Algolabs NLP Internship report

Extracting text from embedded images and enabling Question Answering

May 2022-July 2022

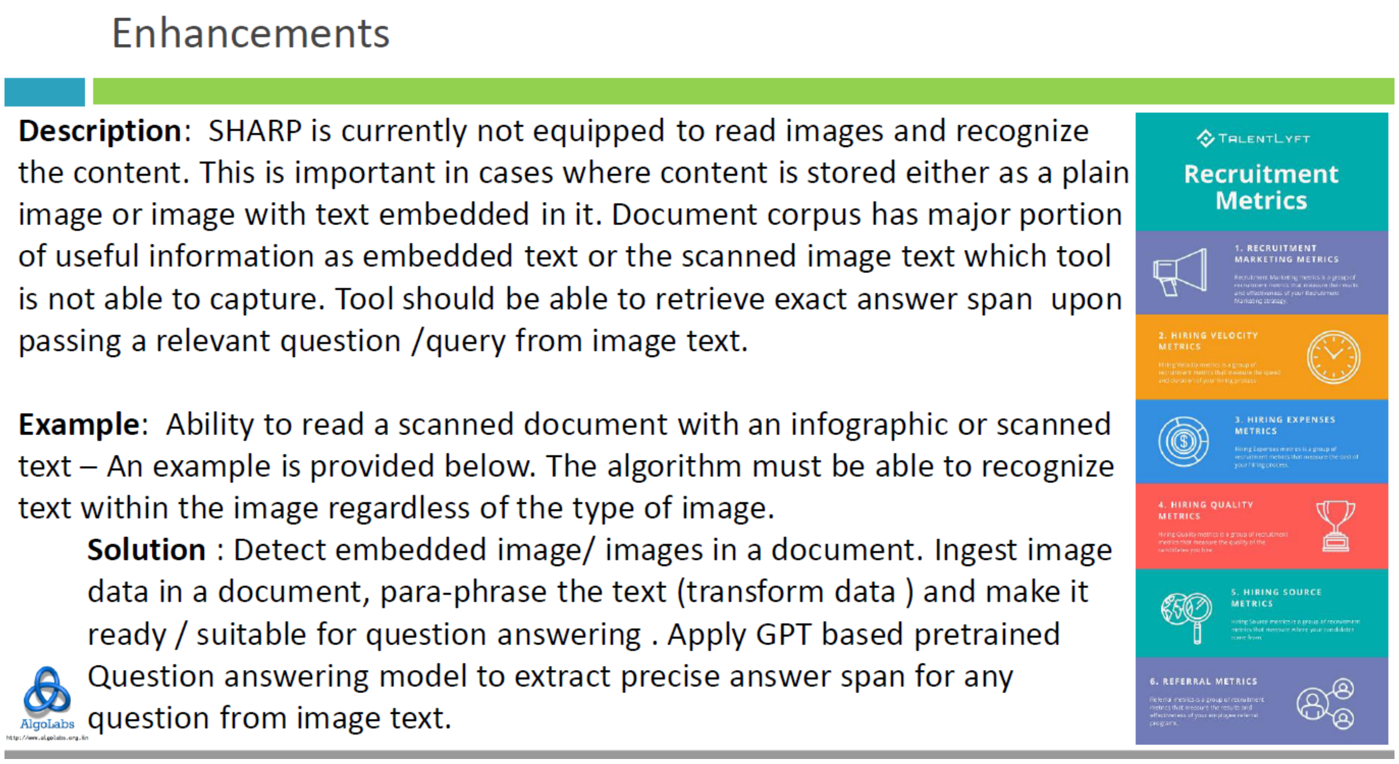
Project Advisor- Ms. Suchitra Karunakaran, AlgoLabs

by

Raktim Dey

Introduction

The issues being faced by the model are:



Enhancements

Description: SHARP is currently not equipped to read images and recognize the content. This is important in cases where content is stored either as a plain image or image with text embedded in it. Document corpus has major portion of useful information as embedded text or the scanned image text which tool is not able to capture. Tool should be able to retrieve exact answer span upon passing a relevant question /query from image text.

