

|  |
| --- |
| **Introduction to Image Processing** |
| FINAL PROJECT REPORT |
| **Department of Computer Science and Engineering**  **The NorthCap University, Gurugram** |

**Puru Behl 18csu162**

**Rashi Kapoor 18csu168**

**Rithwick Negi 18csu173**

**Ritik Chadha 18csu174**

SMART TEXT SCANNER

ABSTRACT

The aim of this project is to develop a Text recognition software from an image. The purpose of this application is to recognize text in scanned text documents, text images, and any picture stored in the device in order to reuse it later. This application will allow its users to perform many actions in a few minutes, such as copy text from these aforementioned documents and modify it by translating into various languages like French, German and Hindi. The project also aims to answer various questions asked by users from the text, summarizing the text and changing templates of the text. We have created a data WebApp using Streamlit for deployment of our software.

INTRODUCTION

In the running world, there is growing demand for the software systems to recognize characters in computer systems when information is scanned through paper documents as we know that we have a number of newspapers and books which are in printed format related to different subjects. These days there is a huge demand in “storing the information available in these paper documents into a computer storage disk and then later reusing this information by searching process”. But to reuse this information it is very difficult to read the individual contents and search the contents from these documents line-by-line and word-by-word. The reason for this difficulty is that the font characteristics of the characters in paper documents are different from the font of the characters in the computer system. As a result, the computer is unable to recognize the characters while reading them. This concept of storing the contents of paper documents in a computer storage place and then reading and searching the content is called document processing. Sometimes in this document processing we need to process the information that is related to languages other than the English in the world. For this document processing we need a software system called Character Recognition System. This process is also called Document Image Analysis (DIA). Thus our need is to develop a character recognition software system to perform Document Image Analysis which transforms documents in paper format to electronic format.

Choosing Optical Character Recognition as the main fundamental technique to recognize characters. To effectively use Optical Character Recognition for character recognition in-order to perform Document Image Analysis (DIA), we are using the information in Grid format. . This system is thus effective and useful in Virtual Digital Library’s design and construction.The main purpose of Optical Character Recognition (OCR) system based on a grid infrastructure is to perform Document Image Analysis, document processing of electronic document formats converted from paper formats more effectively and efficiently. This improves the accuracy of recognizing the characters during document processing compared to various existing available character recognition methods. Here OCR technique derives the meaning of the characters, their font properties from their bit-mapped images. ¬ The primary objective is to speed up the process of character recognition in document processing. As a result, the system can process a huge number of documents with-in less time and hence saves the time.

MODELS THAT ARE USED

# Pytesseract: Python Optical Character Recognition

Humans can understand the contents of an image simply by looking. We perceive the text on the image as text and can read it.

Computers don't work the same way. They need something more concrete, organized in a way they can understand. This is where Optical Character Recognition (OCR) kicks in.

**Optical Character Recognition** involves the detection of text content on images and translation of the images to encoded text that the computer can easily understand. An image containing text is scanned and analyzed in order to identify the characters in it. Upon identification, the character is converted to machine-encoded text.

Steps involved in OCR:

1. The image is first scanned and the text and graphics elements are converted into a bitmap, which is essentially a matrix of black and white dots.
2. The image is then pre-processed where the brightness and contrast are adjusted to enhance the accuracy of the process.
3. The image is now split into zones identifying the areas of interest such as where the images or text are and this helps kickoff the extraction process.
4. The areas containing text can now be broken down further into lines and words and characters and now the software is able to match the characters through comparison and various detection algorithms

The final result is the text in the image that we're given!

The process may not be 100% accurate and might need human intervention to correct some elements that were not scanned correctly. Error correction can also be achieved using a dictionary or even Natural Language Processing (NLP).

