Email Classification

Objective: Build an AI model to classify emails into different categories based on the content of the emails

Data: "A subset of about 1700 labelled email messages" (Link)

Procedure:

- Pre-process and clean the data
- Build a model to categorize them into the following classes

  a) Company Business, Strategy, etc. (elaborate in Section 3 [Topics])

  b) Purely Personal

  c) Personal but in professional context (e.g., it was good working with you)

  d) Logistic Arrangements (meeting scheduling, technical support, etc)

  e) Employment arrangements (job seeking, hiring, recommendations, etc)

  f) Document editing/checking (collaboration)

- Train and test the model and report the performance

Instructions:

- Please use Python for the implementation
- You may use 80% of data for training and 20% for validation
- You may use opensource models, packages or libraries if needed.

About the Dataset:

- File Extraction: The File downloaded from the link is in "<u>.tar.gz</u>" format. Once extracted the file contains 8 folders and a categories.txt file.
- Each folder has a number of "<u>.txt</u>" files and their corresponding "<u>.cats</u>" files.
- Text files contain the email thread whereas the corresponding "<u>.cats</u>" contains the label of the corresponding "<u>.txt</u>"
- The format of the data contained in the "<u>.cats</u>" is explained in categories.txt file as:
  Format of each line in .cats file:
  n1, n2, n3
  n1 = top-level category
  n2 = second-level category
  n3 = frequency with which this category was assigned to this message
  Here are the categories:
  1. Coarse genre
     - 1.1 Company Business, Strategy, etc. (elaborate in Section 3 [Topics])
     - 1.2 Purely Personal
     - 1.3 Personal but in professional context (e.g., it was good working with you)
     - 1.4 Logistic Arrangements (meeting scheduling, technical support, etc)
     - 1.5 Employment arrangements (job seeking, hiring, recommendations, etc)
     - 1.6 Document editing/checking (collaboration)
- For the given question the top-level category n1 is :1 and the second level category n2 ranges from 1-6.
- For the given assignment I have considered the emails that are labelled with top class as 1 at least once.
- From the .cats files it was observed that: Top Category =1, Labels account for 34.1% (or 3096 of 9091) of category ratings.
  - Document editing/checking (collaboration): 302
  - Logistic Arrangements (meeting scheduling, technical support, etc): 884
  - Personal but in professional context (e.g., it was good working with you): 237
  - Company Business, Strategy, etc.): 1429
  - Employment arrangements (job seeking, hiring, recommendations, etc): 168
  - Purely Personal: 76

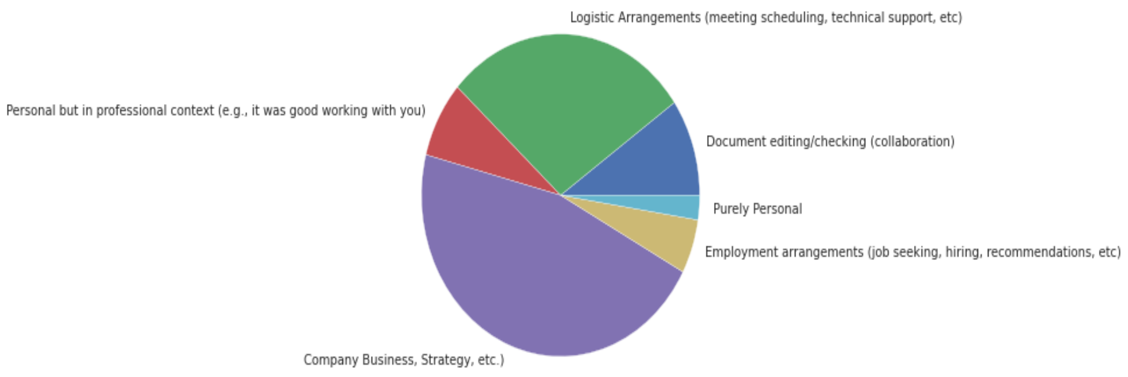


Fig1: Distribution of emails with different labels

Understanding the Data:

- The dataset consists of emails that are labelled by students of CMU students, each has one or more label. It was observed that the emails have more than one label in the class one.
- The text from the .txt files and the labels from their corresponding .cats files are extracted and put together for the further analysis.
- The number of emails vs number of labels plot shows that there are more than 150 emails that have more than one label.