Example: Ability to read a scanned document with an infographic or scanned text – An example is provided below. The algorithm must be able to recognize text within the image regardless of the type of image.

Solution : Detect embedded image/ images in a document. Ingest image data in a document, para-phrase the text (transform data) and make it ready / suitable for question answering . Apply GPT based pretrained Question answering model to extract precise answer span for any question from image text.

TalentLyft
Recruitment Metrics
1. RECRUITMENT MARKETING METRICS
Recruitment Marketing metrics is a group of recruitment metrics that measure the results and effectiveness of your Recruitment Marketing strategy.
2. HIRING VELOCITY METRICS
Hiring Velocity metrics is a group of recruitment metrics that measure the speed and duration of your hiring process.
3. HIRING EXPENSES METRICS
Hiring Expenses metrics is a group of recruitment metrics that measure the cost of your hiring process.
4. HIRING QUALITY METRICS
Hiring Quality metrics is a group of recruitment metrics that measure the quality of the candidates you hire.
5. HIRING SOURCE METRICS
Hiring Source metrics is a group of recruitment metrics that measure where your candidates come from.
6. REFERRAL METRICS
Referral metrics is a group of recruitment metrics that measure the results and effectiveness of your employee referral programs.
AlgoLabs
http://www.algolabs.org.in

Data Description

The field of computer vision has seen tremendous development in the recent past leading to a host of practical applications in a wide variety of industries and use cases. Optical Character Recognition (OCR) is one such application of computer vision with the potential to automate many tedious but necessary tasks. OCR technology can be used to process digital documents (PDFs, scanned documents, images of documents and the like), far more efficiently than humans can. In a nutshell, OCR can "read" a document and convert images of text into actual text. Current state-of-the-art algorithms are capable of near-flawless recognition of printed text, with handwriting recognition not too far behind (as long as the handwriting isn't somewhere between a child's scrawl and a doctor's note like mine).

The first objective was to learn about **Object detection** and **OCR** models to extract text from images embedded in documents of all types and also the documents itself.
OCR and Text extraction are performed on a collection of textbooks, comic books and magazines that can be found here:
https://drive.google.com/drive/folders/1ZLh-6Ty--NW5IlZrInTw-EXaBrozSQKj
The books used were as follows:

- Automotive Engineering May 2022.pdf
- Believer's Magazine June 2022.pdf
- Bowls International Issue 495 – June 2022.pdf
- Boxing News May 26, 2022.pdf
- Britain at War Issue 182 June 2022.pdf
- Cigars of the Pharaoh (The Adventures of Tintin 4) (z-lib.org).pdf,
- Design As Art (Bruno Munari) (z-lib.org).epub
- Diary of a Wimpy Kid (Jeff Kinney) (z-lib.org).pdf
- Early Civilization in Asia, 3000-1000 BCE Map (Maps.com) (z-lib.org).pdf
- Incredible History Of Indias Geography (Sanjeev Sanyal) (z-lib.org).pdf
- Prisoners of Geography Ten Maps That Explain Everything About the World (Tim Marshall) (z-lib.org).pdf
- The Everything Kids Sharks Book (Wagner Kathi, Wagner Obe.) (z-lib.org).pdf
- Tintin in The Land of The Soviets (z-lib.org).pdf

Data Cleaning and Processing

Pytesseract OCR library and **Detectron 2** model were needed for text extraction from images.
PYPDF2 and **PyMuPDF** libraries were needed for text extraction from documents.
Different NLP libraries like **spaCy** and **NLTK** were needed to pre-process the data obtained by the OCR models.

Pytesseract OCR and Detectron 2 models were used to extract text from documents by forming bounding boxes on the text parts of each page of the documents and then extracting text from those boxes.

