Multi-Class Email Text Classification

Predicting the label/folder of an email with the metadata and content of an email.

Pedro Jr. Vicente Valdez
Arjun Patel
Rajan Dutta
June Namgung

Introduction

- For those who are committed to inbox zero
- Labeling each email and finding emails with a certain label can be tedious
- A solution to automatically organize an incoming email to its right:
  - Folder
  - Label
  - Category
- How It Works:
  - User chooses the folders that they want to have
  - We use our trained model with all the labels that are options for the user
  - An incoming email will be grouped into one of the chosen labels

Dataset - Enron Email Dataset

- It contains 517401 rows corresponding to emails generated by employees at the Enron Corporation, mostly senior management.
- Originally published by the Federal Energy Regulatory Commission during the investigation of Enron's collapse in December of 2001.
- Two Columns: File and Message

```
Message-ID: <15464986.1075855378456.JavaMail.evans@thyme>
Date: Fri, 4 May 2001 13:51:00 -0700 (PDT)
From: phillip.allen@enron.com
To: john.lavorato@enron.com
Subject: Re:
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-From: Phillip K Allen
X-To: John J Lavorato <John J Lavorato/ENRON@enronXgate@EN
X-cc:
X-bcc:
X-Folder: \Phillip_Allen_Jan2002_1\Allen, Phillip K.\'Sent
X-Origin: Allen-P
X-FileName: pallen (Non-Privileged).pst

Traveling to have a business meeting takes the fun out of

As far as the business meetings, I think it would be more

My suggestion for where to go is Austin.  Play golf and re
```

Research Questions - Goal

Questions

- What do we need to successfully build an email classification tool?
- What kind of preprocessing and feature engineering is useful and adequate for text classification problems?
- What type of models can we use?
- Are there some models that perform better in text classification problems?
- How do we improve and/or optimize said model?

Goal

- Build a Multi-Class Email Text Classifier Model initially targeted toward companies/enterprises.

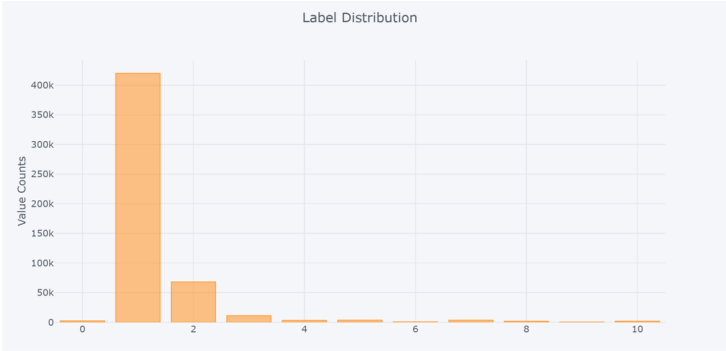
Independent Variables

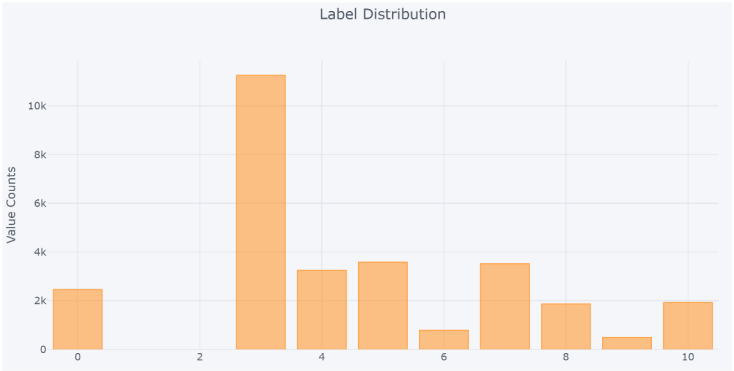
- Embedded message
  - Convert each email message to a 300-length vector embedding
- Message characteristics
  - Message length
- Email characteristics
  - Day of week (as one-hot-encoding)
  - Day of week (mapped value)
  - Contains CC or BCC
- Occurrences of email labels in body

Index	Message Length	Day= Sunday	...	Day= Saturday	Embedded vector, Dimension 1	...	Embedded vector, Dimension 300