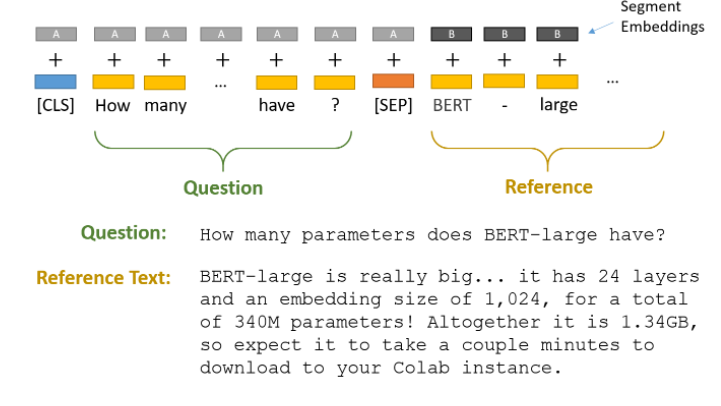
**Pytesseract** is a Python wrapper for Tesseract. Using this you can easily implement your own text recognizer using Tesseract OCR by writing a simple Python script.

# 2. Questing Answering using BERT

Given a question, and a passage of text containing the answer, BERT needs to highlight the “span” of text corresponding to the correct answer.

**BERT Input Format**

To feed a QA task into BERT, we pack both the question and the reference text into the input.

