Email Priority Sorting and Response Recommendation System

*

Sharath Narayana
Computer Science And Engineering
North Carolina State University
Raleigh, United States
snaraya9@ncsu.edu

Shreyas Chikkaballpur Muralidhara
Computer Science And Engineering
North Carolina State University
Raleigh, United States
schikkb@ncsu.edu

***Abstract*—Emails have become the primary mode of communication in schools, colleges, professional industries and other such organizations. With the humongous influx of messages everyday its important to prioritize the emails which should be responded first. A priority based email queue is a solution to address this problem. Also, we can further extend this feature with an automatic response recommendation that would help in cleaning mails faster. The emails we receive can be categorized as Informational and primary emails. The latter needs attention from an individual by an action item, immediate response, task reminders or necessary details for a days work. But the informational ones can be - read only include promotions, daily newsletters, broadcast messages or copied emails that doesn't address directly to individual, irrelevant or spam messages.The response recommendations would be Reply, Delete or Save or Later. Proposed solution is to build a email ranker which lists a priority based email set, this would be augmented with a recommendation system which provides a probable course of action which would be performed on the emails.**

***Index Terms*—Email Priority, NLP, PageRank, Similarity, Connected Network Componen, POS tagging, Doc2Vec, TF-IDF**

I. Introduction

The project consists of two main parts, first one being a priority ranker and the second one is recommendation system. The influx of mails will pass through a pipeline of components, which includes data cleaning or pre-processing, data augmentation, ranking algorithm and through the trained model to predict the recommendation.

The Novelty of this project is the new email ranking which uses new weight extraction methods. The weight extraction included interaction time processing, interaction specifics, similarity and other features which are detailed further in the sections below. Also important findings are hyperparameter tuning and finding the right vector dimensions for the corpus features.

We had a huge corpus of uncleaned data which required tedious pre-processing and annotation. Data cleaning included removal of stop words, punctuation, next line and tab spaces, special characters and finally replacing null values with a blank character. Data augmentation included Lemmatization, Stemming and Parts of Speech Tagging. A modified version of PageRank algorithm was used to rank the emails based on weight of the email users(Explained in further sections). Finally, multiple models such as SVM, NB and Random Forest were trained for recommendation systems.

Extracting features from the cleaned data was the next step in the project. We scraped through thousands of emails and used regular expression to extract the key features. One such example would be, if the subject had a prefix RE which indicates that it is a part of a conversation. Further information on feature extraction is detailed below.

As a baseline model we targeted [3] that uses Naive Bayes classifier had an accuracy of 60 percent. Our trained SVM model crossed this by 10 percent.

Few key challenges which we faced are data extraction, weights augmentation for ranking, cleaning and pre-processing. The next section details our implementation.

To find the best model we tried competing approaches such SVM, Random Forest, KNN and Naive Bayes.

II. RELATED WORK

In [6] the prior work related to response recommendation was done using probabilistic approach. The used Naive Bayes and ngrams train the model.

[7] talks about email prioritization which uses the message body to find the urgency of the email and ranks them in that order. This also uses prior user actions to compute the rank.

[8] looks at email response recommendation based on user actions. Our key focus was to work on the email content rather than the previous history of data.

[9] uses machine learning methods to prioritize the email as low and high critical. They have worked on SVM model and they built ROC curves for the same.

III. IMPLEMENTATION

The implementation of this project can be divided into these major subdivisions.

- Pre-processing
- Annotation
- Data Augmentation

- Page Rank Implementation
- Recommendation System Implementation

A. *Pre-Processing*

We have selected Enron Email Dataset for our project. This dataset is a database of over 600,000 emails generated by 158 employees of the Enron Corporation. Within the dataset the emails are placed in directory structure based on the user profiles. This dataset captures the interaction among the users. Each email is stored in a text format. The most important challenge was to convert this text data into a feature list and finally into a csv format.

The text files where traversed one by one and each of the text file was scraped for data. The scraping was done with regular expression using 're' library. The findall function in re picks all the patterns which are in the string as a list. This list contained all the values which started and ended with the required pattern. We further stripped the contents of the list to get the exact feature values from the email. This features was finally added to the csv file.

As this dataset was huge we had to select specific users to create a compact dataset which would be suitable for computation and easy analysis. We created dataframes from the csv generated and wrote multiple queries to pick top user interactions with considerable data.

Few of the challenges which we faced was to find a suitable pattern to get the feature required from the email. For Example, few emails had names included in the from part of the email. This extra data cannot be stripped as usual. We came up with multiple patterns which extracted these information for us. One other challenge which we faced was few emails didn't have the data which we were expecting from it. We had to either replace it or completely ignore that data point.