Dependent Variable: Folder Labels

- Original dataset included 1,700+ folders
- Using KMeans clustering and some manual edits, we create 11 clusters
- Most emails in ‘Sent’ or ‘Documents’ or similar (labels 1 and 2) so we exclude those labels and train only with labels 0 and 3-10





Models

Decision Tree, Random Forest, Bagging, LSTM

Decision Tree

Decision Tree

- Attempted with different feature sets
- Most effective with using gini index to find best splits
- Max_depth altering got crazy when increasing too much
  - Likewise too small didn't allow the visualization to split based on the entire embedding of the body
  - Min_samples_split
  - Max_leaf_nodes
- Hit a plateau even after tuning the model

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=630,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=3000, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=100,
            splitter='best')
```

Random Forest

Random Forest

- Attempt to beat the decision tree model
- Still plateau in performance at about ~40% test accuracy
- The hyperparameters such as max-depth didn't quite help
- Perhaps needed better feature engineering

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=10, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
                       oob_score=False, random_state=0, verbose=0, warm_start=False)
```

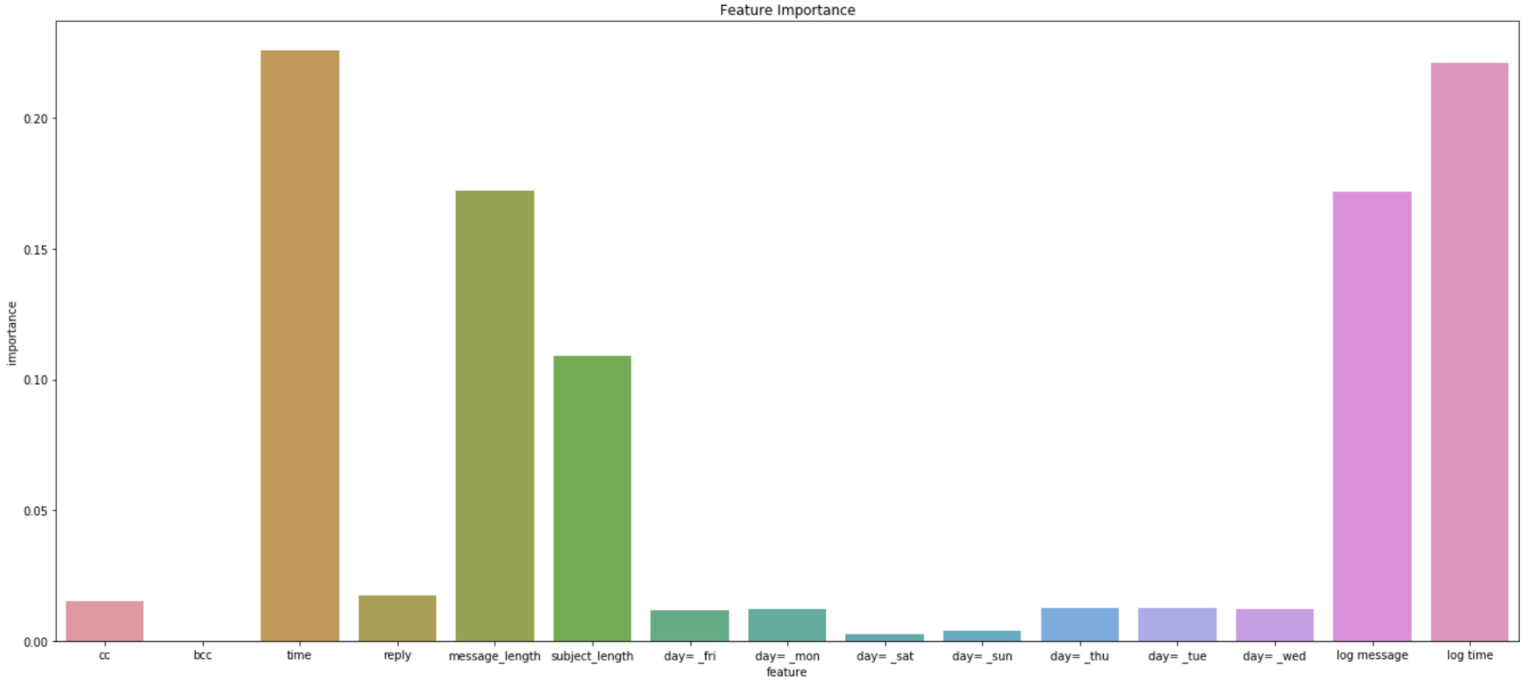
Random Forest

- Attempt to beat the decision tree model
- Still plateau in performance at about ~40% test accuracy
- The hyperparameters such as max-depth didn't quite help
- Perhaps needed better feature engineering

```
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
# Y_train_pred = rf.predict(X_train)
# Y_test_pred_rf = rf.predict(X_test)
print("Accuracy Score of Random Forests on test set",rf.score(X_test, y_test)*100,'%')
# accuracy_score(y_test, Y_test_pred)
```

```
Accuracy Score of Random Forests on test set 35.04144086707045 %
```