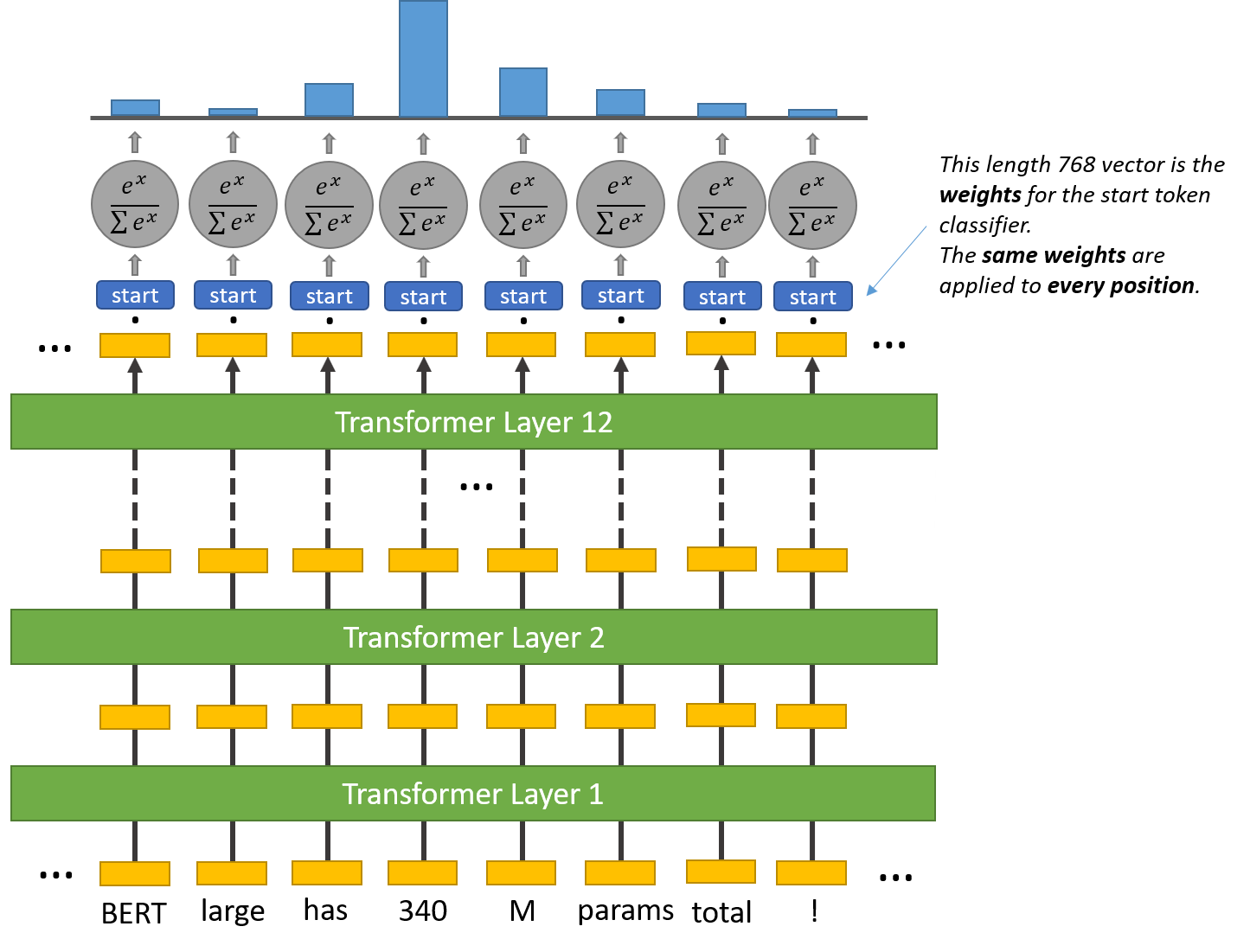


The two pieces of text are separated by the special [SEP] token.

BERT also uses “Segment Embeddings” to differentiate the question from the reference text. These are simply two embeddings (for segments “A” and “B”) that BERT learned, and which it adds to the token embeddings before feeding them into the input layer.

**Start & End Token Classifiers**

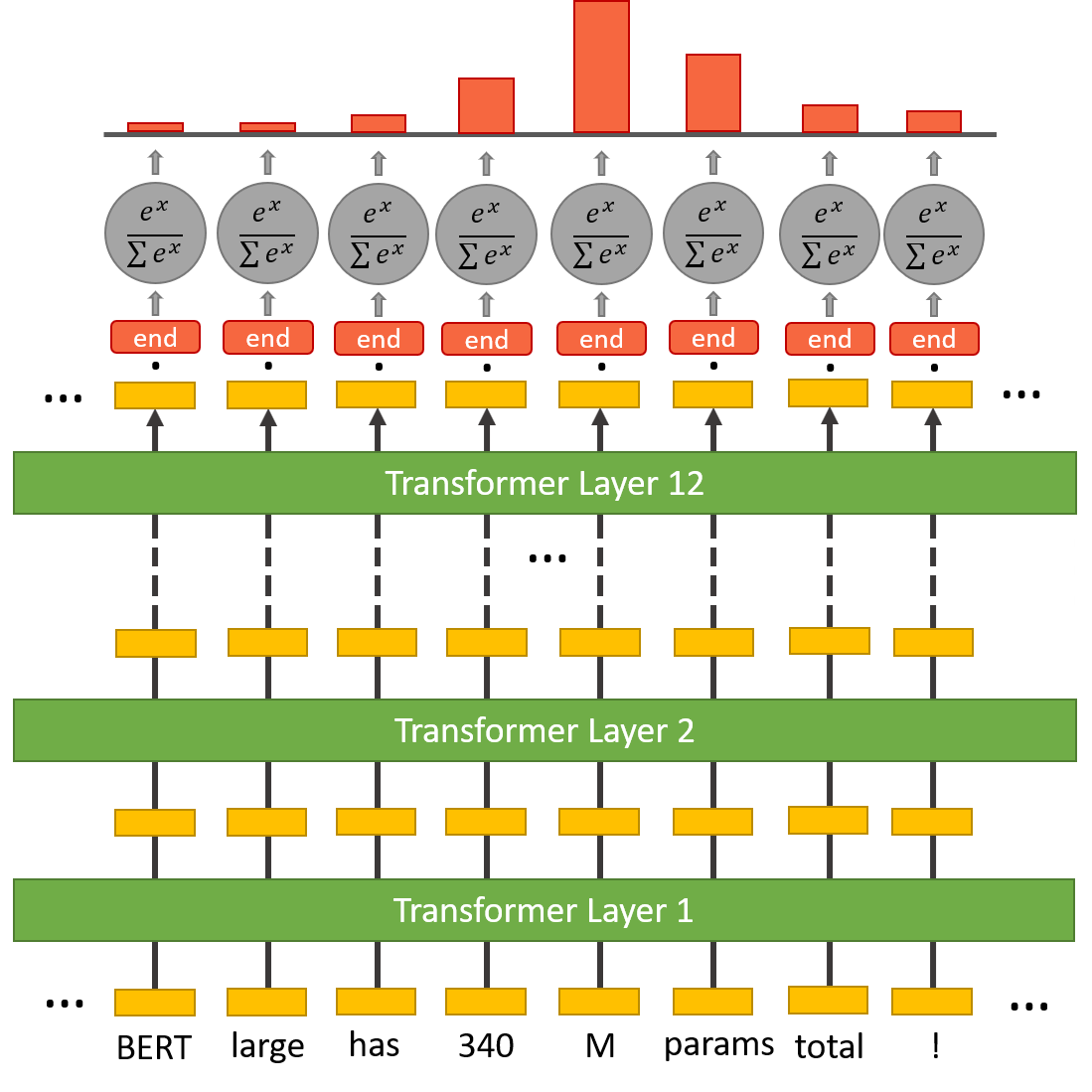
BERT needs to highlight a “span” of text containing the answer–this is represented as simply predicting which token marks the start of the answer, and which token marks the end.



