OUTPUT AFTER URL SUMMARIZATION**

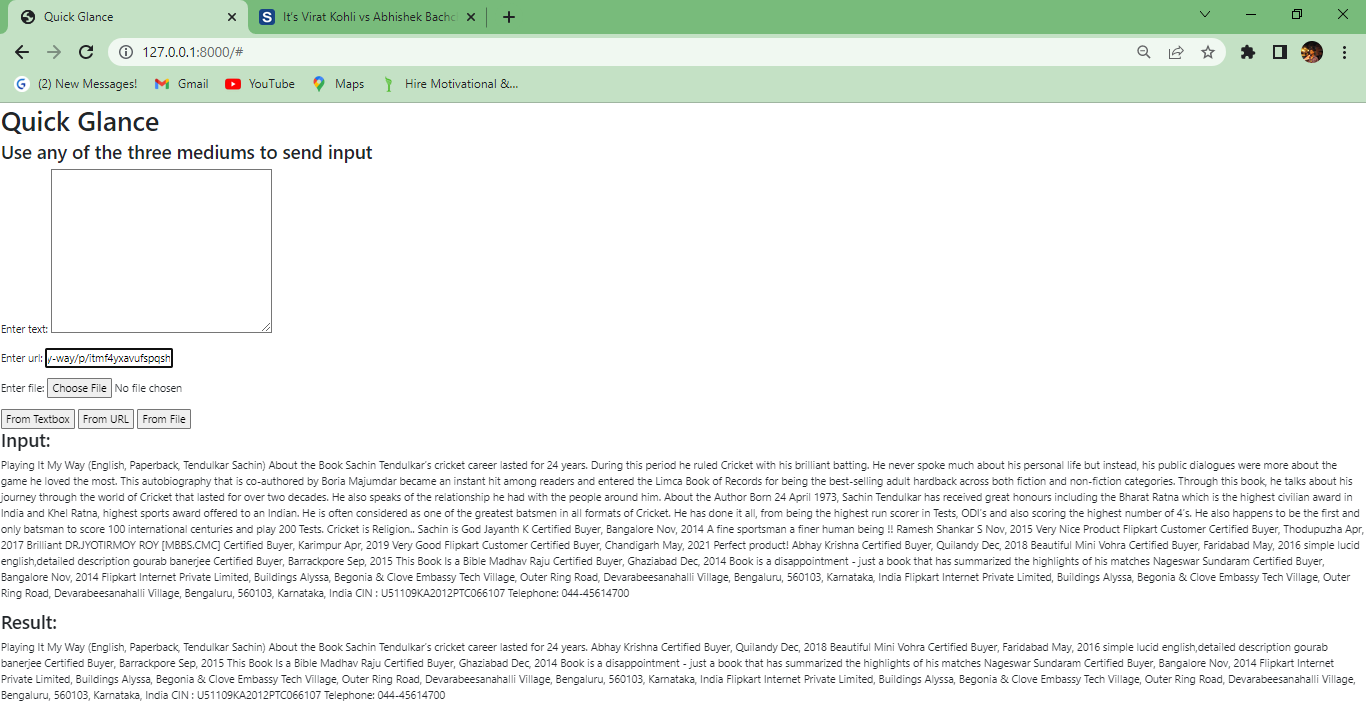
****

Figure 4.3.3 **Output After URL Summarization**

4.3.4 **INPUT BEFORE FILE SUMMARIZATION**

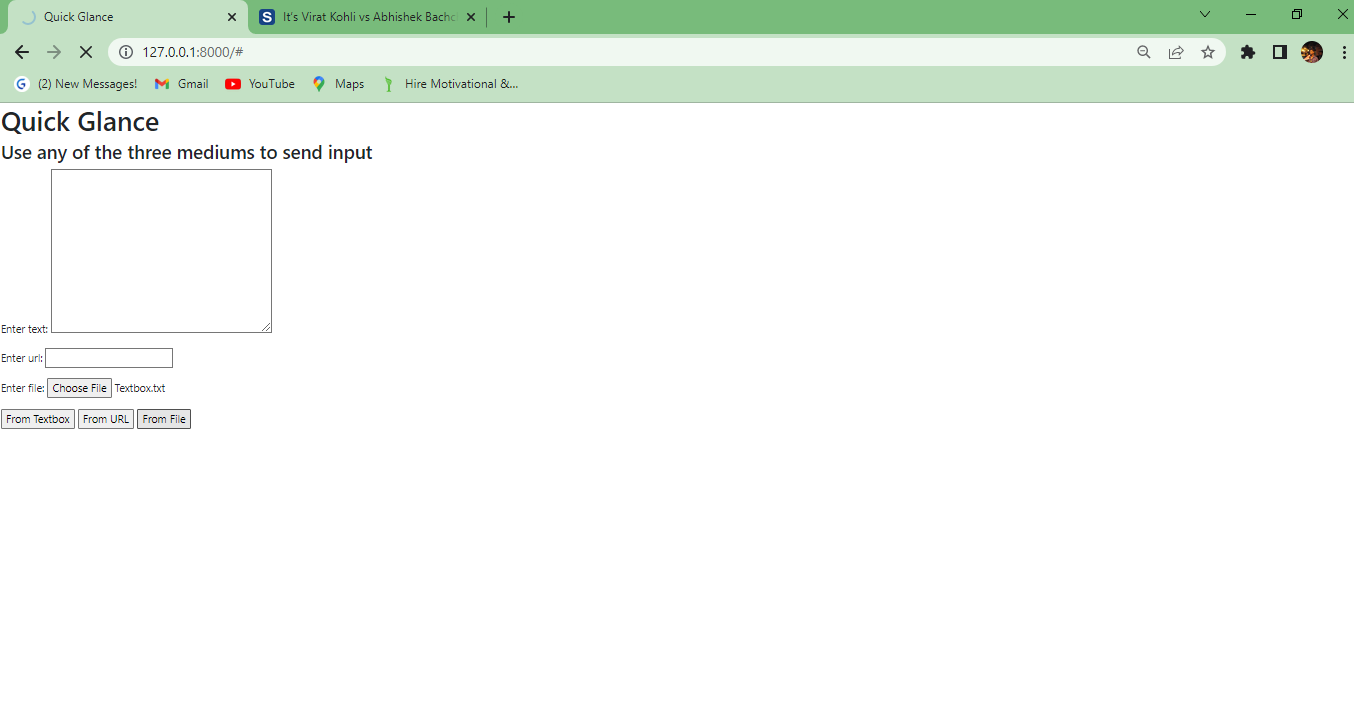
****

Figure 4.3.4 **Input Before File Summarization**

4.3.5 **OUTPUT AFTER FILE SUMMARIZATION**

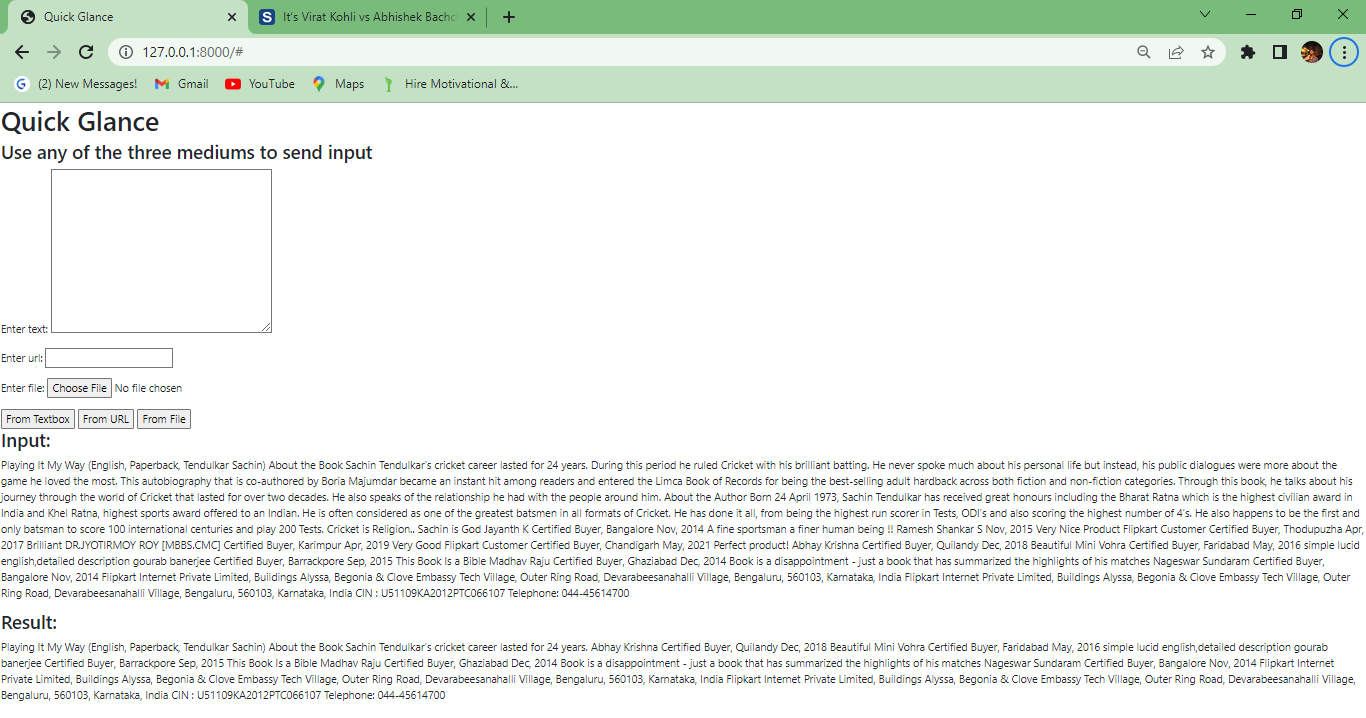
****

Figure4.3.5 **Output After File Summarization**

**CHAPTER 5**

**CONCLUSION AND FUTURE WORK**

**5.1 CONCLUSION**

A user-friendly text summarizer, the QUICK GLANCE is proposed efficiently by retaining the intrinsic information. Experimental results revised that the Quick Glance provides well organized summary with needed information for non-critical applications like review, news glance etc., in a user-friendly manner.

