Natural Language Processing   
CSE4022 RIVIEW III

NAME: AMIT KUMAR SLOT: G2

REG : 19BCE0424 DATE:27/05/2021

Information extraction Using Text Rank Algorithm and Hidden Markov Model

GOOGLE COLAB NOTEBOOK LINK:

<https://colab.research.google.com/drive/1sLohhHnFyjV1586RKzkPfOQz5mJgn9ml?usp=sharing>

1. **ABSTRACT:**

**Information extraction** (**IE**) is the task of automatically extracting structured information from unstructured and/or semi-structured machine readable documents and other electronically represented sources. In most of the cases this activity concerns processing human language texts by means of natural lamguage processing (NLP). Recent activities in multimedia document processing like automatic annotation and content extraction out of images /audio /video /documents could be seen as information extraction.

So in this Project By using concept of natural language processing and by the use of Text Rank Algorithm and Hidden Markov Model, Information Extraction will be done.

1. **Introduction:**

**Information extraction is the process of extracting information from unstructured textual sources to enable finding entities as well as classifying and storing them in a database. Semantically enhanced information extraction (also known as semantic annotation) couples those entities with their semantic descriptions and connections from a knowledge graph. By adding metadata to the extracted concepts, this technology solves many challenges in enterprise content management and knowledge discovery.**

**In this world where everyday millions tirillions petabyte of data has been operated extracting information is a very important task so further in this project motive is to extract information and convert into structured manner from unstructured data.**

Text data contains a lot of information but not all of it will be important to you. We might be looking for names of entities, others would want to extract specific relationships between those entities. Our intentions differ according to our requirements.

Imagine having to go through all the legal documents to find legal precedence to validate your current case. Or having to go through all the research papers to find relevant information to cure a disease. There are many more examples like resume harvesting, media analysis, email scanning, etc.

But just imagine having to manually go through all of the textual data and extracting the most relevant information. Clearly, it is an uphill battle and you might even end up skipping some important information.

For anyone trying to analyze textual data, the difficult task is not of finding the right documents, but of finding the right information from these documents. Understanding the relationship between entities, understanding how the events have unfolded, or just simply finding hidden gems of information, is clearly what anyone is looking for when they go through a piece of text.

Therefore, coming up with an automated way of extracting the information from textual data and presenting it in a structured manner will help us reap a lot of benefits and tremendously reduce the amount of time we have to spend time skimming through text documents. This is precisely what information extraction strives to achieve.

Using information extraction, we can retrieve pre-defined information such as the name of a person, location of an organization, or identify a relation between entities, and save this information in a structured format such as a database.

This enables us to reap the benefits of powerful query tools like SQL for further analysis. Creating such structured data using information extraction will not only help us in analyzing the documents better but also help us in understanding the hidden relationships in the text.

**3.Problem Statement**

**How does Information Extraction work?**

Given the capricious nature of text data that changes depending on the author or the context, Information Extraction seems like a daunting task. But it doesn’t have to be that way!

We all know that sentences are made up of words belonging to different [**Parts of Speech (POS)**](https://medium.com/analytics-vidhya/part-of-speech-tagging-what-when-why-and-how-9d250e634df6). There are eight different **POS** in the English language: **noun, pronoun, verb, adjective, adverb, preposition, conjunction,** and **intersection**.

The POS determines how a specific word functions in meaning in a given sentence. For example, take the word “right”. In the sentence, “The boy was awarded chocolate for giving the right answer”, “right” is used as an adjective. Whereas, in the sentence, “You have the right to say whatever you want”, “right” is treated as a noun.

This goes to show that the POS tag of a word carries a lot of significance when it comes to understanding the meaning of a sentence. And we can leverage it to extract meaningful information from our text.

We were easily able to determine the POS tags of all the words in the sentence. But how does it help in Information Extraction?

Well, if we wanted to extract nouns from the sentences, we could take a look at POS tags of the words/tokens in the sentence, using the attribute **.pos\_**, and extract them accordingly.

Each word is a **node** in the Dependency graph. The relationship between words is denoted by the edges. For example, “The” is a determiner here, “children” is the subject of the sentence, “biscuits” is the object of the sentence, and “cream” is a compound word that gives us more information about the object.

The arrows carry a lot of significance here:

* The **arrowhead** points to the words that are **dependent** on the word pointed by the **origin of the arrow**
* The former is referred to as the**child node** of the latter. For example, “children” is the child node of “love”
* The word which has no incoming arrow is called the **root node** of the sentence

**Text Pre-Processing**

First, we need to clean our text data. When I went over a few speeches, I found each paragraph in the speech was numbered to distinctly identify it. There were obviously unwanted characters like newline character, a hyphen, salutations, and apostrophes, like in any other text dataset.

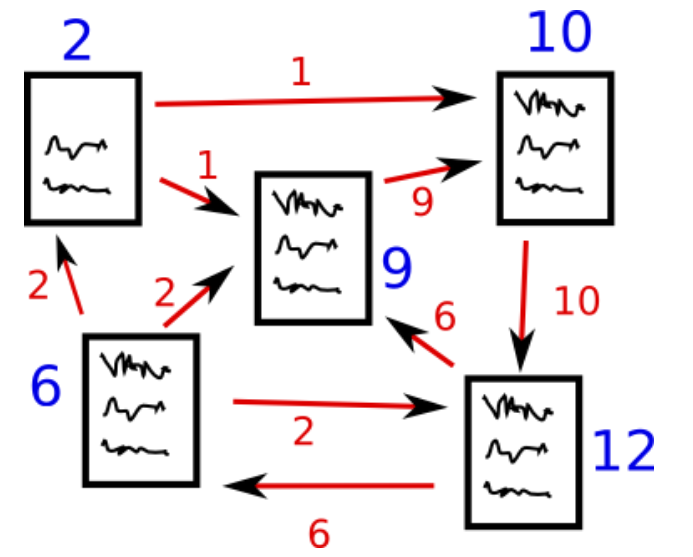
But another unique and unwanted information present were the references made in each speech to other documents. We obviously don’t want that either.