For every token in the text, we feed its final embedding into the start token classifier. The start token classifier only has a single set of weights (represented by the blue “start” rectangle in the above illustration) which it applies to every word.

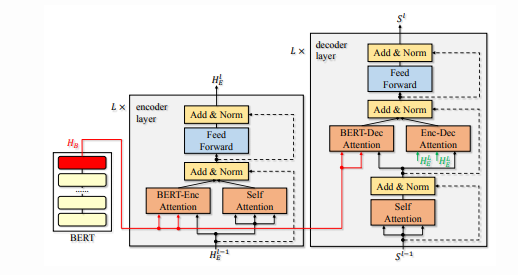
After taking the dot product between the output embeddings and the ‘start’ weights, we apply the SoftMax activation to produce a probability distribution over all of the words. Whichever word has the highest probability of being the start token is the one that we pick.

We repeat this process for the end token–we have a separate weight vector.



3. Translation using BERT

Representations from BERT brought improvement in most natural language processing tasks.The trick with which they make it work is adding other attention layers: from the encoder to BERT and from the decoder to BERT. In this setup, the model does hot have to rely on representations from BERT, it learns its own input representation, but it can cherry-pick a useful piece of information from BERT when necessary.



BERT is most commonly used as part of an embedding-prediction pipeline, where additional neural layers are appended to the end of the twelve-layer BERT model.BERT is able to better capture long-distance dependencies. Furthermore, BERT is pre-trained on two sentences to predict if *Sentence B* naturally follows *Sentence A*. This approach lends itself to tasks that can be modelled with a two-sentence structure.

4. Text summarization with BERT

With the advent of new research in the state of the art models, more powerful algorithms are being developed that focus on low code environments to make Machine learning accessible for everyone. Analogous to transfer learning models for image classification or face recognition in the field of computer vision, BERT is highly powerful models that solve problems using natural language processing. These models have dominated the world of NLP by making tasks like POS tagging,text summarization etc very easy yet effective.

Text summarization is the concept of machine learning to condense a document or a set of documents into brief paragraphs or statements using mathematical methods. NLP broadly classifies text summarization into 2 groups.

1. Extractive text summarization: here, the model summarizes long documents and represents them in smaller simpler sentences.
2. Abstractive text summarization: the model has to produce a summary based on a topic without prior content provided.

The input format of BERTSUM is different when compared to the original model. Here, a [CLS] token is added at the start of each sentence in order to separate multiple sentences and to collect features of the preceding sentence. There is also a difference in segment embeddings. In the case of BERTSUM, each sentence is assigned an embedding of Ea or Eb depending on whether the sentence is even or odd. If the sequence is [s1,s2,s3] then the segment embeddings are [Ea, Eb, Ea]. This way, all sentences are embedded and sent into further layers.

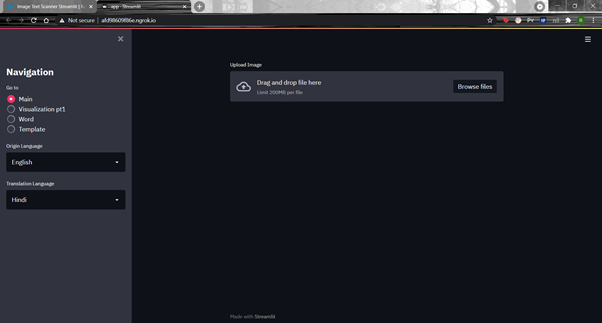
**BERTSUM** assigns scores to each sentence that represents how much value that sentence adds to the overall document. So, [s1,s2,s3] is assigned [score1, score2, score3]. The sentences with the highest scores are then collected and rearranged to give the overall summary of the article.

