CLASSIFYING MULTI PAGE DOCUMENTS – TEXT AND EMAILS

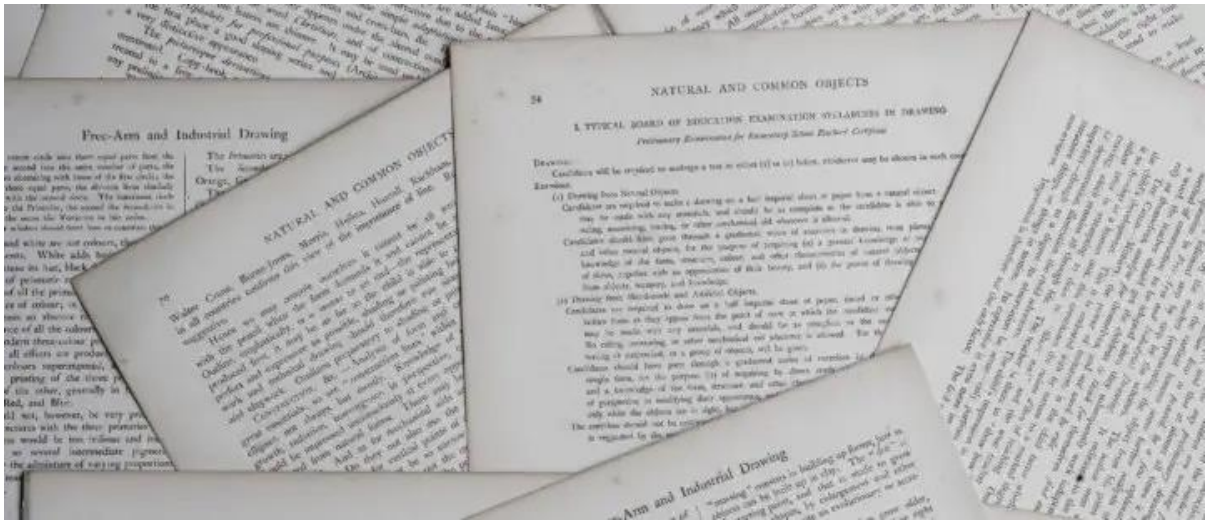
GROUP 11

Anupam Acharya

PCMS Venkatasastry Narasimhadevara

Suresh SV

Vemula Vamshi Krishna



2A Classifying multi-page documents - Text

Problem Statement: This Project involves 2-problem statements:

Problem Statement-2A:

Text is a rich source of information but extracting insights from text is challenging and time-consuming due to the unstructured data. With the growing scale of text data in industrial applications, automatic text classification is becoming increasingly important. Also separating information from the irrelevant can reduce the cost and time of searching and retrieving information from all the documents. In this project, we are aiming to build a model which can categorize the documents to make them easier to manage and sort.

Business Use Case

In the real world, we see a lot of industries where they use massive paper-based manuals, procedures, etc. For example, in industrial plants they have a lot of documents in paper format, scanned documents, word files and intelligent PDFs. It is a massive effort to classify them into drawings, standard operating procedures, manuals etc. But with a Classification of Multi-Page document algorithm, it can be done in a very small amount of time. The same solution can be implemented in hospitals, loan applications, etc.

Data and Data Pre-Processing

In the process of developing a document classification algorithm, the first and an important step is data preparation and pre-processing as the accuracy of the algorithm will heavily depend on the input the model gets.

BBC news website collection of stories from five topical areas

- Consists of 2225 documents from the BBC news website for the years 2004 and 2005
- Class labels are Business, Entertainment, Politics, Sport, and Tech
- All documents are .txt files and it doesn't contain any scanned documents. Therefore, there will be no need for an OCR to run as all the text is readable.

We have below text pre-processing algorithms to identify the features.

1. Count Vectorizer
2. TFIDF Vectorizer

Count Vectorizer:

Count Vectorizer is a great tool provided by the scikit-learn library in Python. It is used to transform a given text into a vector based on the frequency of each word that occurs in the entire text. This is helpful when we have multiple such texts, and we wish to convert each word in each text into vectors.

TFIDF Vectorizer:

TF-IDF stands for Term Frequency Inverse Document Frequency of records. It can be defined as the calculation of how relevant a word in a series or corpus is to a text. The meaning increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the dataset.

Also, we can use below methods to do the data transform before we generate features using above algorithms.