Initially, the text extracted from the pictures had a lot of extra lines, spaces and some non-alphanumeric characters which had to be removed, so I used the Regex library to remove these characters and extra blank lines. Then I saved the processed output as a JSON file, which has values of the form:

I built a class for the above task, which can be seen on the above GitHub link:
https://github.com/raktim022/Text-Extraction-from-Documents-using-Tesseract-OCR

Transformer models were used for text generation in several contexts, that can be seen here:
https://colab.research.google.com/drive/1o2OtWRwZMle5D9R77SZnS7U5ZgcpjrkA?usp=sharing

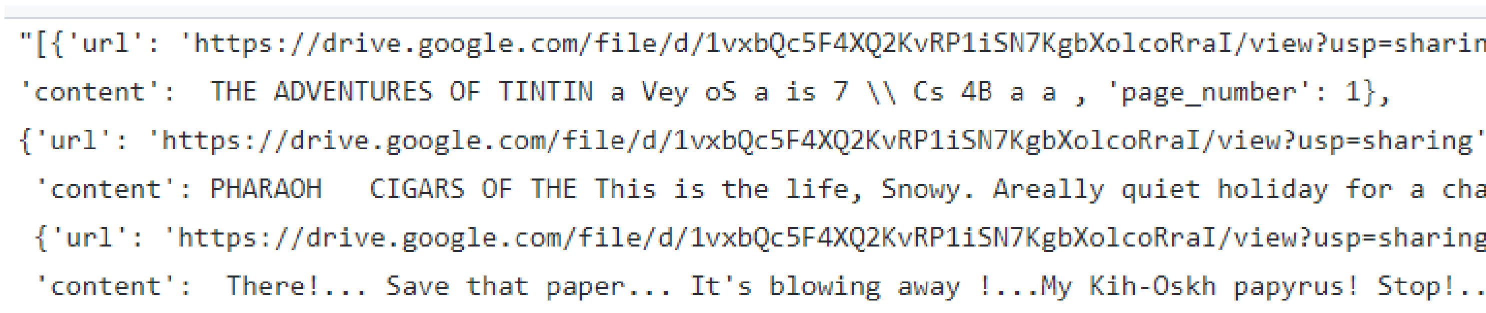
roBERTa and **deepset/minilm-uncased-squad2** models were used for Question Answering. Finetuning the GPT2 model for Question Answering purposes was also needed.

An Information Retrieval architecture using **Hugging-face transformers** and **Happy transformers** was needed to implement a Question Answering system and launch it using a REST API.

So, Happy transformers and NLPCloud were used website for Text generation tasks.
FLASK REST API was used to launch the Question Answering model. HTML and CSS was needed to make the API more visually appealing.

Modelling

Fitz module was used to extract images from the documents. Then Pytesseract library was used to extract pages from documents.
The extracted text was then saved as a .json file with the url of the document and page numbers of the documents containing the pictures attached.
A sample .json file can be seen here:



```
"[{'url': 'https://drive.google.com/file/d/1vxbQc5F4XQ2KvRP1iSN7KgbXolcoRraI/view?usp=sharin
'content':  THE ADVENTURES OF TINTIN a Vey oS a is 7 \\ Cs 4B a a , 'page_number': 1},
{'url': 'https://drive.google.com/file/d/1vxbQc5F4XQ2KvRP1iSN7KgbXolcoRraI/view?usp=sharing'
 'content': PHARAOH   CIGARS OF THE This is the life, Snowy. Areally quiet holiday for a cha
 {'url': 'https://drive.google.com/file/d/1vxbQc5F4XQ2KvRP1iSN7KgbXolcoRraI/view?usp=sharing
 'content':  There!... Save that paper... It's blowing away !...My Kih-Oskh papyrus! Stop!..
```

Transformer models

Transformer models run as follows:

- Each word forming an input sequence is transformed into a
  -dimensional embedding vector.
- Each embedding vector representing an input word is augmented by summing it (element-wise) to a positional encoding vector of the same
  length, hence introducing positional information into the input.
- The augmented embedding vectors are fed into the encoder block, consisting of the two sub layers explained above. Since the encoder attends to all words in the input sequence, irrespective if they precede or succeed the word under consideration, then the Transformer encoder is bidirectional.
- The decoder receives as input its own predicted output word at time-step,
  .
- The input to the decoder is also augmented by positional encoding, in the same manner as this is done on the encoder side.
- The augmented decoder input is fed into the three sub layers comprising the decoder block explained above. Masking is applied in the first sub layer, in order to stop the decoder from attending to succeeding words. At the second sub layer, the decoder also receives the output of the encoder, which now allows the decoder to attend to all of the words in the input sequence.
- The output of the decoder finally passes through a fully connected layer, followed by a soft-max layer, to generate a prediction for the next word of the output sequence.

The second task was to use **Hugging-face** transformer models for text generation and Question answering. Several models like **GPT2, BERT, Pegasus and GPT Neo**(with 1.3 Billion parameters) can be used and that are available on the Hugging-face page.

GPT-2 and GPT Neo were used for text generation and summarization. In order to perform text generation and Question answering, some sample text files are needed. The sample text files that were used can be found here:

https://github.com/raktim022/Text-Generation-to-imitate-fiction-using-GPT-3-model

GPT-2 model could be fine-tuned for Question Answering. So, SQUAD dataset from Kaggle was used for fine-tuning it.
Note: Fine-tuning the GPT-2 model needed a high performance CPU which I do not have, so I had to use a pre-trained model for Question Answering.

The final task was to build a REST API using FLASK to launch a working Question Answering model. The inputs to the model would be a .txt file/text entered through key board, and a query. The model would then generate a wordcloud of the .txt file/text, generate a summary for the same, and using the file as the context, query as the question, generate an answer.
Many models were considered for Text summarizing. **google/pegasus-xsum** and some pretrained models from Happy Transformers were tried, but the time taken by these pretrained models for generating a summary was too long. So finally, the online platorm NLPCloud was selected for the task, which uses **facebook/bart-large-cnn** model.