RESULTS:

We have created a data WebApp using Streamlit to deploy our software.

**Streamlit** is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science.

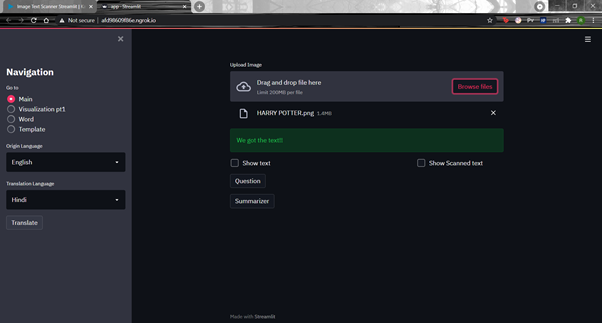
This is the main page of our web application. As you can see, we first have buttons to upload an image. We also added a sidebar for various other features like Visualizations, POS tagging, WordCloud and even added a feature to change the template of our image. We also added various other features/buttons like ‘to ask a question’, ‘summarize’ and ‘translate’ based on the reference text of our image uploaded. These buttons are not seen initially as we start the web app and appear only when we upload an image on our website. This will be demonstrated in the upcoming results.



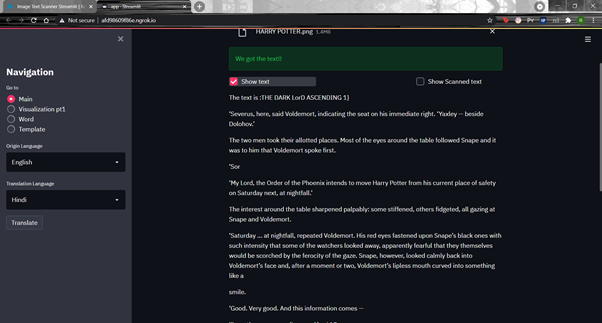
As you can see, as soon as we upload an image the ‘Question’ , ‘Summarize’ and the ‘Translate’ button appear on our screen.

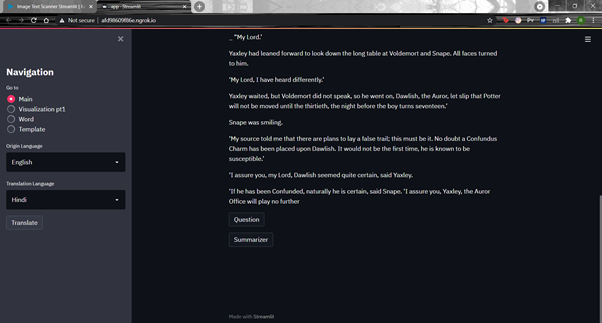
Followed by a text “We got the text” indicating that the image with reference text has been uploaded successfully. We can also see two checkboxes namely ‘Show text’ and ‘Show scanned image’ if the prior is selected our app will extract the text from our image and display it on our website. If the latter is selected our site will show the scanned version of our image which is done using various Image Processing techniques. We will see the demonstration in the upcoming results.

We uploaded an image from the Harry Potter textbook to our website for further analysis.

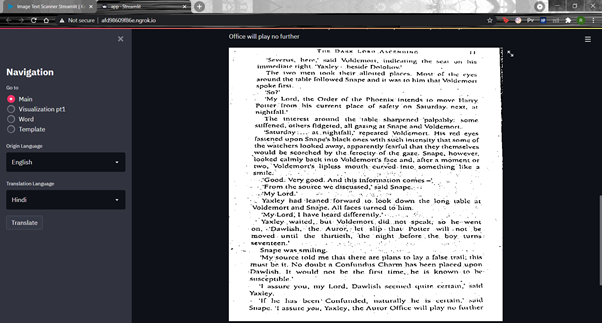


Here, we have selected the ‘Show Text’ checkbox and we can see it has displayed the text for us.





