ClutterCutter Project Report

Tian Xi (Cindy) Hu, Ru Shi (Rosie) Wang, Theodore Wu

Submitted: December 6, 2020

Word Count: 1965

Penalty: 0%

Permissions

Group Member	Permission to post video	Permission to post final report	Permission to post source code
Rosie	Yes	Yes	Yes
Cindy	Yes	Yes	Yes
Theo	Yes	Yes	Yes

Introduction

The relentless flood of emails, from ads and newsletters to course announcements and family correspondences, has become an unavoidable part of everyday life. Obviously, certain emails have a higher priority than others, but finding them amidst an unsorted inbox is challenging. Too often, we miss important emails or spend too much time digging through past archives. While basic spam filters may help, we desire a solution that finely organizes emails into categories that are relevant to us, as students.

ClutterCutter is a neural network solution designed to classify emails into labelled folders, which is a repetitive task that humans can easily perform using a set of internal rules. We suspect that a machine learning model can be trained to imitate human performance. Specifically, Natural Language Processing (NLP) has been proven to be effective in the interpretation of text data, making it highly practical for assessing and labelling email content.

Illustration

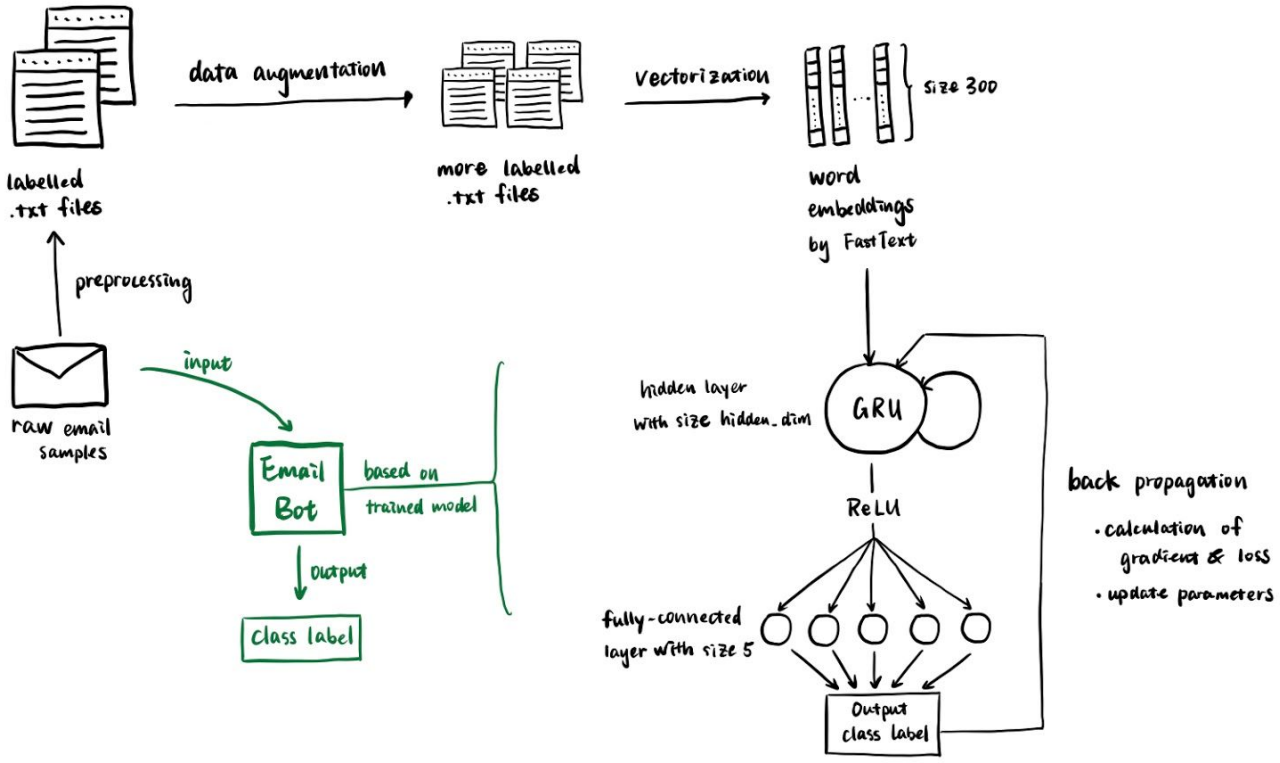

Figure 1: *Illustration of the model structure and data processing pipeline.*

Related Works

We highlight two works that perform a related email filtering task:

1. **IBM's MailCat [1]:** MailCat builds a TF-IDF text classifier by analyzing a user's existing email classification behaviour to filter incoming mail into three possible folders. It provides predictions which are displayed as a shortcut button on each email. If the prediction is approved by the user, automatic sorting can be completed with a simple button click. Additionally, the classifier is adaptive and continues to train by learning the approval patterns of the user.