Random Forest



Bagging

Bagging

- Still plateau in performance at about ~40% test accuracy even with some more feature engineering e.g. stopwords

```
BaggingClassifier(base_estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'),
         bootstrap=True, bootstrap_features=False, max_features=0.5,
         max_samples=0.5, n_estimators=100, n_jobs=None, oob_score=False,
         random_state=42, verbose=0, warm_start=False)
```

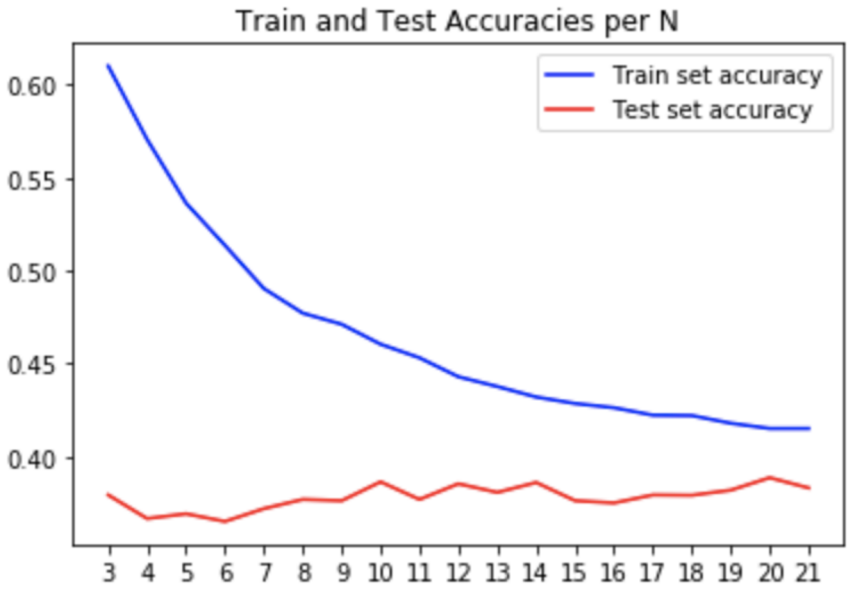
```
print("Accuracy Score of Bagging on train set", bag.score(X_train, y_train)*100,'%')
print("Accuracy Score of Bagging on test set",bag.score(X_test, y_test)*100,'%')
```

```
Accuracy Score of Bagging on train set 93.13112682310346 %
Accuracy Score of Bagging on test set 37.097545425565826 %
```

