Classifying Emails with NLP

Aidan Aguilar, Josh Kang, Matthew Purvis

Introduction

During the four years a midshipman spends at USNA, they get on average around twenty to sixty emails a day. Over time throughout each month, semester, and academic year, the emails add up quickly. Overall they get way too many emails from various officers, civilians, and other midshipmen. Additionally, these officers, civilians and other midshipmen all write emails differently to those subordinate, superior, or sometimes they may require specific action. Due to the constant share of emails, there should be a model that predicts who the email is from before it needs to be read. A model that classifies emails will allow the following questions to be asked:

1. Does the model predict if an email is from a midshipman?
    a. And specifically whether it is from a plebe or firstie?
2. Does the model predict if an email is for action required?
3. Does the model predict if an email is from a company officer?
4. Are there certain groups/clusters that emails tend to fall into?

By focusing the model on answering these questions, it allows for people to see what specific or distinctive ways different groups of people write emails that a classifier can pick up on. By classifying emails and predicting where the email is from based on the body of the text and using further sentiment analysis we can gain more insights about emails and who sends what kinds of emails and what kinds of different emails there are.

Datasets Used

The data collection was conducted by utilizing midshipman based usna.edu emails. The emails came from Midshipman Aguilar, Midshipman Kang, and Midshipman Purvis. Due to significant ITSD permissions on the ability to get a GMAIL API, the team had to utilize a different process of acquiring the emails. Microsoft Outlook had to be downloaded to be used to download the emails. Over 47000 emails were downloaded and exported from outlook. The emails were put into .pst files. The .pst corpuses were then parsed in python with assistance from the Pickle Module in Python. By utilizing the Pickle Module the .pst files were then made usable to be put into the models that the team created. Saving the data as .pkl files allowed a quickly accessible dataset that could be loaded in faster than other formats such as CSVs.

Algorithm and Model

The general approach for designing the algorithm consisted of testing various different classifiers and parameters to find the one that performed the best. The final algorithm used a logistic regression classifier to create models that would then pick whether or not the body of an email is one class or another. The classes were determined by whether or not the email was from a plebe,

firstie, company officer, midshipman, etc. Before the final classifier was used, decision trees, support vector machines, and k nearest neighbors were all tested, but the best classifier to come out of it all was the logistic regression method. After deciding on logistic regression being used, sklearn with the ngram range from 1 to 5 was utilized for sequencing when the text was classified. The range was determined through trial and error by choosing the range that maximized accuracy, but a flaw of this approach was that it may lead to overfitting. To automatically label all of the data before entering the classifier, regex was also used on the sender information to automatically label all of the data where the regular expression looked to match any given format. For example, the regex looked for the pattern "m2XXXXX" when looking for a Midshipmen and it returned 1 if it matched the pattern and 0 was returned if it didn't. The image below labeled as **Figure 1** displays the example pattern:

```
# Regular expression pattern to match the format 2XXXXX@usna.edu
pattern = r"m2\w{5}@usna\.edu"

# Return 1 if the email matches the pattern, else return 0
return 1 if re.match(pattern, sender) else 0
```

Figure 1

After the corpuses of emails go through the pattern parser and are labeled, they are put into the train_classification_model function and from there they are split into training and testing sets. A pipeline using TF-IDF vectorizer with character level n-grams and logistic regression are then used. From there the model is trained and then it is evaluated. The trained model and test data were then saved using joblib. The email content or body is used as features and the labels, which are where the email is from, are used as targets. The model's code is shown below in **Figure 2**