I have written a simple function to clean the speeches. **An important point here is that I haven’t used lemmatization or changed the words to lowercase as it has the potential to change the POS tag of the word.** We certainly don’t want to do that as you will see in the upcoming subsections.

## Understanding the TextRank Algorithm

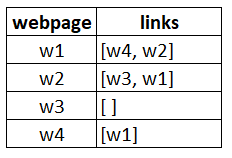
Before getting started with the TextRank algorithm, there’s another algorithm which we should become familiar with – the PageRank algorithm. In fact, this actually inspired TextRank! **PageRank is used primarily for ranking web pages in online search results.** Let’s quickly understand the basics of this algorithm with the help of an example.

### PageRank Algorithm



Source: http://www.scottbot.net/HIAL/

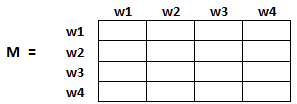
Suppose we have 4 web pages — w1, w2, w3, and w4. These pages contain links pointing to one another. Some pages might have no link – these are called dangling pages.



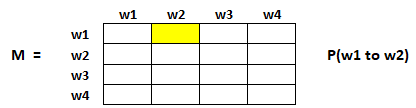
* Web page w1 has links directing to w2 and w4
* w2 has links for w3 and w1
* w4 has links only for the web page w1
* w3 has no links and hence it will be called a dangling page

In order to rank these pages, we would have to compute a score called the **PageRank score**. This score is the probability of a user visiting that page.

To capture the probabilities of users navigating from one page to another, we will create a square **matrix M**, having n rows and n columns, where **n** is the number of web pages.



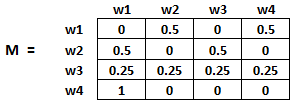
Each element of this matrix denotes the probability of a user transitioning from one web page to another. For example, the highlighted cell below contains the probability of transition from w1 to w2.



The initialization of the probabilities is explained in the steps below:

1. Probability of going from page i to j, i.e., M[ i ][ j ], is initialized with **1/(number of unique links in web page wi)**
2. If there is no link between the page i and j, then the probability will be initialized with **0**
3. If a user has landed on a dangling page, then it is assumed that he is equally likely to transition to any page. Hence, M[ i ][ j ] will be initialized with **1/(number of web pages)**

Hence, in our case, the matrix M will be initialized as follows:



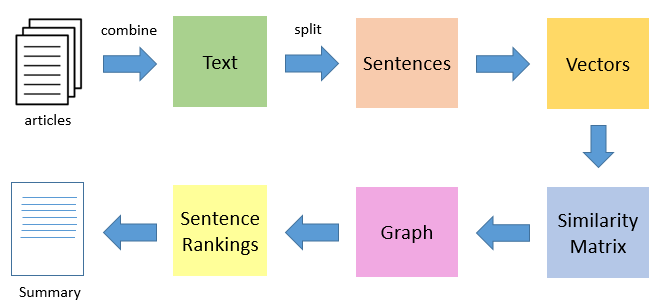
Finally, the values in this matrix will be updated in an iterative fashion to arrive at the web page rankings.

### TextRank Algorithm

Let’s understand the TextRank algorithm, now that we have a grasp on PageRank. I have listed the similarities between these two algorithms below:

* In place of web pages, we use sentences
* Similarity between any two sentences is used as an equivalent to the web page transition probability
* The similarity scores are stored in a square matrix, similar to the matrix M used for PageRank

**TextRank is an extractive and unsupervised text summarization technique.** Let’s take a look at the flow of the TextRank algorithm that we will be following:

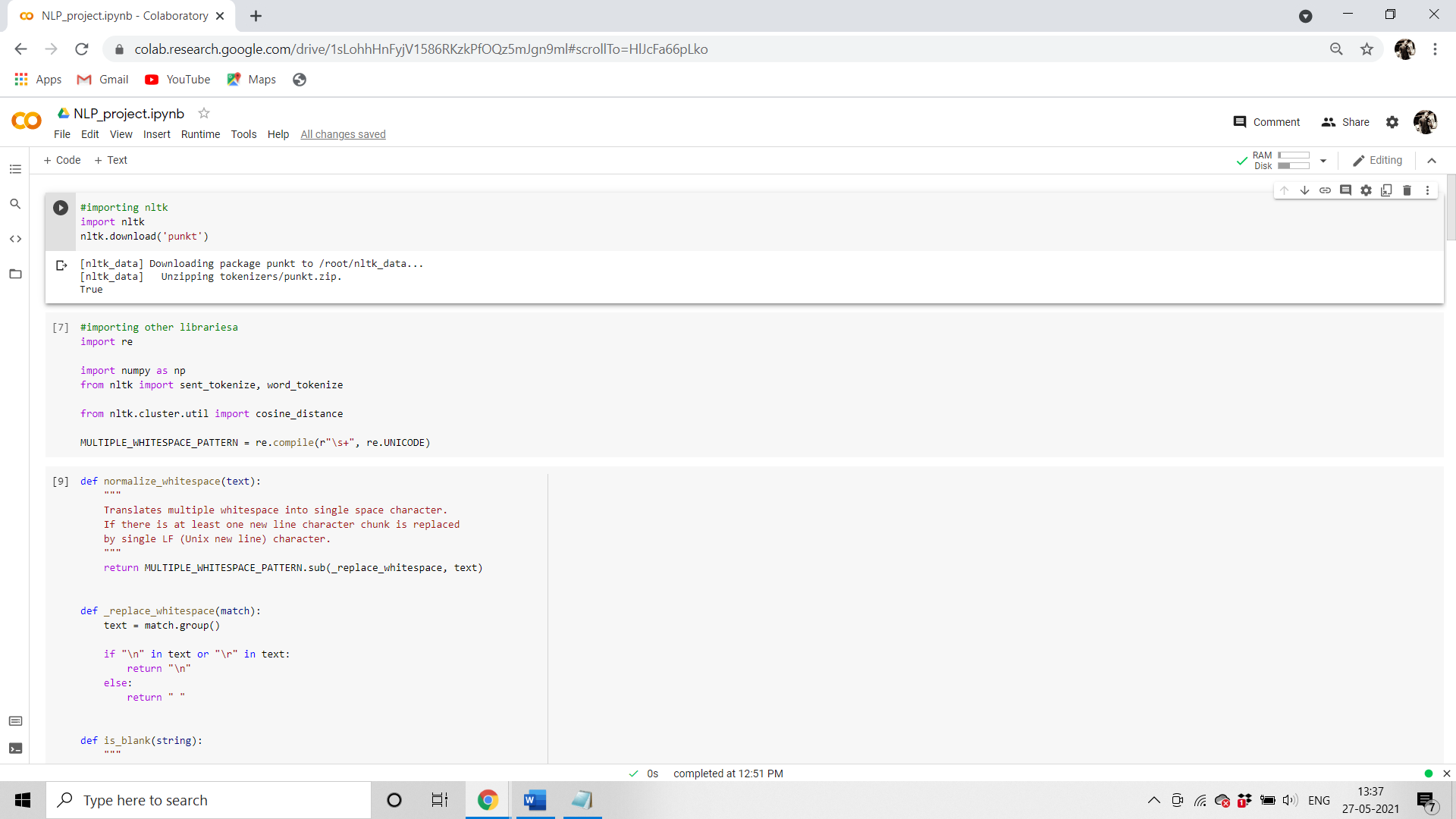


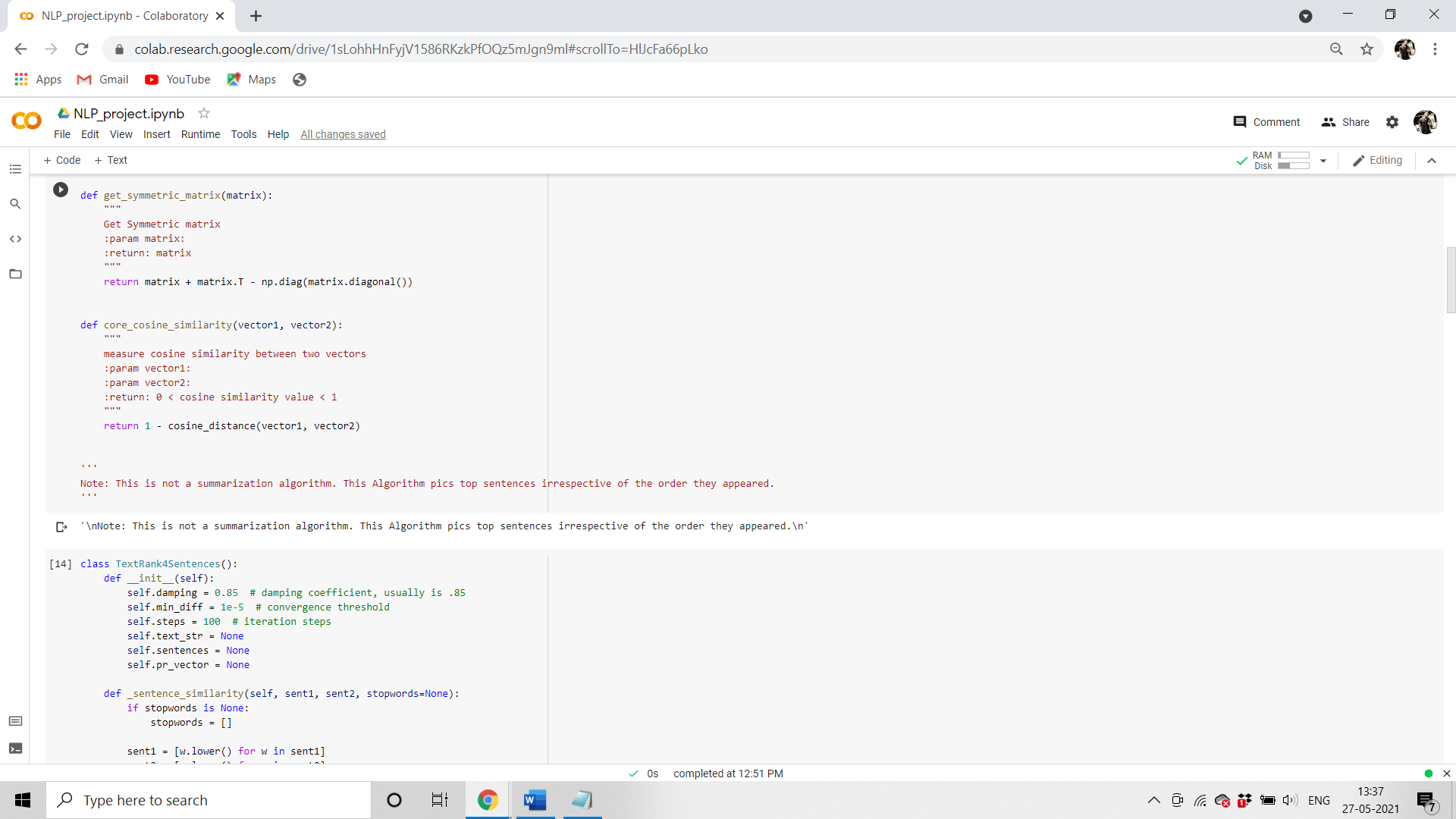
* The first step would be to concatenate all the text contained in the articles
* Then split the text into individual sentences
* In the next step, we will find vector representation (word embeddings) for each and every sentence
* Similarities between sentence vectors are then calculated and stored in a matrix
* The similarity matrix is then converted into a graph, with sentences as vertices and similarity scores as edges, for sentence rank calculation
* Finally, a certain number of top-ranked sentences form the final summary