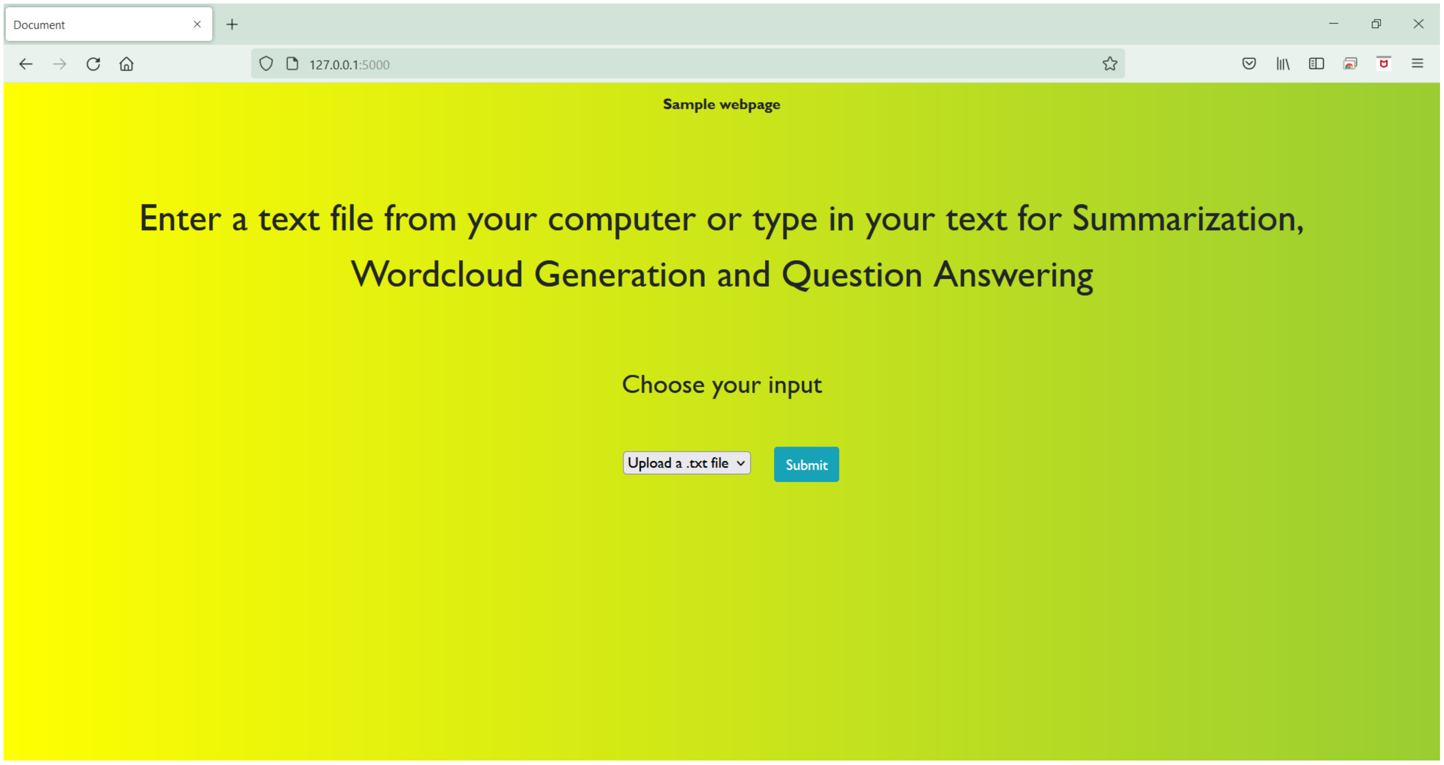
The problem being faced by the Question Answering task was that each model would take a maximum input size of 512 tokens, which is very small compared to the size of the .txt file. So, instead of making a random guess as to which part of the .txt file would be the appropriate context for the query entered by the user, the text file was divided into small strings of 512 tokens each. Then, each string was passed to the QA model as a context and an answer was generated. Then we sort the scores of these generated answers, and the best five answers were taken and displayed as the output in the API.
HappyQuestionAnswering was used for this task, which took a running time of 14 minutes to generate all the answers and return the best five possibilities.

Results

The final task was to build an Information Retrieval architecture by using **Huggingface transformers** and **Happy transformers** to implement a Question Answering system and launch it using a REST API.