KNN

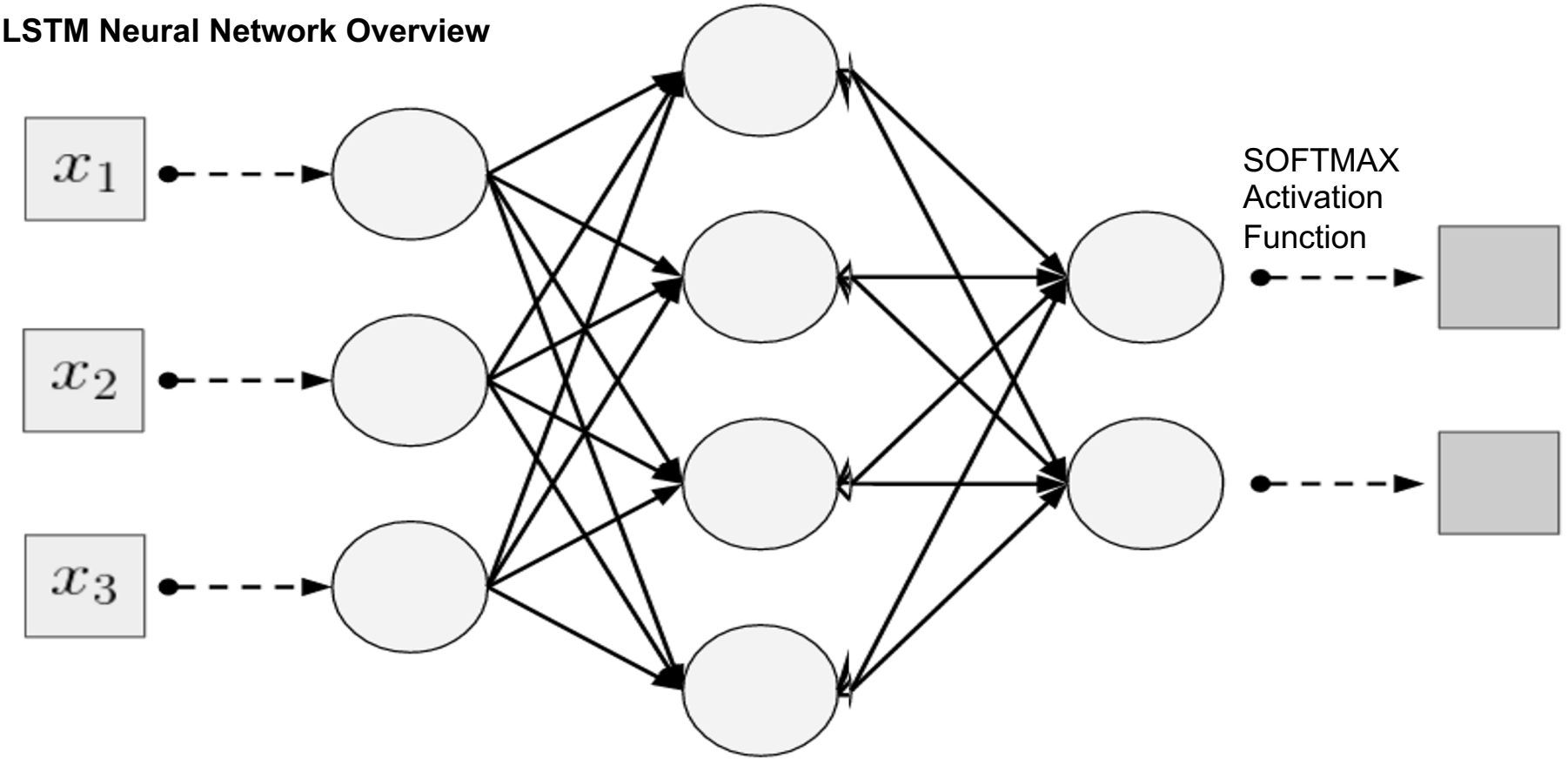
KNN Model

- Experimented with various values for the number of neighbors to look at
- Found that n = 21 performed the best (least overfitting)
- Overall, very weak performance from the KNN model
    - The data is too complex and the clusters aren't distinct enough for KNN to succeed



Long Short Term Memory Neural Network (LSTM-NN)

LSTM Neural Network Overview



Input Values

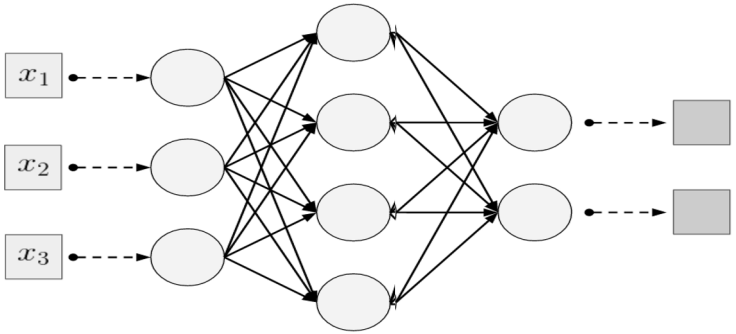
Embedding
Input Layer

LSTM (100-unit)
Hidden Layerr 2

11-Unit Dense
Output Layer

LSTN-NN

- Embedding Layer
  - (250 Vector Input)
- Spatial Dropout
- 100-unit LSTM Layer
- 11-unit Dense Output
  - Softmax Function