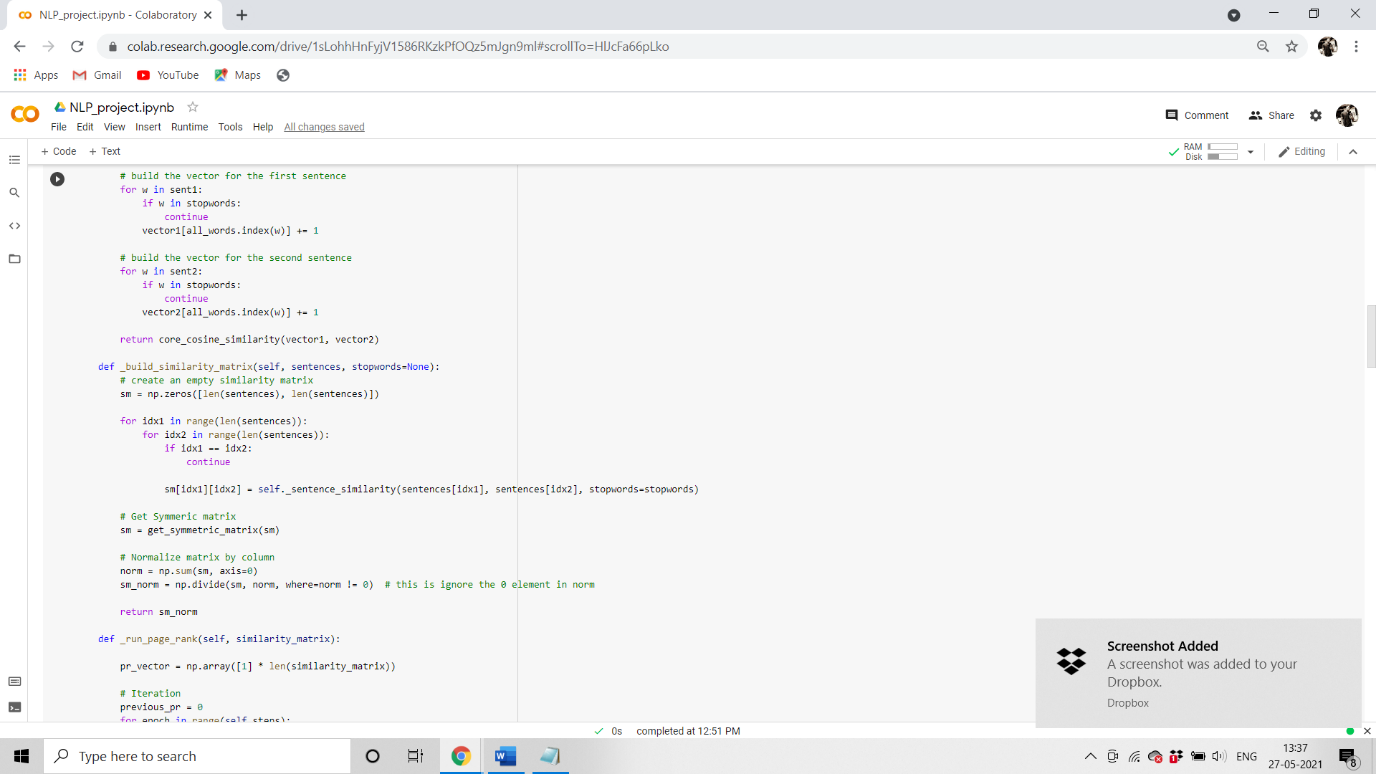
1. **EXPERIMENT AND RESULT:**

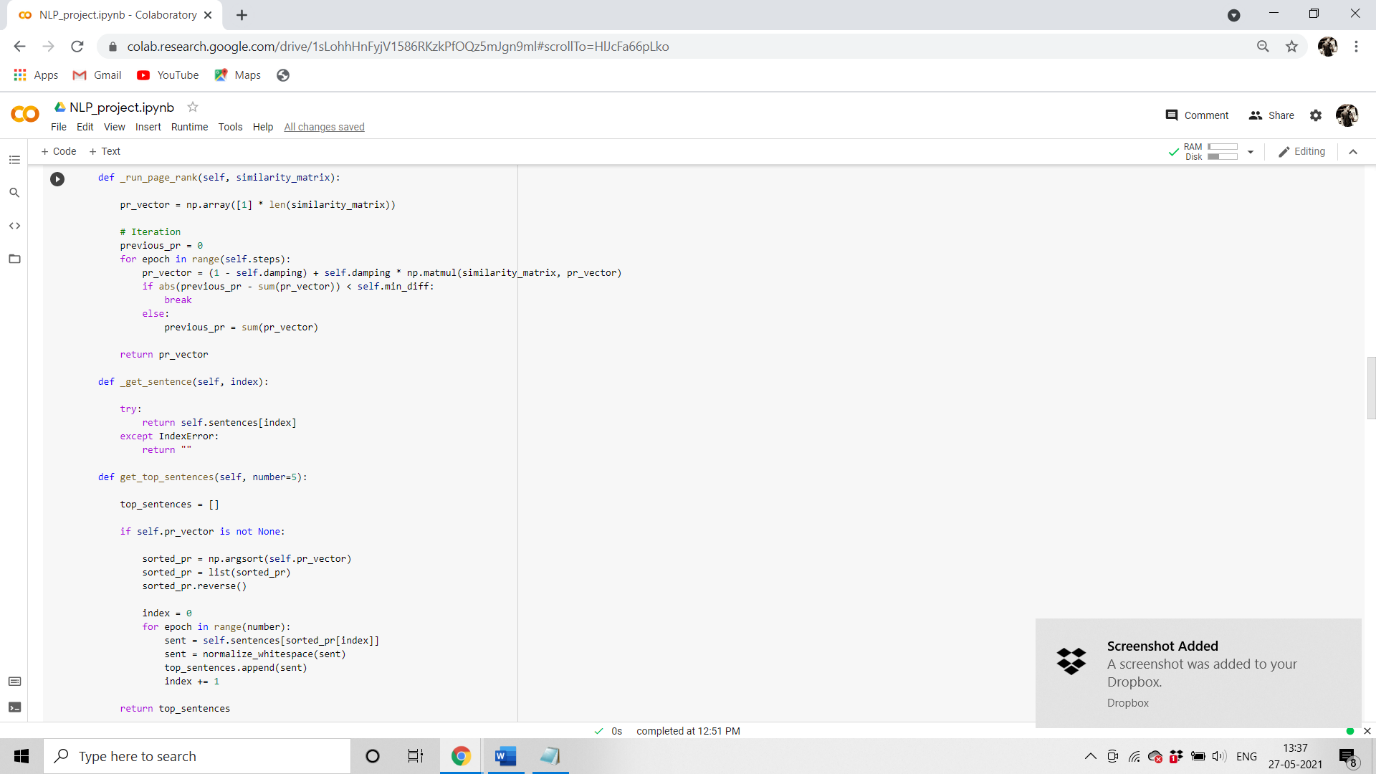
**GOOGLE COLAB NOTEBOOK:**

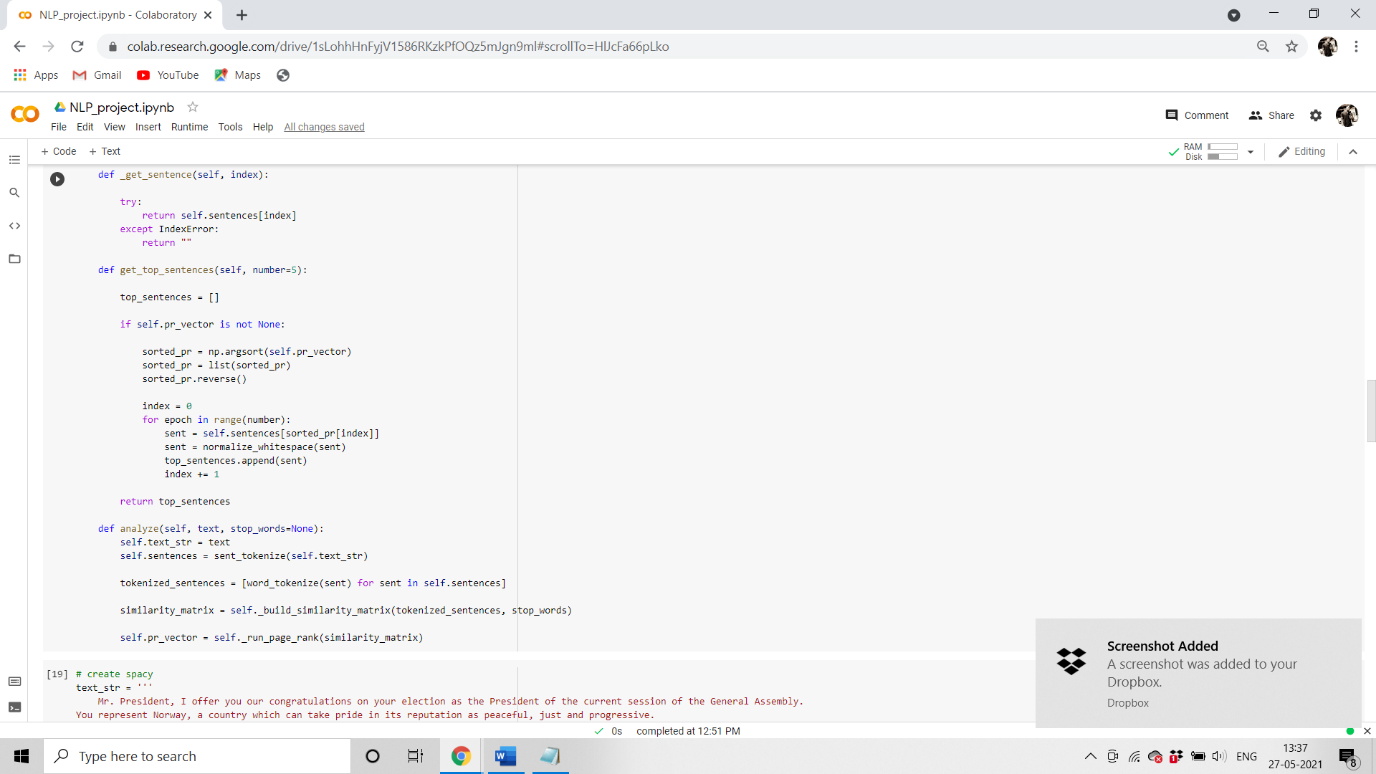
[**https://colab.research.google.com/drive/1sLohhHnFyjV1586RKzkPfOQz5mJgn9ml?usp=sharing**](https://colab.research.google.com/drive/1sLohhHnFyjV1586RKzkPfOQz5mJgn9ml?usp=sharing)