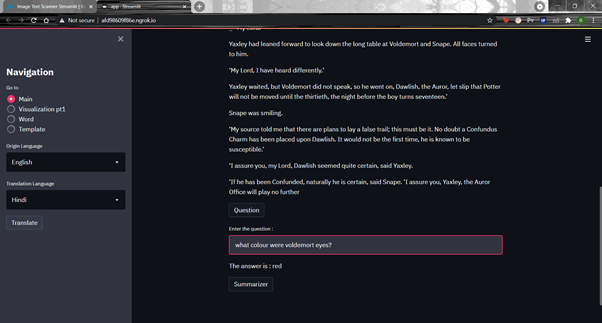
Then, we selected the ‘Show scanned image’ checkbox and we can see it has displayed the scanned image for us. Various Image Processing techniques like Gray scaling, Thresholding, Gaussian blur, Median filtering, Canny edge detection and finally Biggest Contour method.



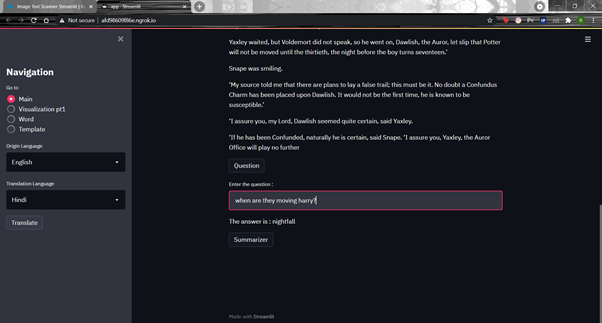
Then we click on the ‘Question’ button and the website asks the user to input a question. After entering the question, it is passed to the question answering model (BERT) in the backend and we see the output in the form of answer just below our button.

Here we asked a question based on our reference text. “What colour were Voldemort’s eyes?” and the model predicted the right answer was ‘red’.

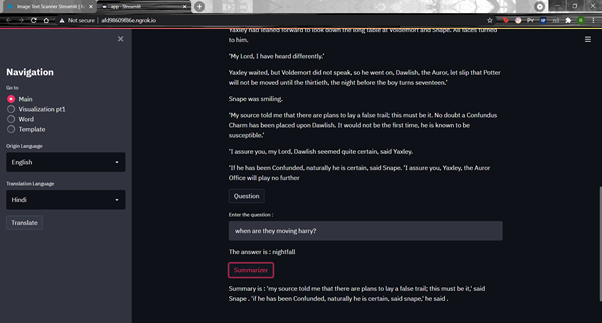
NOTE: Our model only works on the question asked from the reference text. If the question is related to something out of context, the model will not be able to predict correctly but still it will predict something.



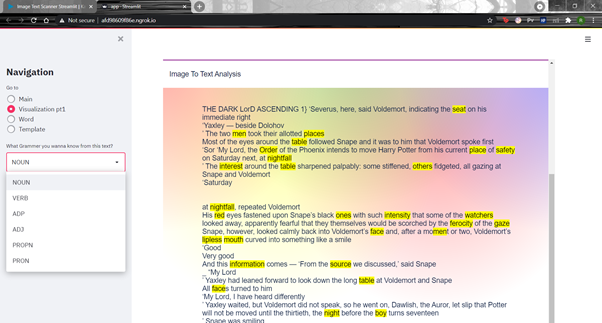
Here we asked another question: “When are they moving harry?” and the model predicted this right as well with the output “Nightfall”



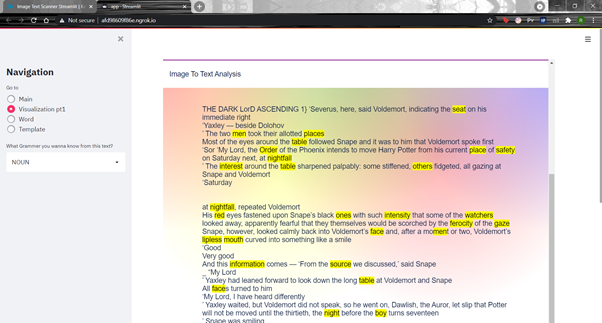
As we click on the ‘Summarizer’ button The summarizer model is doing its work on the backend and finally the website displays a short summary of the uploaded image or reference text.