**5.2 FUTURE WORK**

Here, the appropriate number of sentences in the summary will be decided by PCA using a variance preservation ratio β. In future, PCA can be replaced with Linear Discriminant Analysis (LDA), and non-negative matrix factorization (NMF). As of now, the summarized output is displayed in the GUI’s Homepage. Future works can include that the output can also be sent to the users valid mail. It can be also implemented as a mobile application which will help out the students for quick grasping of the context.

**CHAPTER 6**

**REFERENCES**

[1] M. Jang and P. Kang, "Learning-Free Unsupervised Extractive Summarization Model," in IEEE Access, vol. 9, pp. 14358-14368, 2021, doi: 10.1109/ACCESS.2021.3051237.

[2] McDonald, ``A study of global inference algorithms in multi-document summarization,'' in Proc. Eur. Conf. Inf. Retr. Berlin, Germany: Springer, 2007, pp. 557564.

[3] T. Berg-Kirkpatrick, D. Gillick, and D. Klein, ``Jointly learning to extract and compress,'' in Proc. 49th Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol., vol. 1. Stroudsburg, PA, USA: Association Computational Linguistics, 2011, pp. 481490.

[4] D.Galanis, G. Lampouras, and I. Androutsopoulos, ``Extractive multidocument summarization with integer linear programming and support vector regression,'' in *Proc. COLING*, 2012, pp. 911\_926.

[5] F. Boudin, H. Mougard, and B. Favre, ``Concept-based summarization using integer linear programming: From concept pruning to multiple optimal solutions,'' in *Proc. Conf. Empirical Methods Natural Lang. Process.*,2015,

[6] M. Liu, L. Wang, and L. Nie, ``Weibo-oriented Chinese news summarization via multi-feature combination,'' in *Natural Language Processing and Chinese Computing*. Cham, Switzerland: Springer, 2015.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.

N. Gomez,L. Kaiser, and I. Polosukhin, ``Attention is all you need,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998\_6008