2. **Artificial Neural Network for Spam Classification [2]:** This paper implements a neural network solution to classify emails as spam/not spam. The network trains with inputs correlating to the number of occurrences of specific words and other text properties in the email samples.

Data and Data Processing

Our data consists of email samples collected from our team's inboxes, individually stored as .txt files. They were manually organized into five categories: Academics, Alerts, Personal, Professional, Promotions and Events. For emails that fit into multiple categories, team discussion was carried out to set a specific label. Hyperlinks and personal information was also manually removed.

In total, we collected 1000 emails (i.e., 200 emails per class). Data augmentation with back translation [3] and lexical substitution [4] was performed to increase our data count to 2850. With back translation, each text file was randomly translated to one of four new languages using a Google Translate API, then re-translated back into English. With lexical substitution, a random set of words in each text file was replaced with a WordNet synonym.

To standardize all text samples, we first converted all text into lowercase. Numbers, punctuation, extra spaces, and short words with lengths less than 3 were removed. Contracted words were expanded. The processed text files were then concatenated with their corresponding labels, forming the complete dataset. Finally, our data was split into three subsets: training, validation and test, with a split of 0.64, 0.16 and 0.20. Overall, the dataset contains 8872 unique words. **Table 1** lists the top 5 words in each category in decreasing order of occurrence. **Figure 2** shows an example of an original email alongside its standardized version.

Table 1: *5 most frequently appearing words for each email category.*

Academics	Alerts	Personal	Professional	Promotions and Events
'course', 'students', 'final', 'questions', 'engineering'	'order', 'email', 'account', 'password', 'information'	'practice', 'think', 'going', 'other', 'email'	'application', 'interview', 'position', 'interest', 'assessment'	'students', 'engineering', 'november', 'student', 'university'

```
Happy New Year!

We hope you had a restful and wonderful holiday!
We are aiming to start off this new year with
another tournament, and we were wondering if we
were able to hold it at your establishment once
again. If possible, is Saturday, January 20th for
about 5 hours in the afternoon work? We apologize
for the mishap with our bank last year, and we can
guarantee that wont happen again.

Wishing you and your family the best for the new
year!
```

```
happy new year hope you had restful and wonderful
holiday are aiming start off this new year with
another tournament and were wondering were able
hold your establishment once again possible
saturday january for about hours the afternoon
work apologize for the mishap with our bank last
year and can guarantee that wont happen again
wishing you and your family the best for the new
year
```

Figure 2: *Comparison between the original email (left) and its standardized version (right).*

Architecture

Prior to training, we first passed our cleaned data into an embedding layer to transform each word into dimension-300 vectors. FastText word embedding was chosen due to its higher performance in considering subword information, making the resulting embedding robust to out-of-vocabulary words [5]. For input consistency, each sequence of vectors was padded to equal length.

Our central model is an RNN composed of a single GRU unit with a hidden layer of adjustable size. It proved to be highly practical for our NLP task since the RNN aggregates contextual information over every word vector to make its final prediction. After the entire sequence was processed, we captured the final state of the RNN's hidden layer, applied ReLU for non-linearity, then passed this output to a 5-neuron fully connected layer. Each of the five outputs could then be interpreted as a "class score" for each category. Multi-categorical cross-entropy loss could then be applied for training.

Baseline Model

Using the same word-to-vector encoding scheme described above, the baseline model computes the average word vector over the entire email sample. We selected this approach because our pre-processing steps reduced each email to a "bag-of-words". Thus, an average word vector could capture the email's central meaning. This average vector was passed into a 5-neuron fully connected layer, where again, the outputs of each neuron directly correlated to a "score" for each of the five labels.