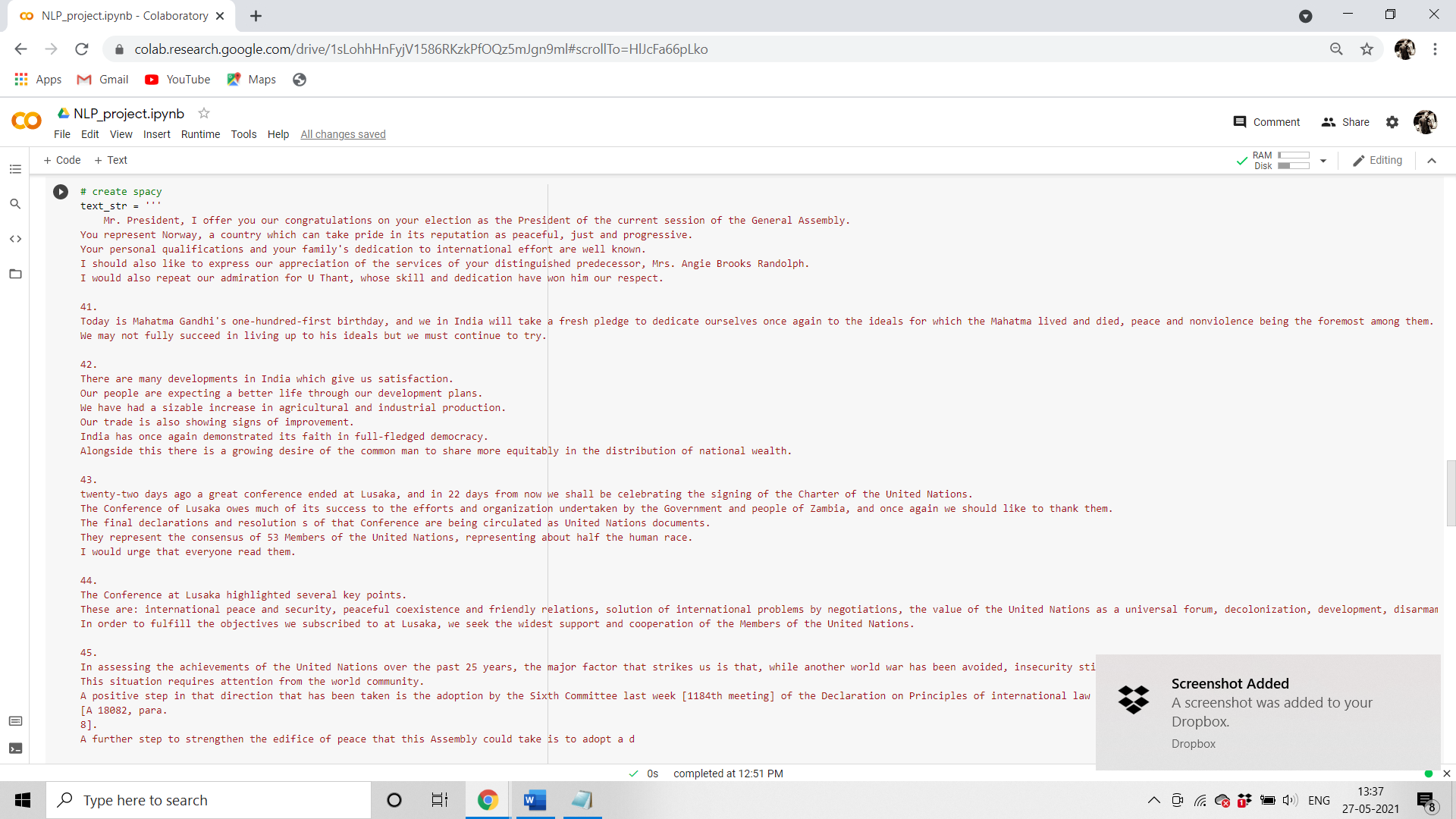


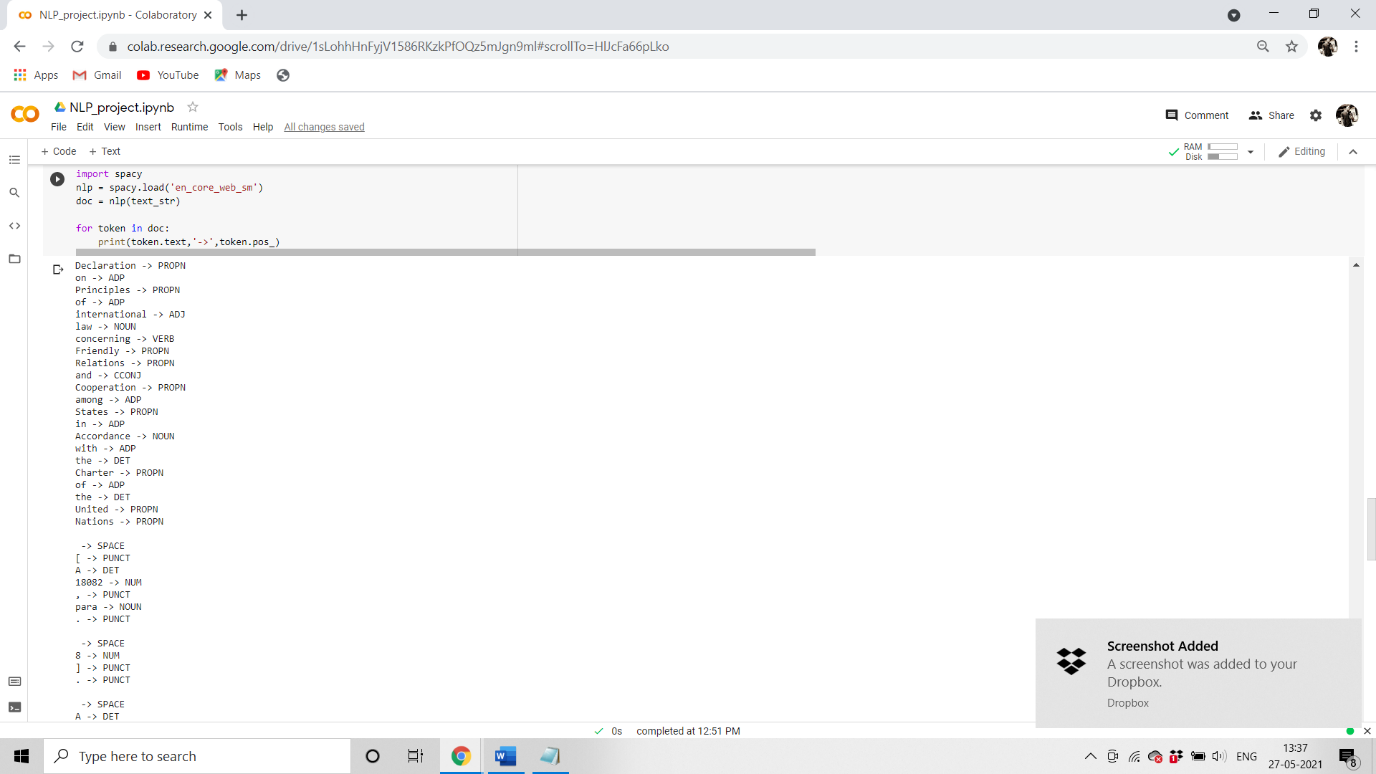


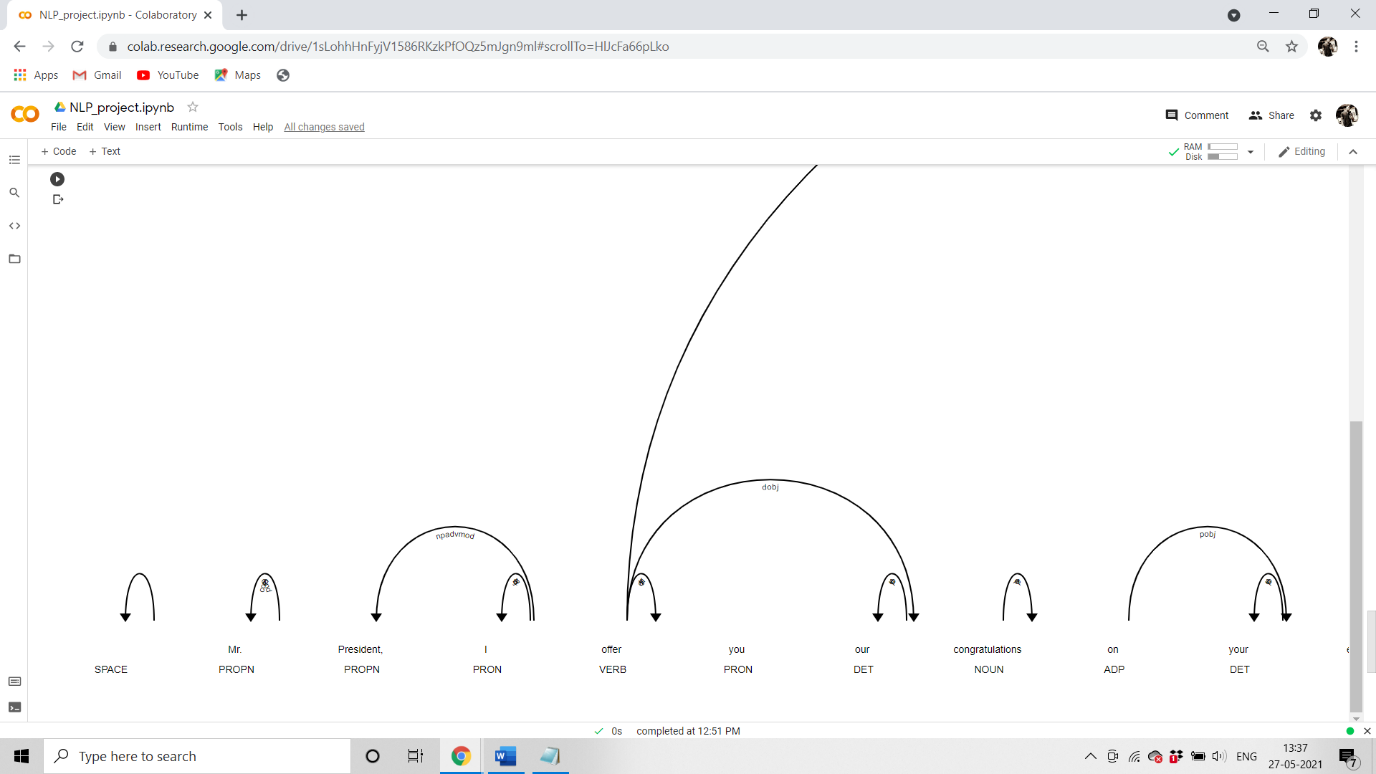


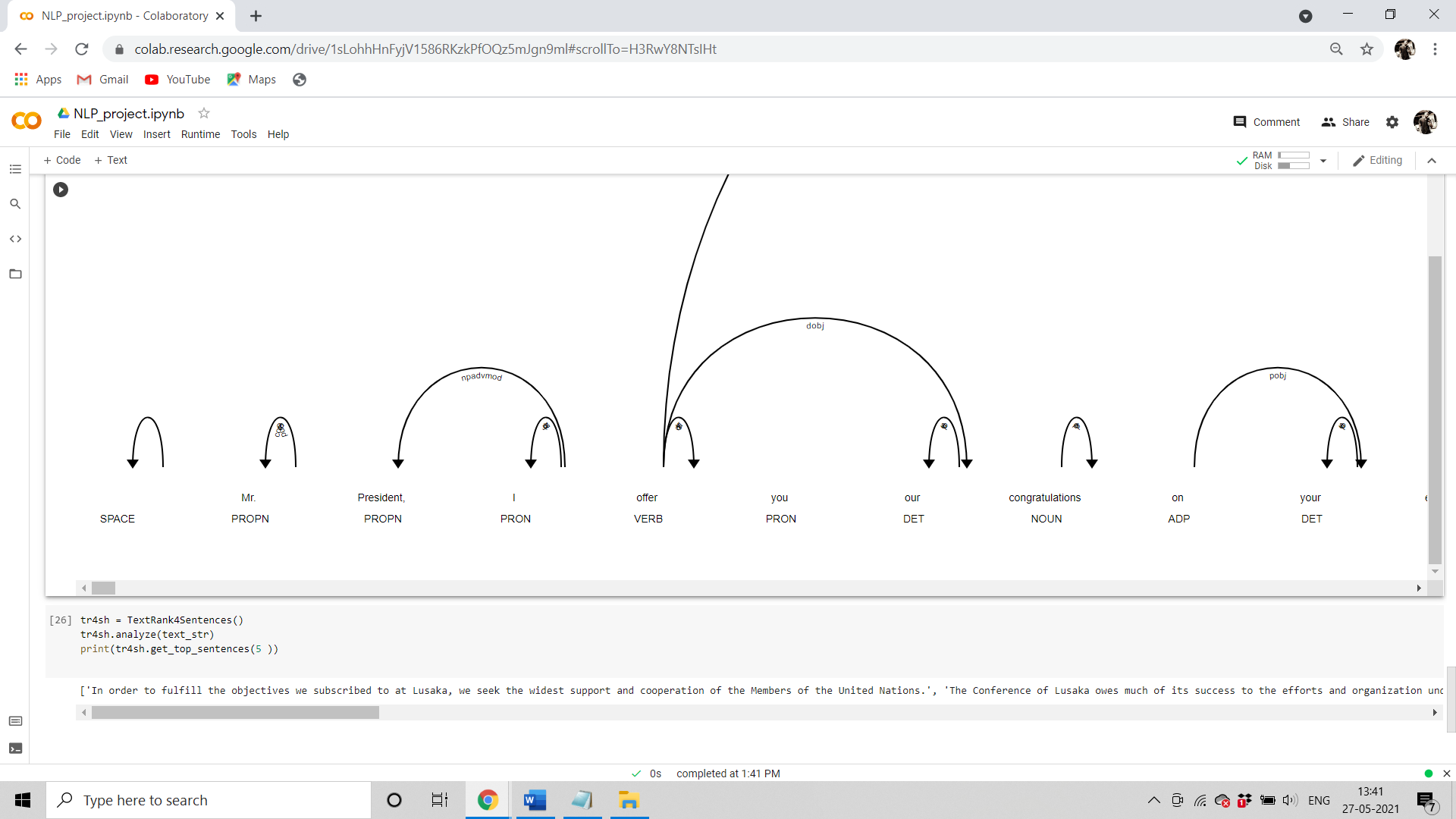












1. **CONCLUSION**

Text has been pre-processed, and cleaned with suitable functions and libraries. Some basic data visualization has been done and analysed.

Data converted from texts to vector so that they can be trained with suitable algorithm. Text Rank Algorithm has been implemented and corresponding information has been extracted from given texts.

**6.REFERENCES**

1.

<https://www.ontotext.com/knowledgehub/fundamentals/information-extraction/>