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = make_pipeline(
    TfidfVectorizer(analyzer='char', ngram_range=(1, 5), stop_words=None),
    LogisticRegression(max_iter=200, class_weight='balanced')
)
model.fit(X_train, y_train)
```

Figure 2

Experiment Setup

The purpose of the predictMODELNAMETester.py was to have a script that evaluates a trained classification model to identify its performance and errors. The saved model was evaluated on the test data and the false positives and false negatives were also identified for error analysis. In the evaluate_model function, the previously saved model, test features and test labels were all

loaded in for use as a cached model. The predicted labels for the test set are predicted using the loaded model. Then afterwards, the classification report and accuracy score were calculated and produced for overall performance metrics. Additionally, the function identifies and analyzes the errors where false positives and false negatives were determined. A false positive was identified in cases where the model predicted 1 in a case of a match but the true label was 1 a match. False negatives were identified in cases where the model predicted a 0, a no match, but the true label was a 1 which was a match. For each error type, a number of test examples of test instances were then printed for manual inspection to determine patterns. The experimental code also included error handling for cases where the model file failed to load properly in the tester. The outputted metrics also consisted of a classification report with precision, recall, F1-score, and support for each class.

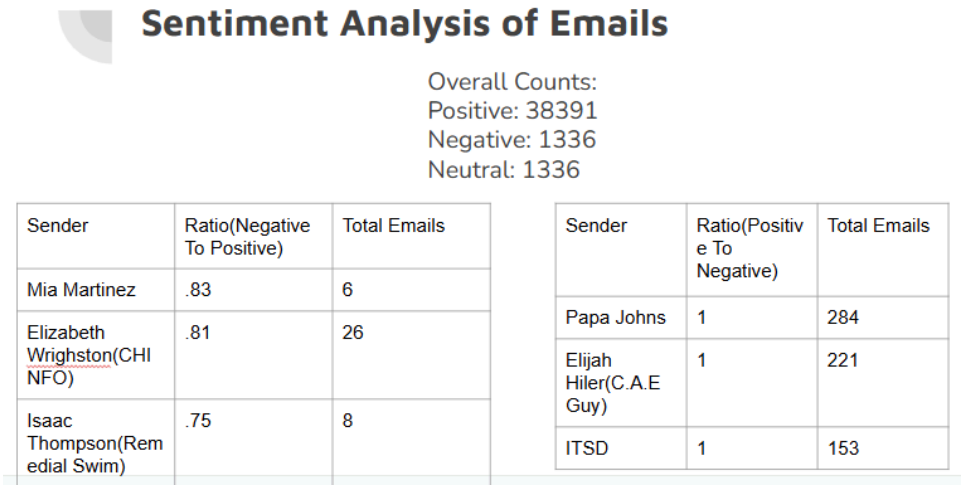
However, when we first evaluated the test cases we realized that there could be some aspects of the emails that were causing biases/problems. For example, numbers that could signify class years and emails being included in the body could allow the model to potentially "cheat" and use these to make predictions rather than the overall text of the email. Therefore, we made a cleaner(numDeleter.py) that would strip the email bodies of all numbers and emails to prevent this sort of potential "cheating".

Results

The results of the model were quite accurate after the training and evaluation was complete. Below is a breakdown for the various models

- Predict Midshipman:
    - Accuracy: 96.2%
    - False positives: Consisted of being mostly from professors who had short emails. It showed that the model thinks an email being short correlates to being from a midshipman.
    - False negatives: Consisted of longer emails with links to google docs/forms. It showed that the model thinks and email that links to google docs are NOT from midshipmen.
- Predict Plebe/Firstie:
    - Accuracy of Plebe: 92.89%.
    - Accuracy of Firsite: 93.78%.
    - Of note, both plebe and firstie datasets had to artificially be balanced because both datasets had lots more of non-firstie/plebe than firstie/plebe.
    - False positives: Came from emails that were not necessarily from a plebe or a firsite by forwarding an email from a plebe/firstie.
    - False negatives: Mostly came from emails from a plebe or firstie that did not end in Very Respectfully.

- Predict Company Officer:
    - Accuracy: 90.4%
    - Also artificially balanced this dataset
    - False Positives: Mostly emails from people with more lines in their email signature
    - False Negatives: From company officers with shorter email signatures
    - This shows that the model associates long email signatures with being a company officer
- Predict Action Required:
    - Accuracy: 85.96%.
    - Notable, this prediction was most likely to have mislabeled emails because it just labeled an email as "Action Required" if the phrase "Action Required" was in the subject line.
    - Most "false" positives/negatives actually did have some sort of action required,
    - requested, or needed, but they were not labelled correctly



Sender	Ratio(Negative To Positive)	Total Emails
Mia Martinez	.83	6
Elizabeth Wrighston(CHINFO)	.81	26
Isaac Thompson(Remedial Swim)	.75	8

Sender	Ratio(Positive To Negative)	Total Emails
Papa Johns	1	284
Elijah Hiler(C.A.E Guy)	1	221
ITSD	1	153

Figure 3

The next portion of the results consisted of running sentiment analysis of the emails.See visual in **Figure 3**. Overall counts consisted of 38391 were positive, 1336 were negative, and 1336 were neutral. The senders with the highest ratio of negative to positive consisted of .83, .83, and .75. The emails from CHINFO were tied for second on being the most negative. Whereas the top three positive to negative senders all had a ratio of 1. The top 3 in the positive area were Papa-Johns, Elijah Hiler, and ITSD. CHINFO and Remedial swim emails make sense considering world news is typically negative and emails pertaining to Mids who need remediation are negative. All of the positive emails make sense because marketing from Papa

Johns will be positive, academic support will be positive, and ITSD requesting action will do so politely.

Lastly to see whether or not the emails had a distinct pattern, BERT was used to perform cluster analysis. We tested various numbers of clusters, but eventually settled on the best number of clusters for meaningful groupings being 10 because we would get approximately 10 different types of emails. Cluster 0 was academics/student support, cluster 1 was ECA's, cluster 2 was admin, cluster 3 was online coursework, cluster 4 was news/external/publications, cluster 5 was sports events, cluster 6 was community/social engagement, cluster 7 was academic assignments, cluster 8 was military news, and cluster 9 was personal/informal messages with a casual tone.

Future Work

Future work for our project would be to focus on three main areas: model enhancement, data expansion, and action prediction improvements. First, enhancing the model by using more advanced classification techniques through transformers or neural networks, could increase accuracy and reduce false positives and negatives. Also, implementing methods that combine multiple classifiers could provide a more balanced prediction system. Second, expanding the dataset by including more emails from a diverse range of midshipmen, faculty, and staff would improve model generalization. Enhancing the automated labeling process through the incorporation of text summarization could also allow more accurate data labeling. Finally, action prediction could be improved by incorporating intent detection models to distinguish between different action types. Developing a task management assistant that automatically prioritizes emails based on urgency would provide practical application of the model in real-world email management. These three areas would advance the systems performance, scalability, as well as usability.