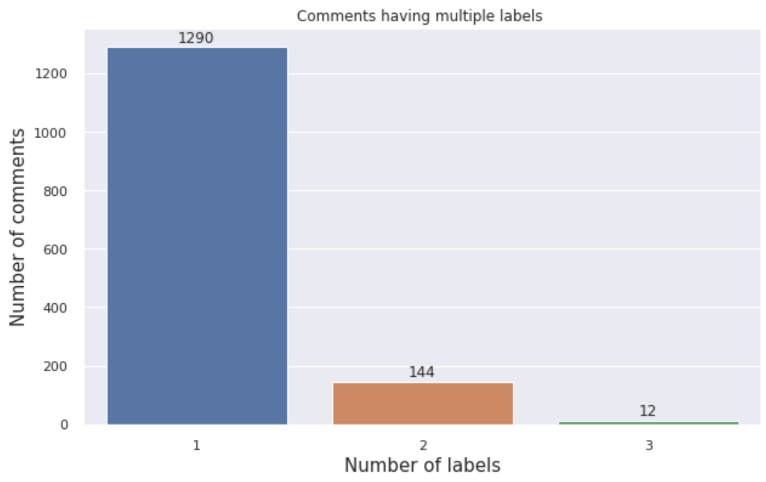
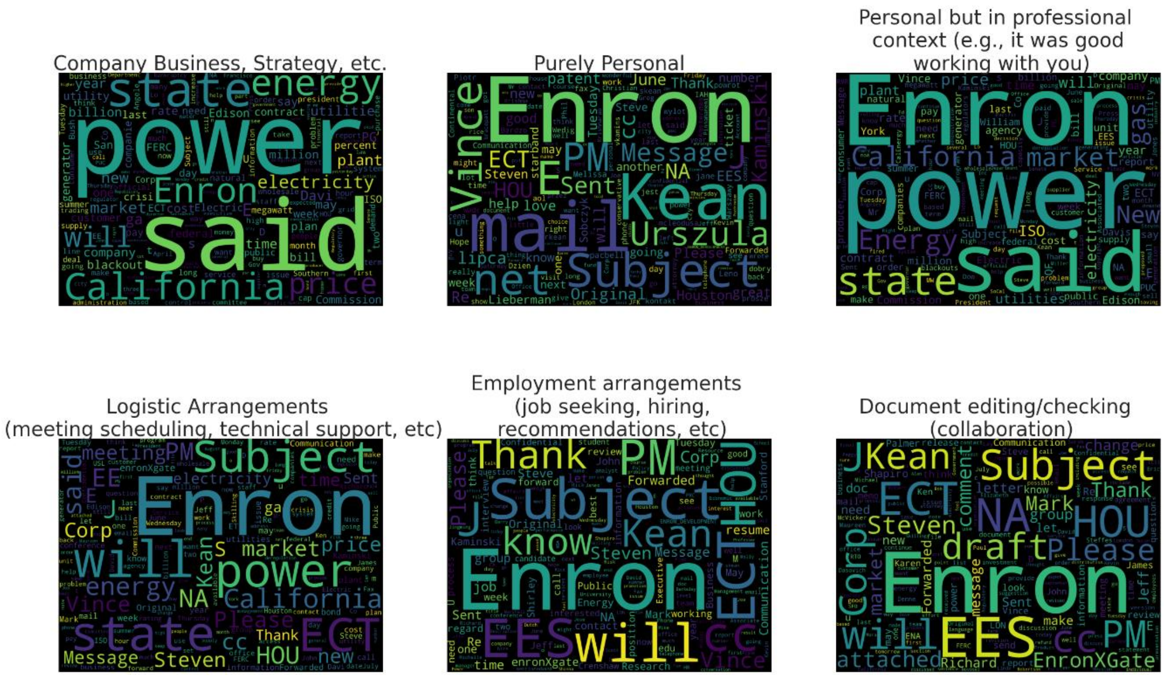


Fig2: Number of emails with different labels

- <u>Multi-label Classification:</u> It is a type of machine learning task in which an algorithm is trained to assign multiple labels or categories to an input instance. In contrast to traditional classification, where each input is assigned to a single predefined class, multi-label classification allows for multiple categories to be assigned to a single input instance. In this case we use multi-label classification to classify the emails.
- The below figure shows the most repeated words for the 6 labels are shown in the form of word cloud.



Data Pre-processing:

The following steps are followed for cleaning the data.

- Tokenization: The process of splitting the text into individual words or tokens.
- Lowercasing: Converting all text to lowercase can help reduce the number of unique words in the dataset and simplify the learning process for machine learning algorithms.
- Stop word removal: Stop words are common words in a language that do not carry much meaning, such as "the" or "and." Removing these words can help reduce the size of the dataset and improve the quality of the remaining words.
- Stemming and lemmatization: These are techniques used to reduce words to their base or root form, which can help reduce the number of unique words in the dataset and improve the accuracy of machine learning algorithms.
- Removing punctuation and special characters: This involves removing characters that do not add much meaning to the text, such as punctuation marks and symbols.
- Removing numbers: Numbers may not be relevant to the analysis of text, so removing them can help improve the quality of the remaining words.

Bag of Words (BoW) is a natural language processing (NLP) technique that is used to represent text data as numerical features that can be used for machine learning tasks. The basic idea behind BoW is to consider a document as a collection of words, without considering the order of the words or the context in which they are used. The resulting representation of the document is a "bag" of words, where each word in the document is represented by a count of how many times it appears.

Once the data is cleaned, I have tried to apply different type of classification models. The accuracy and loss of the models is shown in the following table:

Model	Accuracy	Loss
Logistic Regression (OneVsRest)	44%	13.19
KNN Classifier	41%	14.78
Random Forest Classifier	48%	11.15
MLP Classifier	47%	11.57

From the above table we can see that the accuracy of the models is low and the loss is high. This accuracy can be improved and the loss can be reduced by applying neural networks.

In the next step I have tried to implement LSTM, the loss has reduced to 5.14.

Classification Using Pretrained BERT models:

BERT stands for "Bidirectional Encoder Representations from Transformers". It is a pre-trained language model developed by Google in 2018 that uses the transformer architecture for natural language processing (NLP) tasks.

I initialized XLMForMultiLabelSequenceClassification from the checkpoint of a model trained on another task or with another architecture that is initializing a BertForSequenceClassification model from a BertForPreTraining model. Error has been further reduced.

The codes for the models:

- Enron_Email_classification_ml_models
- Pre_Trained_BERT_Email_Classification