#### Task 1:

Intent detection on Enron email set. We define "intent" here to correspond primarily to the categories "request" and "propose".

In some cases, we also apply the positive label to some sentences from the "commit" category if they contain datetime, which makes them useful. Detecting the presence of intent in email is useful in many applications, e.g., machine mediation between human and email. The dataset contains parsed sentences from the email along with their intent (either 'yes' or 'no').

You need to build a learning model which detects whether a given sentence has intent or not. Its a 2-class classification problem.

Find the train and test dataset in **enron.zip.** 

Please do not hard code any values. Be prepared to explain your choice of classifier.

Although its not required you can refer this paper for more information on the dataset : Cohen, William W., Vitor R. Carvalho, and Tom M. Mitchell. "Learning to Classify Email into``Speech Acts"." EMNLP. 2004.

Tip: Try to add feature engineering into your model. Simple baseline with logistic regression gives 71% accuracy.

## **SOLUTION**

- 1. Importing the data
- **2.** Finding the features that can be used and extracted from the data.
  - **a.** Count characters, unique words, average sentence length, count mentions
  - b. remove punctuations
  - c. count capitals and average of stop words and words
- 3. Vectorize the features using **TFIDF** and create the train dataframe.
- 4. Train with Random Forest Classifier with target names "Yes" or "No"
- 5. Test the results on test data obtained accuracy of 87.11 %

Accuracy	=>	87.11			
Random Forest Classifier results:					
		precision	recall	f1-score	support
	Yes	0.86	0.92	0.89	474
	No	0.89	0.81	0.85	387
accur	acy			0.87	861
macro	avg	0.87	0.87	0.87	861
weighted	avg	0.87	0.87	0.87	861

Figure1: Classification report

Observe the table below from few rows of train data with features.

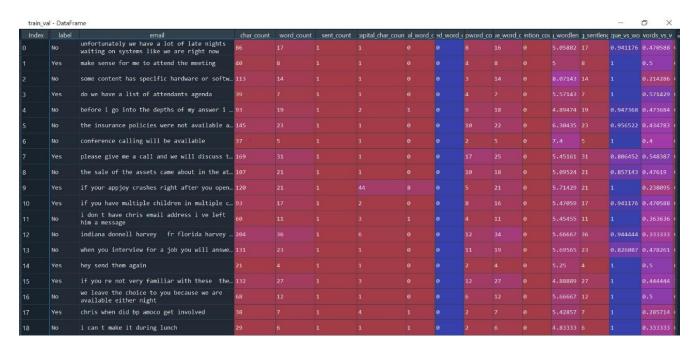


Figure 2: Train Dataframe with features

#### Task2:

This is an information extraction exercise from financial textual data.

Your task is to write a scalable Information Extraction algorithm to extract datapoints from the text and compute the results. The program will be evaluated on surprise test data.

From the text we need to extract the following:

- Currency
- Amount
- Rounding (up / down/ nearest) If not mentioned, take it as 'nearest' by default.

for 'Delivery Amount' and 'Return Amount'.

We have included the textual data in **isda\_data.json**. Refer **sample\_isda.py** for the basic implementation framework.

**Input:** "Rounding. The Delivery Amount and the Return Amount will be rounded to the nearest integral multiple of EUR 100,000; provided that if an amount corresponds to the exact half of such multiple, then it will be rounded up; and provided further that, for the purpose of the calculation of the Return Amount where a party's Credit Support Amount is, or is deemed to be, zero, the Return Amount shall not be rounded."

## **Output:**

```
{
"delivery_currency": "EUR",
"delivery_amount": "100,000",
"delivery_rounding": "nearest",
"return_currency": "EUR",
"return_amount": "100,000",
"return_rounding": "nearest"
}
```

### **SOLUTION:**

- 1. Load the textual data isda\_data.json
- 2. Create the train data
  - a. Collect all the currency entities
  - b. Collect all the rounding entities
- 3. Annotate all the entities in the data at their character positions, extract them.
- 4. Add to the train data
- 5. Train Named Entity Recognition
- 6. Test the trained model.

E.g: Text = "I have to deliver the amount of the nearest to EUR 10,000"

# **Extracted intenties:**

```
Entities [('EUR', 'currency'), ('10,000', 'amount')]
Entities [('EUR', 'currency'), ('10,000', 'amount')]
```