Quantitative Results

For our classification problem, we chose accuracy as the performance metric due to its intuitive interpretability in multinomial classification. Other classification metrics such as F1-score were omitted since our problem is not framed in terms of positive/negative results.

To demonstrate the effectiveness of our proposed solution, we can examine a comparison of test accuracies achieved by the RNN and baseline models, displayed in **Figure 3.** The RNN outperforms the baseline by approximately 10%.

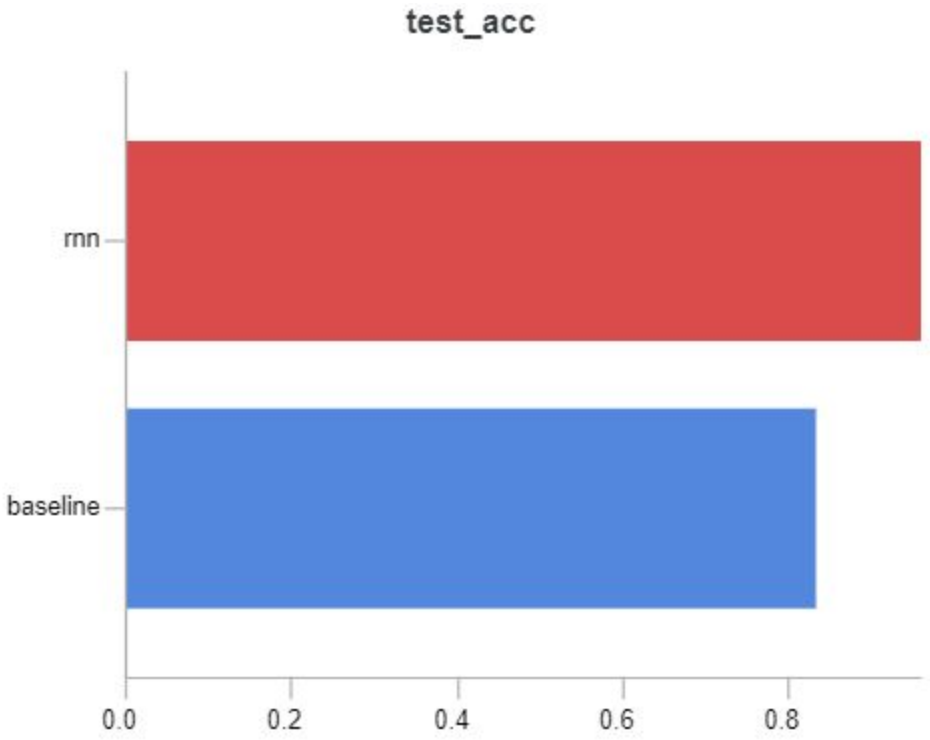

Figure 3: *Test accuracy comparison of baseline and RNN model. Both models are learned with the same training hyperparameters.*

Hyperparameter search was conducted through a grid search over specific ranges. **Figure 4** aggregates a selection of training runs to demonstrate hyperparameter-test accuracy relationships. Examining the areas of greater clustering, we see a correlation between a higher batch size and a lower learning rate with better test accuracy. On the contrary, the hidden dimension size had a less obvious correlation. **Figure 5** showcases these observations in an alternative manner.