- Loss Metric:
  - Categorical Cross Entropy
- Grid Search:
  - Dropout: [0.2:0.5]
  - BatchSize: [50,100,150]
  - LSTM Unit: [50, 100, 150]
  - Optimizer: [Adam, SDG, Adagrad]

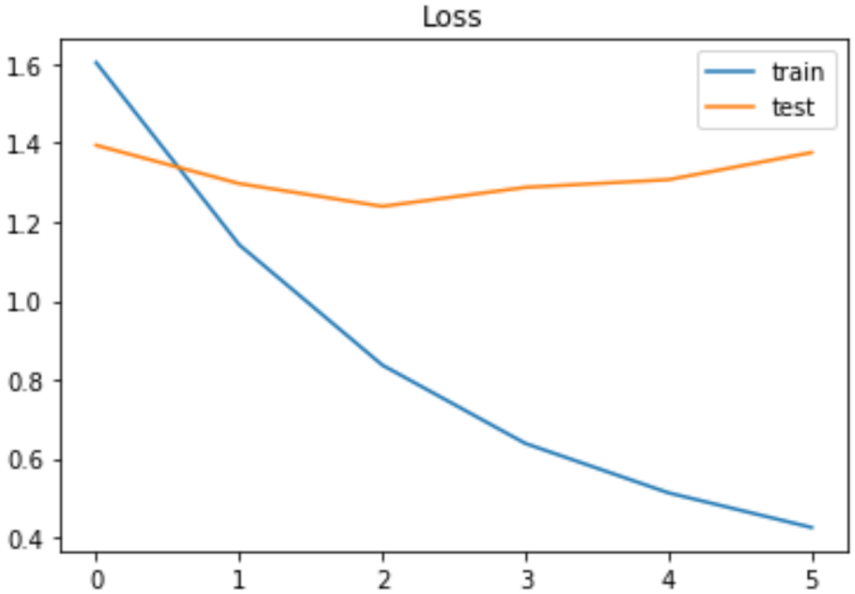
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_22 (Embedding)     (None, 250, 100)          100000000
_________________________________________________________________
spatial_dropout1d_21 (Spatia (None, 250, 100)          0
_________________________________________________________________
lstm_21 (LSTM)               (None, 100)               80400
_________________________________________________________________
dense_38 (Dense)             (None, 11)                1111
=================================================================
Total params: 100,081,511
Trainable params: 100,081,511
Non-trainable params: 0
_________________________________________________________________
None
```

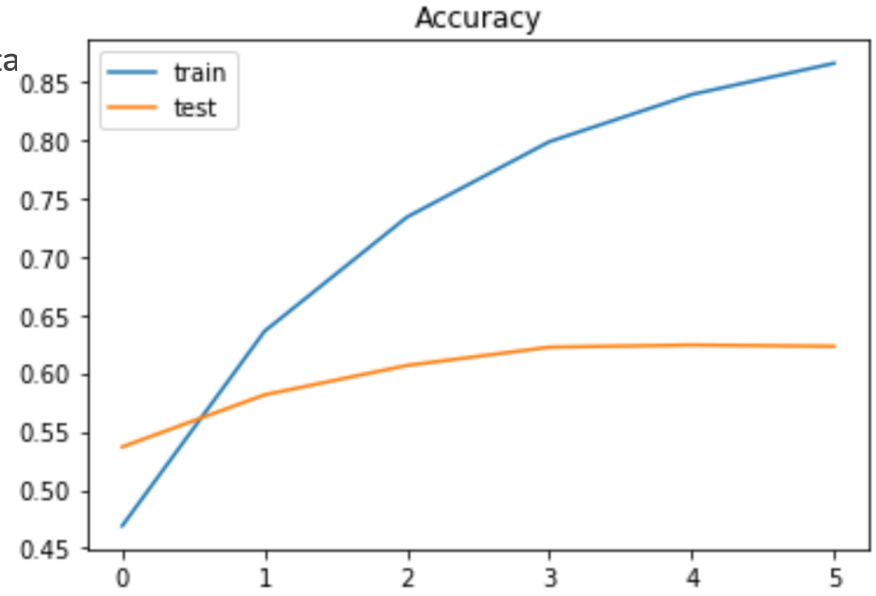
Alternative Approach (Failed)

Layer (type)	Output Shape	Param #	Connected to
input_19 (InputLayer)	(None, 250)	0	
embedding_23 (Embedding)	(None, 250, 100)	100000000	input_19[0][0]
input_20 (InputLayer)	(None, 8)	0	
spatial_dropout1d_22 (SpatialDr	(None, 250, 100)	0	embedding_23[0][0]
dense_40 (Dense)	(None, 64)	576	input_20[0][0]
lstm_22 (LSTM)	(None, 100)	80400	spatial_dropout1d_22[0][0]
dense_41 (Dense)	(None, 32)	2080	dense_40[0][0]
dense_39 (Dense)	(None, 11)	1111	lstm_22[0][0]
dense_42 (Dense)	(None, 11)	363	dense_41[0][0]
concatenate_5 (Concatenate)	(None, 22)	0	dense_39[0][0] dense_42[0][0]
dense_43 (Dense)	(None, 11)	253	concatenate_5[0][0]

Total params: 100,084,783
Trainable params: 100,084,783
Non-trainable params: 0

LSTM - Loss and Accuracy





Testing LSTM Model