2.<https://www.analyticsvidhya.com/blog/2020/06/nlp-project-information-extraction/>

1. <https://en.wikipedia.org/wiki/Information_extraction#:~:text=Information%20extraction%20(IE)%20is%20the,and%20other%20electronically%20represented%20sources>.
2. <https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/>
3. <https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/>
4. <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0254-8>
5. <https://www.cse.iitb.ac.in/~soumen/readings/papers/Sarawagi2008ie.pdf>
6. <http://keg.cs.tsinghua.edu.cn/jietang/publications/Tang-et-al-Information_Extraction.pdf>
7. <https://nanonets.com/blog/named-entity-recognition-ner-information-extraction/>
8. <https://whatis.techtarget.com/definition/information-extraction-IE>
9. <https://www.analyticsvidhya.com/blog/2019/09/introduction-information-extraction-python-spacy/>
10. <https://www.researchgate.net/publication/282382161_Information_Extraction_methods_and_extraction_techniques_in_the_chemical_document's_contents_Survey>
11. <http://cseweb.ucsd.edu/~elkan/BWI.pdf>
12. <https://www.aclweb.org/anthology/P04-3020.pdf>
13. <https://medium.datadriveninvestor.com/discussing-textrank-a-unsupervised-algorithm-for-extracting-meaning-from-text-217d10de066>