B. *Annotation*

For recommendation system, we needed annotated email feature list. The main recommendation includes delete, reply and thread. Delete means that the message is either no longer relevant or it contains details which are out of scope for the user. Reply means that the message has an action item which requires the user to respond the email. Finally, a thread email is one which goes to the inbox and no specific action is required from the user.

We had considered one user[MANN-K] for our analysis. This user had 9571 email interactions, which included this user either being in the To address or the from address. The cleaned data had these features, To, From, Subject, Content, Date, and filename. We manually annotated these emails based on our understanding of the user interaction. If the email consisted of promotional data, out of scope information for the user, spam mails and of it was an end of the conversation it was labelled as delete. If the email contains a reply then it was tagged as reply and finally if the email was not replied and did have some important information it was tagged as thread.

The main challenge of annotation was the sheer volume of the dataset which we had selected. This was a strenuous and manual work which took considerable amount of time.

C. *Data Augmentation*

After we had the cleaned and annotated data, to improve information from the dataset we had to further augment the data. We mainly used three important augmentation techniques, namely Lemmatization, Stemming and POS Tagging.

Stemming is the process of finding the base word of the given word by removing the affixes. For example, Eaten, eating, eats has the stem as eat. We used PorterStemmer from nltk.stemmer to get the stem of the words.

Lemmatization if the process of finding the root word of the given word. For example, better has good as the lemma. We used WordNetLemmatizer from nltk.lemmatize to get the lemma of the words.

Finally, parts of speech in context with the sentence is an important data which will increase the accuracy of the model. So we tagged each word with POS and appended it to the word with a delimiter('/'). NLTK.POSTAG to get the POS of each word.

D. *Page Rank Implementation*

PageRank is an algorithm used to calculate the weight for web pages. This algorithm considers each web page as a directed graph node with the links in each page to other being the edge. If webpage A has the link to web page B, it can be represented as a directed edge from A to B.

The weights of the nodes are given by,
$S(V_i) = (i - d) * d * \sum_{j \in ln(v_i)} S(V_j)$

So we implemented a variant of this algorithm where each node would be the user email. We compute the weight of the nodes using NLP. The links between the nodes would be user interactions. As we didn't have any other data such as marked as important or starred mail, we had to go with the only data of user interactions, time between the interactions, the content of the email and repetitions between the users. We used the above parameters to find the weight of the nodes which was multiplied with the node links in multiple step iterations to find the rank. The above formula was used to find the same.

The complete implementation was done in Python. We queried the pandas dataframe which we had created from the csv data file. For the selected user(MANN-K) we grouped by the interactions with multiple users. This node pair was converted in to an ordered dictionary. Where the key would be the From email and the value being To email. For each of this node pair we computed the weights.

We had multiple weight parameters such, finding a urgency match, time duration between interactions, interactions count, spam match and so on. We selected the first three to compute the weight of each node.

Key challenges faced here were to compute the weight matrix, build a framework were people can build upon.

E. *Recommendation System Implementation*

Recommendation system suggests the type of response for a given email. The default type is *thread*. Based on the corpus of subject with prefix *RE:* or sharing the same subject thread, we have mentioned the response type as *reply*. Based on the content corpus, if it was promotional/generic email that was not interacted earlier can be labeled as *Deleted*. Email corpus for profile MANN-K was manually annotated, to recommend the response.

We implemented two models for identifying the response type - *Term frequency- Inverse document frequency(Tf-Idf)* and *Doc2Vec sentence embedding model*. Tf-Idf considers the features *From, To, Email date, Subject, email content* for generating count vector matrix. the features extracted were concatenated and fed as statement for the Count Vectorizer - sklearn feature extraction package.Then the count matrix was transformed to TF-IDF using Tfidf transformer - sklearn feature extraction package. Since the corpus consisted of more words over the other features, it was assigned the most weights. Classification models, - K Nearest Neighbor, Random forest, Decision trees and Naive Bayes models were tested for the accuracy and correctness. Naive Bayes had the highest accuracy but it couldn't categorize the *delete* responses, due to relatively smaller sample size. Thus the need for more robust weighted sentence embedding model Doc2Vec.

Doc2Vec embedding model generates the individual tagged documents for the features *From, To, Email date, Subject and Content*. Vocabulary was constructed for each features, later was used to generated the vectors for the defined feature model. Concatenating feature vectors with equal weights, we represented the email features as a single vector. Since Support vector machine works the best for vector models, Radial Basis kernel was used as classifier. The regularization factor was hyper parameter tuned. Later the weights associated with each of the features was updated to capture the Subject and Corpus in higher vector space, From and To feature vectors remaining the same, and the Date vector space reduced to lowest priority.