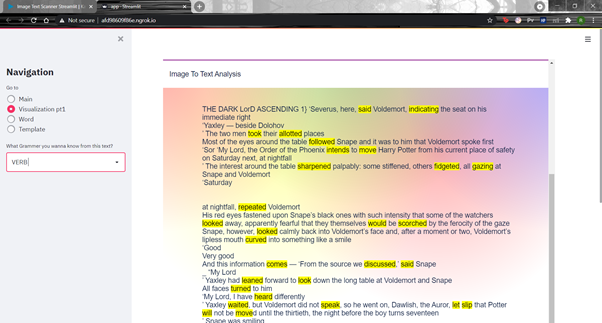
In the visualisation part, We can highlight different parts of speech from our reference text. This is done using POS tagging.



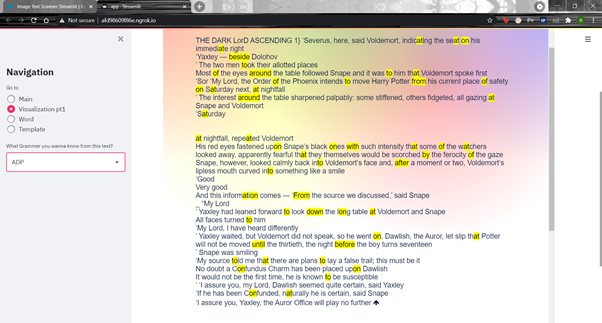
**Highlighting Nouns**



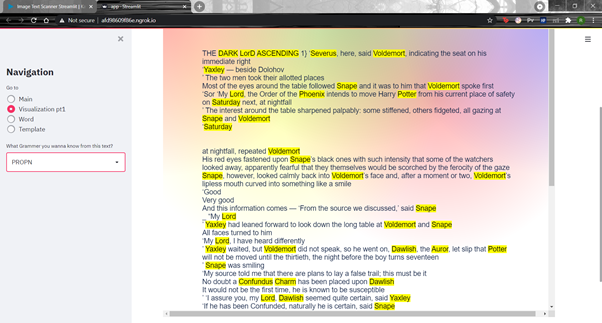
**Highlighting Verbs**



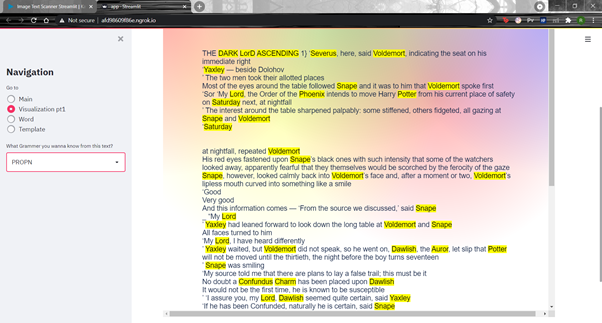
**Highlighting Adposition**



**Highlighting Adjectives**



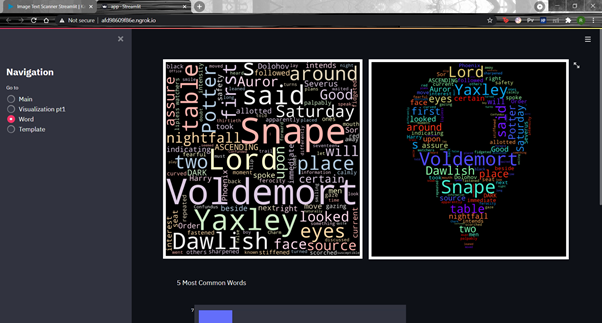
**Highlighting Proper Nouns**

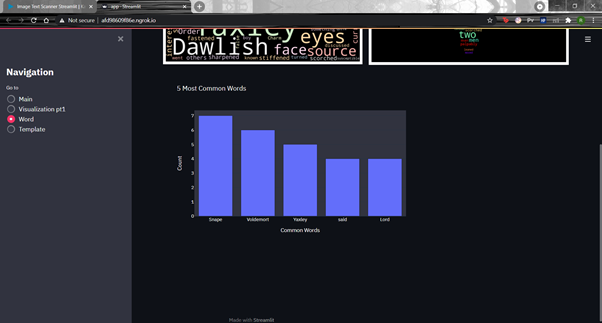


As we select ‘Word’ button from the sidebar we can see three visualisations:

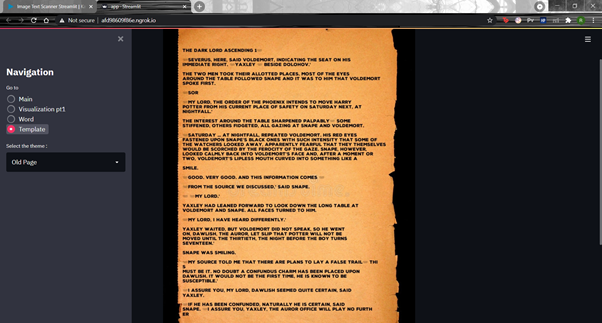
Two **Word Clouds** - One with a mask and one without the mask.

One **Countplot** depicting the Frequency of words in our text. We displayed the Top 5 words in our reference text or uploaded image

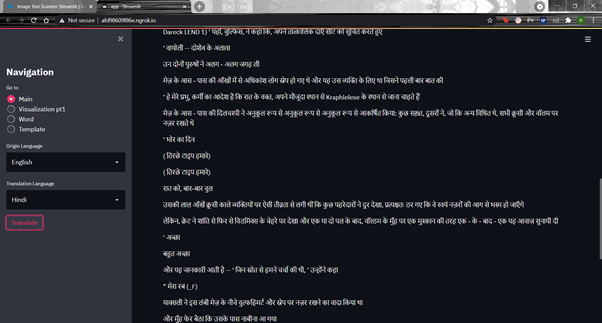


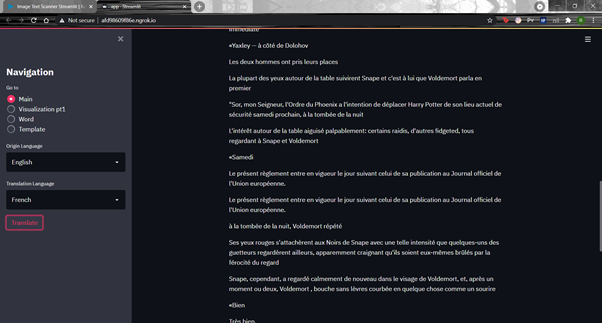


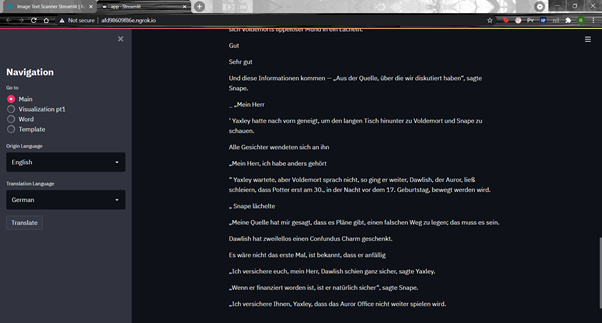
As we select the ‘Template’ button from the sidebar we can see our image in two different templates. Template 1 is called ‘Old page’ and Template 2 is called ‘Blank page’. The results are shown below.



Lastly, this is the working of our translator button. We have four translation choices in this task namely English to Hindi, English to French, English to German and Hindi to English. The translation model is doing its work on the backend and the result is a translated output of the desired language as you can see below.







Conclusion

We would like to conclude that our image to text scanner is working perfectly. It is able to scan the text from the image and provide us with appropriate text. It is able to solve questions from the text which is tricky and is also able to summarize and translate the text. We can change the template of the image and add the text from the actual image to the new template which also provides new fonts to the text. It also provides the function of pos tagging which might be useful for teachers in the lower standards to teach to small kids using some basic visualization since pictures tell a better picture than the text.