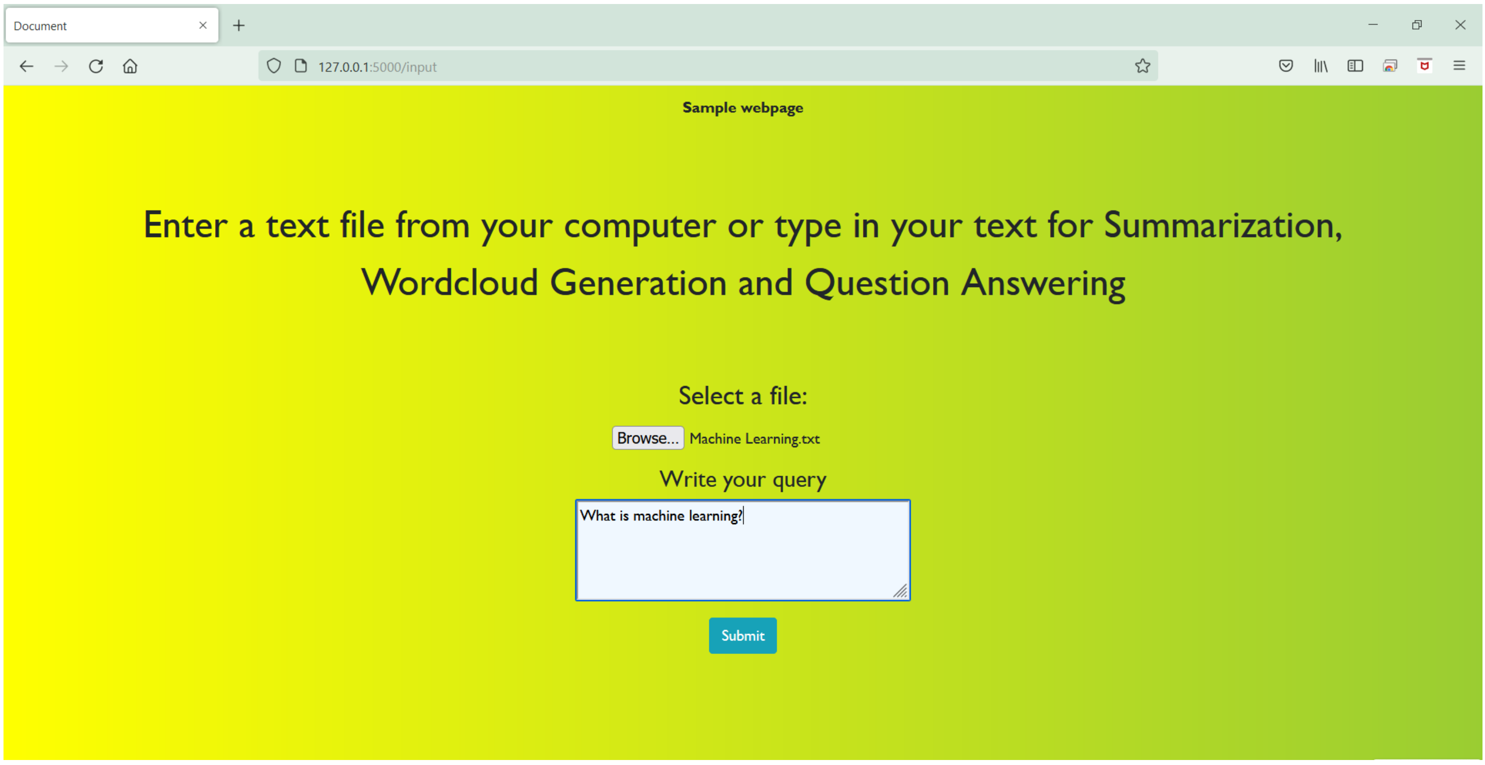
So, Happy transformers and NLPCloud were explored for this task. **Flask REST API** was also explored for launching the model as an API.
So, after downloading the sample text files, preprocessing the text, ad using the model for Question Answering, the script was transferred to PyCharm to make use of the FLASK API, where the models were first saved in the local machine to use them in the API. The script was then connected to the necessary HTML and CSS files for launching and redirecting the API to the web-pages made for this model.

The model works as such: after inputting a text file and a query string, the model would generate a Worcloud of the text file, a summary of the text file, and using the text file as context and the query vector as the question, generate an answer.
A sample example of the REST API can be seen here below:



Enter a text file from your computer or type in your text for Summarization, Wordcloud Generation and Question Answering

Choose your input



Document

127.0.0.1:5000/input

Sample webpage

Enter a text file from your computer or type in your text for Summarization, Wordcloud Generation and Question Answering