IV. CHALLENGES

The complete project was built in python. The important libraries used were nltk, re, pandas, numpy, sklearn, matplotlib and graphviz.

The first and most difficult challenge we faced was to convert the data from a directory based text email structure to csv based feature list. The emails had to traversed and scraped from data. We had to find a generic pattern which would be good to find the expression across all the emails. There were multiple date formats which had to be brought into a single one. The dates were given in formats such as Sun, 30 Dec 2001 10:19:42 -0800 (PST), 11/27/2001 and 11-27-2001. All these different date types was brought into a single datetime format. Also while extracting the features we had to ignore the thread details.

The next challenge was creating the framework which could be reused in the ranking system. We had to think about multiple exit points. For example, computing the weights could be done in multiple ways. So we have exit points from the compute weight function where new functions can we augmented. How do we use this project created for a single user for other users was another challenge. We wrote generic queries which when given an user can runs the same, given the data is scarped for that user.

Another challenge we faced was annotating the large corpus of data and correctly annotating them. Again we had to come up with a framework before we annotated the emails. Understanding the emails and its context was an added challenge in this task. As we considered reply as an important aspect while categorizing the reply email, we had to ensure that the subject was indeed a response while internally checking the content of the mail.

Assigning weights to the feature vectors for the Doc2vec model was challenging. Increasing the feature vectors space captures the more attributes for specific feature over weighing the others. Tuning the feature weights for higher precision and recall for all class labels was a challenge.

V. RESULTS

Fig.1 shows the users as the node vertices and the edges as the interactions between them. As we can see that the second node is the user under consideration has interactions with every other user. Interactions between other users increases their weight.

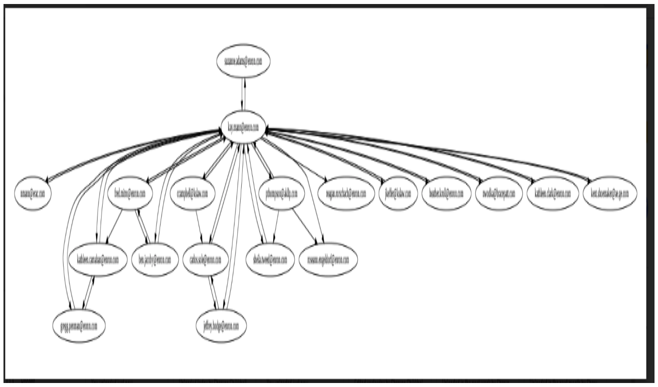
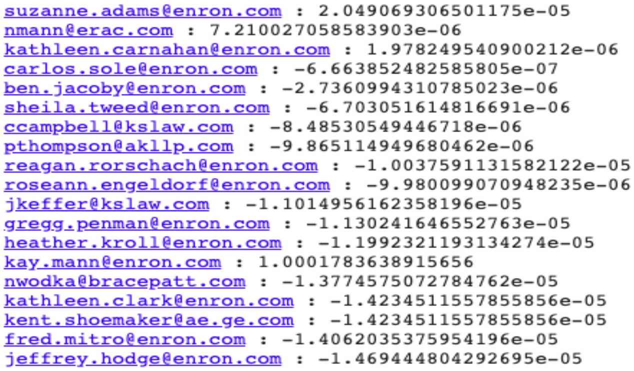


Fig. 1. Digraph between users and their interactions

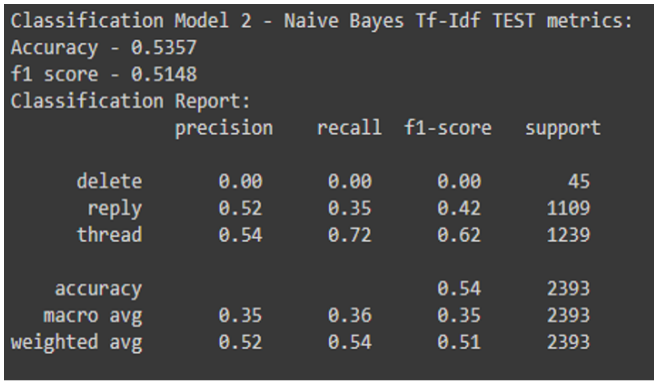
Fig.2 shows the rank score for of users who interacted with MANN-K. The values can be sorted to find the user with highest rank.



```
suzanne.adams@enron.com : 2.049069306501175e-05
nmann@erac.com : 7.210027058583903e-06
kathleen.carnahan@enron.com : 1.978249540900212e-06
carlos.sole@enron.com : -6.663852482585805e-07
ben.jacoby@enron.com : -2.7360994310785023e-06
sheila.tweed@enron.com : -6.703051614816691e-06
ccampbell@kslaw.com : -8.48530549446718e-06
pthompson@akllp.com : -9.865114949680462e-06
reagan.rorschach@enron.com : -1.0037591131582122e-05
roseann.engeldorf@enron.com : -9.980099070948235e-06
jkeffer@kslaw.com : -1.1014956162358196e-05
gregg.penman@enron.com : -1.130241646552763e-05
heather.kroll@enron.com : -1.1992321193134274e-05
kay.mann@enron.com : 1.0001783638915656
nwodka@bracepatt.com : -1.3774575072784762e-05
kathleen.clark@enron.com : -1.4234511557855856e-05
kent.shoemaker@ae.ge.com : -1.4234511557855856e-05
fred.mitro@enron.com : -1.4062035375954196e-05
jeffrey.hodge@enron.com : -1.469444804292695e-05
```