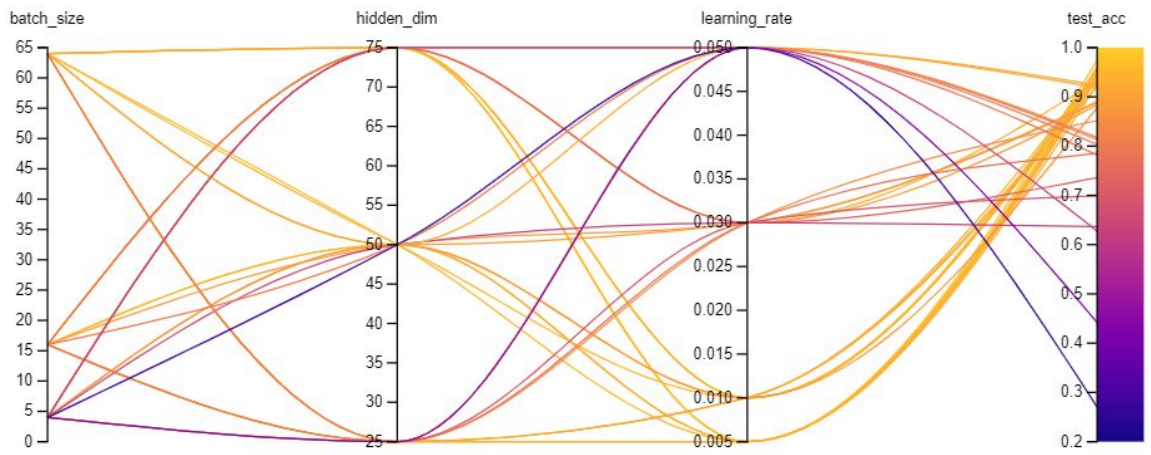

Figure 4: *Hyperparameter sweep over 36 search experiments. Each line denotes a particular experiment, where the intersections with the first three columns show the values of the hyperparameters for the given run and the intersection with the final column gives the test accuracy achieved.*

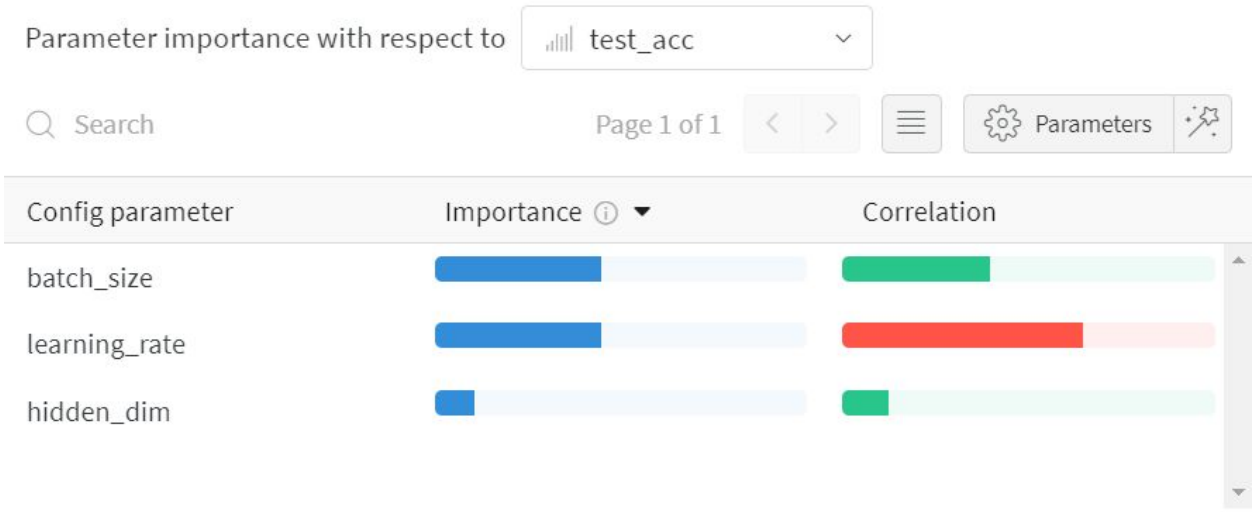

Figure 5: *Importance and correlation between hyperparameters with the final test accuracy. The length of the bars indicates a greater score. For the correlation score, green denotes a positive correlation whereas red denotes a negative correlation.*

After tuning our model with these heuristics, we obtained the training/validation loss and accuracy curves in **Figure 6.** Although the RNN still overfits slightly, the margin between its training and test performance is small (<5%). Furthermore, the model achieves a test accuracy of ***96.65%***, suggesting strong generalizability.

The confusion matrix in **Figure 7** shows the prediction distribution of the model on the test data. Though the matrix shows very strong performance, we observe that most errors occur from incorrect predictions of the “Promotions & Events” class, suggesting that the scope of the class may have been more ambiguous. For instance, “engineering” and “student” are words appearing in high frequency in both

"Academics" and "Promotions and Events" because our data is biased to contain a large number of engineering-related promotional emails. On the other hand, the least confused class is "Professional", which is reasonable since it has a distinct set of words (e.g. "application", "interviews") that provide a clear indication of an email's professional nature.

