Enron Email Analysis

Soham Talukdar

***Abstract*—As a part of interview process for Vlabs, I take this opportunity to present this as a personal challenge to get an early impression on understanding NLP. Code for this paper can be found here: https://github.com/sohamtalukdar/Enron-Email-Analysis**

NB: This paper shouldn't be considered as a part of academic literature.

***Index Terms*—NLP, Enron**

I. Introduction

The Enron email dataset is a collection of approximately half a million emails generated by employees of the Enron Corporation, one of the largest energy companies in the United States, during the period of 1998 to 2002. The emails were made public as part of the investigation into the company's financial scandals, which ultimately led to its collapse in 2001. The dataset has become a popular resource for researchers and data scientists due to its size and the insights it provides into the inner workings of a major corporation and the individuals who worked there. The Enron email dataset provides a unique opportunity to study corporate communication patterns and can be used for various NLP tasks such as sentiment analysis, text classification, and email topic modeling. For our purpose we will be using "A subset of about 1700 labelled email messages".

II. Data Preparation

The process described in this section is a crucial step in the pre-processing pipeline for a text classification problem. The goal is to gather information from text files stored in multiple folders and consolidate it into a single Pandas dataframe. To start, a dictionary named "labels" is defined, which maps the numerical folder names to corresponding label values. The numerical folder names, ranging from 1 to 8, indicate different categories of messages, while the "labels" dictionary translates these numbers into human-readable names. The root directory is set as the location where the folders containing the text files are stored. An empty list named "rows" is created to store the information extracted from each text file. The procedure then iterates over all folder names in the root directory and only continues if the path leads to a directory. If it does, the label value of the folder is retrieved from the "labels" dictionary. For each folder, the procedure goes through all filenames and only considers files ending with ".txt". The contents of each file are opened, read, and the message body is extracted by dividing the contents by "\n\n" and taking the last item from the resulting list. The number from the filename is also extracted and added to a dictionary which is then added to the "rows" list, along with the label and message body. Finally, the information extracted from each text file is consolidated into a Pandas dataframe named "df" through the "rows" list. The dataframe consists of three columns - "#", "Label", and "Message" - representing the number, label, and message body of each text file, respectively.

The next phase of our data pre-processing pipeline focuses on cleaning and preparing the "Message" column in our dataframe "df" for further analysis and modelling. This is a crucial step that involves several stages of processing to ensure the data is in optimal condition. The first stage is to remove all non-word characters and digits from the messages. This is done by using a regular expression in a lambda function applied to the "Message" column, which removes all characters and digits that are not words or white spaces. The resulting strings are then converted to lowercase. Next, email addresses and URLs are removed from the messages. This is achieved by using another lambda function with regular expressions that specifically identify email addresses and URLs and remove them from the messages. The messages are then tokenized, stopwords are removed, and the remaining words are stemmed. This is done through another lambda function, which uses the Natural Language Toolkit (NLTK) to carry out these operations. To further refine the data, we remove rows with labels "Empty message (due to missing attachment)" or "Empty message". Additionally, any rows with an empty "Message" column are removed by using the "dropna" method, and all rows with an empty "Message" value are removed using boolean indexing(not under consideration for our use case). Finally, we ensure the dataframe only contains unique messages by removing duplicates using the "drop_duplicates" method. By carrying out these steps, we can be confident that our data is in optimal condition and ready for further analysis and modelling.

III. Model Implementation

The development of an email classification model is a multi-step process, beginning with text data pre-processing. This step is essential in transforming the raw text data into a numerical representation that can be fed into machine learning algorithms. A widely used method for this purpose is the Tf-IDF (Term Frequency-Inverse Document Frequency) technique, which can be implemented using TfidfVectorizer. In NLP, effective representation of text data is a crucial challenge. The goal is to convert text data into a numerical format that can be effectively processed by machine learning algorithms to make predictions. To achieve this, there are a variety of techniques available, including Tf-Idf, CBOW (Continuous Bag-of-Words), and Skip-Gram. TF-IDF is a method of representing text data by calculating the importance of each word in a document based on its term frequency (the number of times it appears in the document) and inverse document frequency (the rarity of the word across all documents). While TF-IDF provides a compact representation of text data and emphasizes the importance of each word, it does not capture the context in which words appear, which can limit its accuracy in some NLP tasks. CBOW and Skip-Gram, on the other hand, are word embedding techniques that capture the semantic and syntactic relationships between words. CBOW predicts the target word based on its surrounding context words, while Skip-Gram predicts the surrounding context words based on the target word. These models provide a dense and more informative representation of words and are well suited for NLP tasks such as email classification. When choosing between these text representation techniques, the size of the dataset, available computational resources, and desired level of detail in the representation should be considered. CBOW or Skip-Gram may be preferred for large datasets, while TF-IDF may be a better option for smaller datasets or limited computational resources. The type of problem being solved, such as binary or multi-class classification, should also be considered.