- Stop Word Removal is not necessary as TFIDF automatically takes care of it by assigning low weights to repeated articles and stop words. With Count Vectorizer we will pass it as a parameter.
- Lemmatization
- Stemming
- LSTM
- GRU
- BERT

Model and Implementation

In terms of Machine Learning, this problem is defined as a classification problem.
Following steps are planned for building the ML Classifier:

- Define the classes in which the documents must be classified.
- Import the data and view it.
- Pre-Process the Data.
- Divide the data into Train and Test
- Train the Model
    - By making use of any of below baseline models we can train and test for the better accuracy, then we can use that trained model to identify the respective class of each input text file.
        - Decision Tree Classifier
        - K-Neighbours Classification
        - SGD Classifier
        - Logistic Regression
        - SVC
- Test the Model
- Choose the best model using accuracy.

Proposed Approaches: The baseline methods defined above only run a model to provide an accuracy. They will provide a baseline to the entire exercise. We plan to use some advanced approaches to ensure we can run low-cost models without compromising on Accuracy. Since we will be dealing with large amount of textual data for the model, we plan to use LSTM. Since, LSTM has features to memorize the sequence of the data and to work on the elimination of unused information, it can be extremely helpful to optimise on the model running time and cost. This makes LSTM a powerful tool for performing text classification which we intent to leverage.

Conclusion

Texts are structured data, therefore we will use basic pre-processing and run the classification models. The best fit model (ensuring no underfitting or overfitting) based on accuracy of test data will be used as the final model to classify multi-page documents for text.

2B Classifying multi-page documents – Email Messages

Problem Statement-2B:

Automated classification of email messages into user-specific folders and information extraction from chronologically ordered email streams have become interesting areas in text learning research. In this project, we focus on the problem of assigning messages to a user's folders based on the fields such as "From", "Subject", "Body", and "To, CC".

Business Use Case

While the earlier use cases for this problem statement focussed on identifying spam and non-spam, today the classification can be further broken down into categories. One of the major business use cases is personal email boxes. With classification running on the background, it can identify important, financial, sports, food, category, advertisements, promotions, etc. and label them so that with the limited time we have. It can also add a lot of value to banks where they receive a lot of emails daily. This algorithm can easily triage the emails received and break it down into categories where each category can be seen by dedicated individuals.

Data and Data Pre-Processing

In the process of developing a document classification algorithm, the first and an important step is data preparation and pre-processing as the accuracy of the algorithm will heavily depend on the input the model gets.

- The Enron email dataset contains approximately 500000 emails generated by employees of the Enron Corporation.
- It was obtained by the Federal Energy Regulatory Commission during its investigation of Enron's collapse.
- This is the May 7, 2015 version of the dataset, as published at https://www.cs.cmu.edu/~./enron/

We have below text pre-processing algorithms to identify the features.

1. Count Vectorizer
2. TFIDF Vectorizer

Count Vectorizer:

Count Vectorizer is a great tool provided by the scikit-learn library in Python. It is used to transform a given text into a vector based on the frequency of each word that occurs in the entire text. This is helpful when we have multiple such texts, and we wish to convert each word in each text into vectors.

TFIDF Vectorizer:

TF-IDF stands for Term Frequency Inverse Document Frequency of records. It can be defined as the calculation of how relevant a word in a series or corpus is to a text. The meaning increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the dataset.

Also, we will use below methods to do the data pro-processing before we generate features using above algorithms.

- Stop Word Removal
- Lemmatization
- Stemming
- LSTM
- GRU
- BERT

Model and Implementation

In terms of Machine Learning, this problem is defined as a classification problem.
Following steps are planned for building the ML Classifier:

- Define the classes in which the documents must be classified.
- Import the data and view it.
- Pre-Process the Data.
- Divide the data into Train and Test
- Train the Model
    - By making use of any of below baseline models we can train and test for the better accuracy, then we can use that trained model to identify the respective class of each input text file.
        - Decision Tree Classifier
        - K-Neighbours Classification
        - SGD Classifier
        - Logistic Regression
        - SVC
- Test the Model
- Choose the best model using accuracy.

Proposed Approaches: The baseline methods defined above only run a model to provide an accuracy. They will provide a baseline to the entire exercise. We plan to use some advanced approaches to ensure we can run low-cost models without compromising on Accuracy. Since we will be dealing with large amount of textual data for the model, we plan to use LSTM. Since, LSTM has features to memorize the sequence of the data and to work on the elimination of unused information, it can be extremely helpful to optimise on the model running time and cost. This makes LSTM a powerful tool for performing text classification which we intent to leverage.

Conclusion

Emails, being slightly informal and unstructured will go through heavier pre-processing and scrutiny before running the models for better results. The best fit model (ensuring no underfitting or overfitting) based on accuracy of test data will be used as the final model to classify multi-page documents for emails.

Citations and References for 2a and 2b:

- https://towardsdatascience.com/multi-page-document-classification-using-machine-learning-and-nlp-ba6151405c03
- https://towardsdatascience.com/multi-class-text-classification-with-lstm-1590bee1bd17
- https://ieeexplore.ieee.org/abstract/document/8260658
- https://paperswithcode.com/task/document-classification