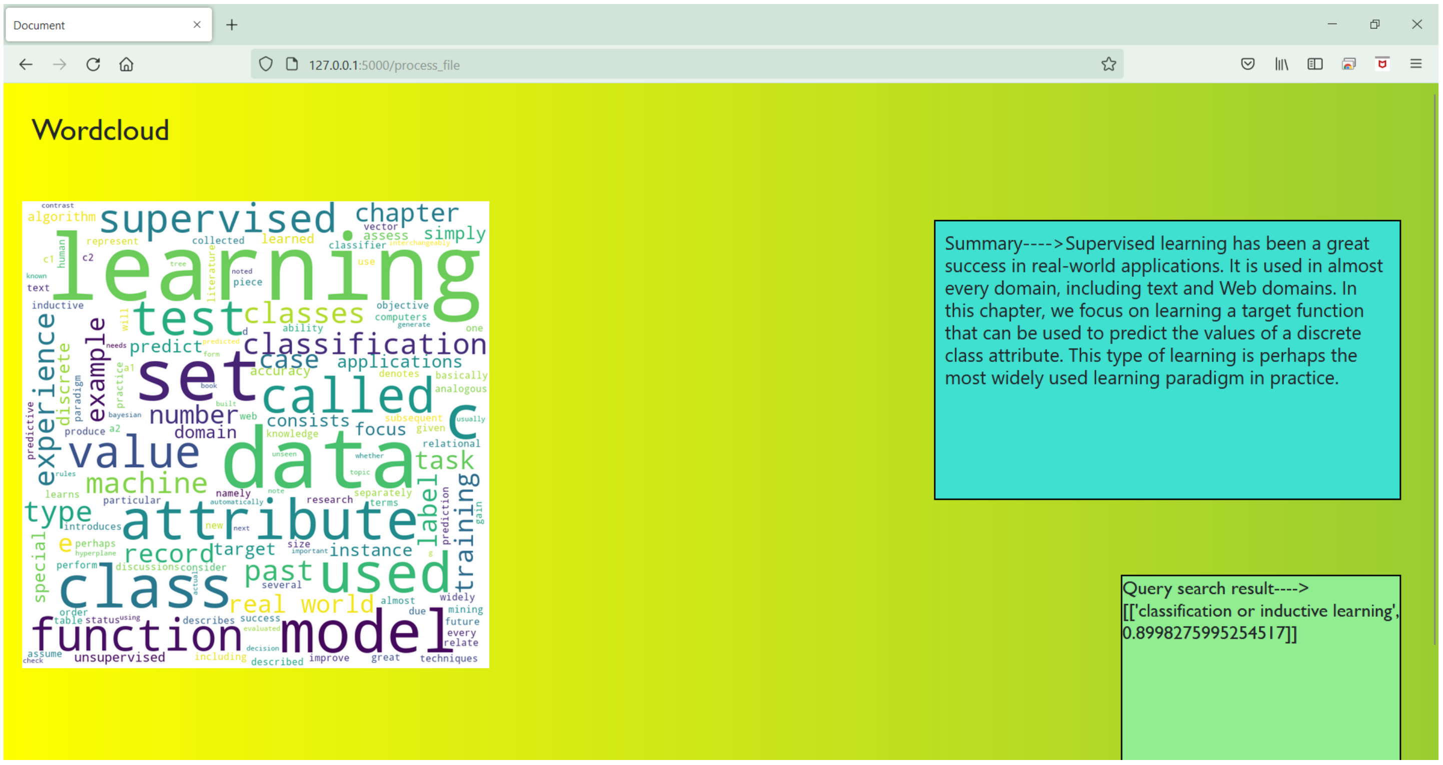
Select a file:

Browse... Machine Learning.txt

Write your query

What is machine learning?

Submit



Document

127.0.0.1:5000/process_file

Wordcloud

Summary---->Supervised learning has been a great success in real-world applications. It is used in almost every domain, including text and Web domains. In this chapter, we focus on learning a target function that can be used to predict the values of a discrete class attribute. This type of learning is perhaps the most widely used learning paradigm in practice.

Query search result---->
[['classification or inductive learning', 0.8998275995254517]]

Conclusions

It's important to note here that the REST API took about 15 minutes to generate the outputs, which is a very long time for any Information retrieval system. This happened because of the limitations of my machine, and it's limited Hard disk and RAM.
In order to improve on it, I used the multiprocessing library to distribute the task over the number of CPUs available on my local machine, which reduced the time considerably.