In the code provided, the Word2Vec model is trained on the messages column of the dataframe using the CBOW architecture. After the model is trained, the messages in the train and test sets are converted into average word vectors. This is done by looping through each message, summing up the word vectors for each word in the message, and then dividing the sum by the number of words to get the average word vector for the message. Finally, the average word vectors for the train and test messages are stored in the train_vectors and test_vectors arrays, respectively. These arrays can now be used as features for training a machine learning model to classify emails into different categories based on the content of the emails.

Multinomial Naive Bayes (MultinomialNB) is a simple probabilistic classifier that makes predictions based on the presence or absence of certain words in a text document. It is fast and effective for text classification tasks, and works well when the underlying text data follows a multinomial distribution.

Support Vector Machines (SVM) is a powerful algorithm for classification problems. It uses linear separation to determine the boundary between different classes. The algorithm maximizes the margin, or distance, between the boundary and the closest data points from each class to achieve maximum separation. SVC, or Support Vector Classifier, is a type of SVM that uses a linear kernel and is particularly useful for text classification tasks. This is because Tf-IDF representation of text data provides numerical features that can be used as inputs for the SVC model. Although SVC is effective for small to medium-sized datasets with well-separated classes, it may struggle with non-linear separations and large datasets with many features.

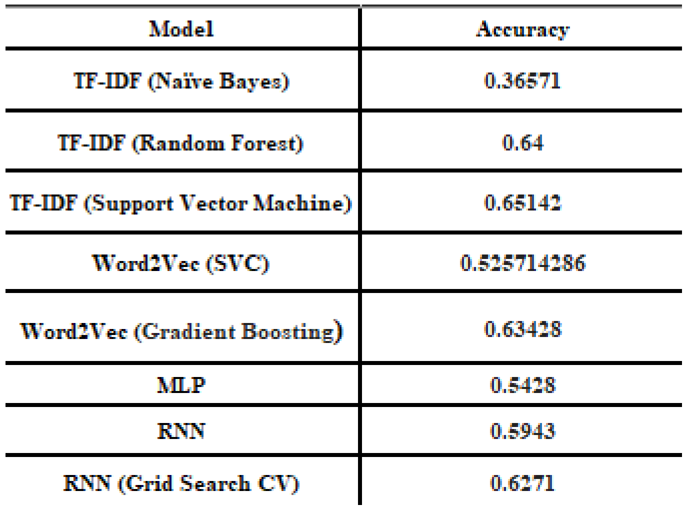
Gradient Boosting is an ensemble method that trains multiple weak learners and combines them to produce a strong learner. It is widely used for binary and multi-class classification problems and is known to perform well on complex, non-linear datasets. However, gradient boosting requires a significant number of computational resources and requires careful hyperparameter tuning to achieve good results.

Random Forest is an ensemble learning method that builds multiple decision trees and combines their predictions to make the final prediction. It is a powerful and versatile model that is able to handle a large number of features and complex decision boundaries. In the context of email classification, Random Forest can be used to identify the most important words in the text data and make predictions based on their presence or absence.

Multilayer Perceptron (MLP) is a type of neural network that can learn complex non-linear relationships between the input and output variables. This makes it well-suited for large datasets with a large number of features and it is widely used for binary and multi-class classification problems. However, MLP requires a significant amount of data to train effectively and can be computationally intensive.

Recurrent Neural Networks (RNNs) are a popular type of neural network used in NLP tasks such as email classification. RNNs are designed to handle sequences of varying length, allowing them to capture the sequential information and relationships between words in a text document. This ability makes them well-suited for accurately classifying emails based on context. To further improve the performance of an RNN model, hyperparameter tuning using GridSearchCV can be employed. This method automates the process of searching through a set of predefined hyperparameters, training the model with each set, and selecting the optimal set that results in the best performance. It is important to have a solid understanding of the mathematical principles behind RNNs before implementing them. These principles include the use of sequential information to model the relationships between words and the calculation of the hidden state of the RNN at each time step, based on the current input and the previous hidden state.

IV. Result



Model	Accuracy
TF-IDF (Naïve Bayes)	0.36571
TF-IDF (Random Forest)	0.64
TF-IDF (Support Vector Machine)	0.65142
Word2Vec (SVC)	0.525714286
Word2Vec (Gradient Boosting)	0.63428
MLP	0.5428
RNN	0.5943
RNN (Grid Search CV)	0.6271

V. Future Improvements

Based on the result, further improvement can be done with more data or making relevant changes to the pre-processing of it. After using Recurrent Neural Network (RNN) for email classification based on context, exploring other advanced NLP techniques, such as Named Entity Recognition (NER), Part-of-Speech (POS) tagging, Sentiment Analysis, and Topic Modeling, can provide further opportunities to enhance email classification. Transfer learning with pre-trained models such as BERT or ELMo can also be applied to improve performance. These techniques extract information such as entities, word roles, sentiment, and themes to create features for the classification model.