```
new_complaint = ['What are the consequences of a lawsuit?']
seq = tokenizer.texts_to_sequences(new_complaint)
padded = pad_sequences(seq, maxlen=max_length)
pred = model.predict(padded)
labels = ['Weather/Natural','Sent Mail','Random/NA','Financial/Logistics','Related to Other People',
          'Places','Legal','Buisness','2-Letter/Random','Other Firms','HR/Recruiting/MBA']
print(pred, labels[np.argmax(pred)])
```

```
[[3.2427564e-02 4.2977501e-04 4.4141154e-04 8.4516507e-01 2.4327002e-02
  1.6769949e-02 7.1043829e-03 2.1948883e-02 3.2583706e-02 6.2899133e-03
  1.2512243e-02]] Financial/Logistics
```

```
new_complaint = ['Have you transferred the funds to their bank account?']
seq = tokenizer.texts_to_sequences(new_complaint)
padded = pad_sequences(seq, maxlen=max_length)
pred = model.predict(padded)
labels = ['Weather/Natural','Sent Mail','Random/NA','Financial/Logistics','Related to Other People',
          'Places','Legal','Buisness','2-Letter/Random','Other Firms','HR/Recruiting/MBA']
print(pred, labels[np.argmax(pred)])
```

```
[[0.25816587 0.00338857 0.00290285 0.19505277 0.03923525 0.07588409
  0.02688199 0.3290266  0.03238906 0.01407197 0.02300098]] Buisness
```

```
new_complaint = ['Thank you for contactig us regarding the MBA program, Do you have any questions?']
seq = tokenizer.texts_to_sequences(new_complaint)
padded = pad_sequences(seq, maxlen=max_length)
pred = model.predict(padded)
labels = ['Weather/Natural','Sent Mail','Random/NA','Financial/Logistics','Related to Other People',
          'Places','Legal','Buisness','2-Letter/Random','Other Firms','HR/Recruiting/MBA']
print(pred, labels[np.argmax(pred)])
```

```
[[0.01847702 0.00082748 0.00112296 0.10232089 0.13056819 0.00560034
  0.00132967 0.00850991 0.1549232  0.02847201 0.54784834]] HR/Recruiting/MBA
```

```
new_complaint = ['How will the climate be tomorrow? Can I have the weather forecast for tomorrow?']
seq = tokenizer.texts_to_sequences(new_complaint)
padded = pad_sequences(seq, maxlen=max_length)
pred = model.predict(padded)
labels = ['Weather/Natural','Sent Mail','Random/NA','Financial/Logistics','Related to Other People',
          'Places','Legal','Buisness','2-Letter/Random','Other Firms','HR/Recruiting/MBA']
print(pred, labels[np.argmax(pred)])
```

```
[[0.6322321  0.00068294 0.00081632 0.0568786  0.02822019 0.08666048
  0.00875696 0.01268484 0.10277729 0.04104653 0.02924373]] Weather/Natural
```

Conclusion

Conclusions and Future Work

Conclusions

- Traditional models such as KNN, Decision Tree, and Random Forest were ineffective due to data complexity
- LSTM performed the best because its ability to remember past information in an email allowed it to better classify

Future Work

- Expanding to Multi-Class Multi-Label Classification
- Try SVM models
- Making it more consumer friendly as a plugin
- Integration with Gmail or a third party client
- Deploy model with our own web app