Fig. 2. Page rank of users interactions

Fig.3 shows the test result for the Tf-Idf Model with the Naive Bayes Classifier. The accuracy was **53.57%** and Precision was **51.48%** along with rest of the labels class wise is displayed. Since the train sample size for *deleted* was significantly less than the other 2 labels, accuracy and precision was almost 0.



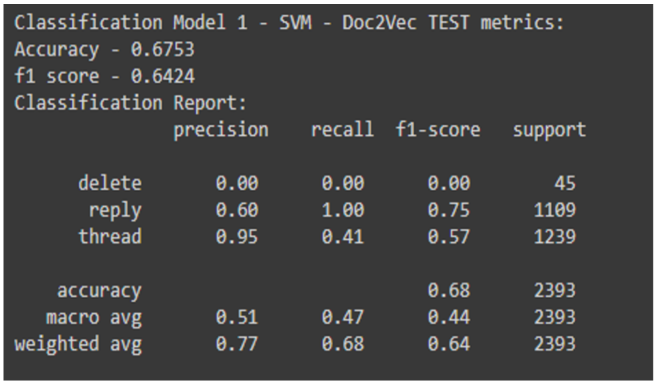
```
Classification Model 2 - Naive Bayes Tf-Idf TEST metrics:
Accuracy - 0.5357
f1 score - 0.5148
Classification Report:
              precision    recall  f1-score   support

      delete       0.00      0.00      0.00        45
       reply       0.52      0.35      0.42      1109
      thread       0.54      0.72      0.62      1239

    accuracy                           0.54      2393
   macro avg       0.35      0.36      0.35      2393
weighted avg       0.52      0.54      0.51      2393
```

Fig. 3. Tf-Idf model with Naive Bayes classifier

Fig. 4 shows the test result for the Doc2Vec model with the Support vector machine - Radial Basis kernel for the Hyper parameter C = **0.1**. Accuracy achieved was **67.53%** and the precision was **64.24%**. The classification report and the metrics is captured as well.



```
Classification Model 1 - SVM - Doc2Vec TEST metrics:
Accuracy - 0.6753
f1 score - 0.6424
Classification Report:
              precision    recall  f1-score   support

      delete       0.00      0.00      0.00        45
       reply       0.60      1.00      0.75      1109
      thread       0.95      0.41      0.57      1239

    accuracy                           0.68      2393
   macro avg       0.51      0.47      0.44      2393
weighted avg       0.77      0.68      0.64      2393
```

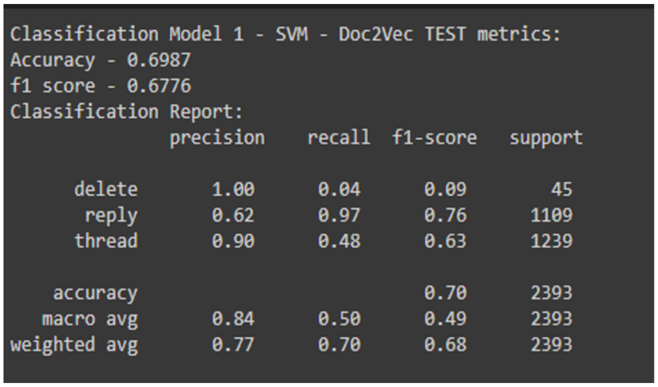
Fig. 4. Doc2vec Sentence embedding with SVM-RBF kernel and C - 0.1

Fig. 5 shows the test result for the Doc2Vec model with the Support vector machine - Radial Basis kernel for the Hyper parameter C = **1**. Accuracy achieved was **69.87%** and the precision was **67.76%**. The classification report and the metrics is captured as well.

VI. LIMITATIONS

Even though we tried our best to create a generic framework which would be cover every dataset and problem statement, few limitations are ought to pop up.

As the email data is personal and due to privacy issues not much data has been published outside. Hence related work was as minimal as possible for this kind of project, to form a baseline and compare.



```
Classification Model 1 - SVM - Doc2Vec TEST metrics:
Accuracy - 0.6987
f1 score - 0.6776
Classification Report:
              precision    recall  f1-score   support

      delete       1.00      0.04      0.09        45
       reply       0.62      0.97      0.76      1109
      thread       0.90      0.48      0.63      1239

    accuracy                           0.70      2393
   macro avg       0.84      0.50      0.49      2393
weighted avg       0.77      0.70      0.68      2393
```

Fig. 5. Doc2vec Sentence embedding with SVM-RBF kernel and C - 1

To start of with the data, our project and scraping is specific to Enron data set. Even though we can use parts of our code, we cannot completely reuse. Few regular expression built for scraping is again email specific, like date field which had multiple formats may not fit in other data sets.

Moving ahead, the page rank algorithm takes in the weighted matrix for the nodes which is relative and sometimes outlier may corrupt the same. These outliers are not handled in our project yet.

The recommendation system was trained on a supervised classifier which used word embeddings, this doesn't take named entity recognition into consideration which is an useful information while processing email data.

Intrinsic negations of urgency are not considered while processing the weights of the nodes in the page rank algorithm. This adds noise to the data, which will again result in erroneous results.

A simple yet trivial limitation are the packages and language used for programming this project cannot be used across multiple platforms and programmers.

VII. PREREQUISITES TO APPLY

We tried our best to create a generic framework to incorporate as many datasets possible and try new things out.

First and foremost prerequisite is the technology stack which is used in the project. It mainly consists of python, nltk and relevant regular expressions packages. To use this, you should have basic understanding of the stack.

Email Ranking framework can be directly incorporated by getting the nodes in an ordered dictionary format and weights in a matrix. This can be fed to the email ranking algorithm and desired output can achieved.

Recommendation system models can be directly used by again getting the email data to the specific format and feeding it to the model. By generating the csv or using the csv generator will achieve this.

Using the accuracy or citing it somewhere would require similar dataset and approach.

VIII. CONCLUSION

In this project, we experimented email ranking which is one of its kind and created a framework to extend the ranking algorithm's weights. Our cleaned and annotated data can be used for other NLP based applications. We have also created a program to extract important features from the directory structure.

The recommendation system was trained on various models and we got the following accuracy. We can see that SVM with RBF kernal of 1 gave the highest accuracy.

Model	Accuracy	F1
SVM(Doc2Vec)(C=1)	69.87	67.76
SVM(Doc2Vec)(C=0.1)	67.53	64.24
Naive Bayes(TFIDF)	53.57	51.48
Random Forest(TFIDF)	43.13	43.16
KNN(TFIDF)	49.19	48.98

Future scope of this project is that it can be experimented on various ranking algorithms. Neural network models can be trained for the recommendation systems. Various new dataset can be incorporated with the same model to check the accuracy and other metrics. Applications can be built on this project which could be commercialized.

REFERENCES

[1] Towards Explainable NLP: A Generative Explanation Framework for Text Classification, Hui Liu, Qingyu Yin, William Yang Wang, 2018, arXiv:1811.00196, https://arxiv.org/abs/1811.00196

[2] Yoo, S. (2010). Machine learning methods for personalized email prioritization (Doctoral dissertation, Carnegie Mellon University).

[3] Conway, D. and White, J.M., 2011, January. Machine Learning for Email: Spam Filtering and Priority Inbox isbn-9781449320706.

[4] Ha, Minh Tran, Quang Luyen, Thu. (2012). Personalized Email Recommender System Based on User Actions. 280-289. 10.1007/978-3-642-34859-4_28.

[5] Page Rank Algorithm Reference https://github.com/bhaveshgawri/PageRank

[6] Improving Email Response in an Email Management System Using Natural Language Processing Based Probabilistic Methods - https://www.researchgate.net/profile/Abdulkareem_Al-Alwani/publication/283028323

[7] Hasegawa, Takaaki Ohara, Hisashi. (2000). Automatic Priority Assignment to E-mail Messages Based on Information Extraction and User's Action History. 573-582. 10.1007/3-540-45049-1_69.

[8] Personalized Email Recommender System Based on User Actions https://link.springer.com/chapter/10.1007/978-3-642-34859-4_28

[9] Horvitz, E. et al. "Attention-Sensitive Alerting." UAI (1999).