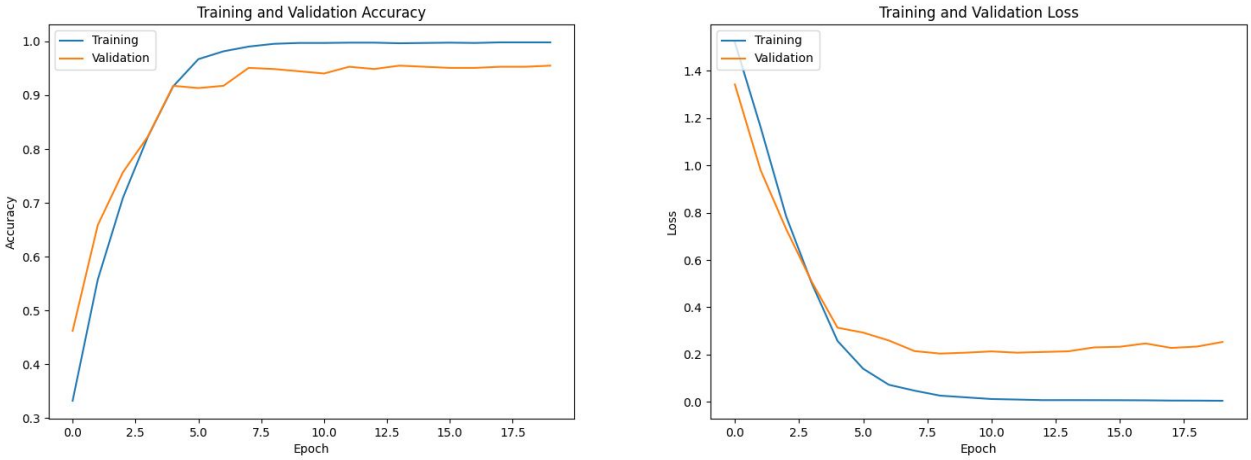

Figure 6: *Accuracy and loss curves of the best RNN model on the training and validation data. Hyperparameters: num_epochs=20, learning_rate=0.005, batch_size=32, hidden_dim=75.*

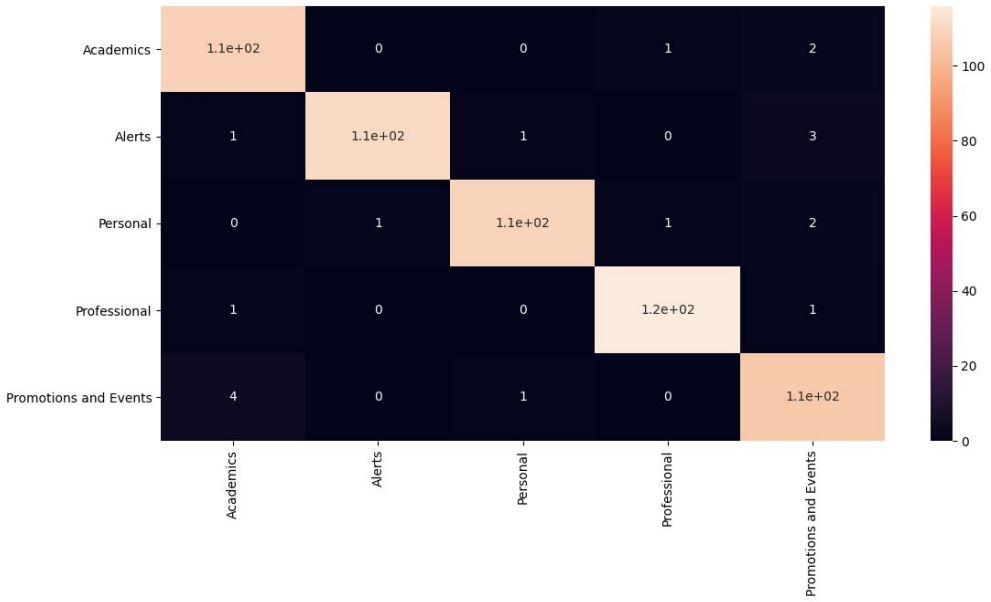

Figure 7**:** *Confusion matrix of RNN predictions on the test dataset. Columns are labelled with the predicted labels; rows are labelled with ground truth labels.*

Qualitative Results

To better understand the performance of our model on unseen data, we created the Email Bot. This bot leverages our trained model to generate a prediction of an input email's category. **Figure 8** shows the model correctly predicting two unseen emails. The model correctly categorized these emails because the subject matter is focused and the message contains keywords indicative of the label.

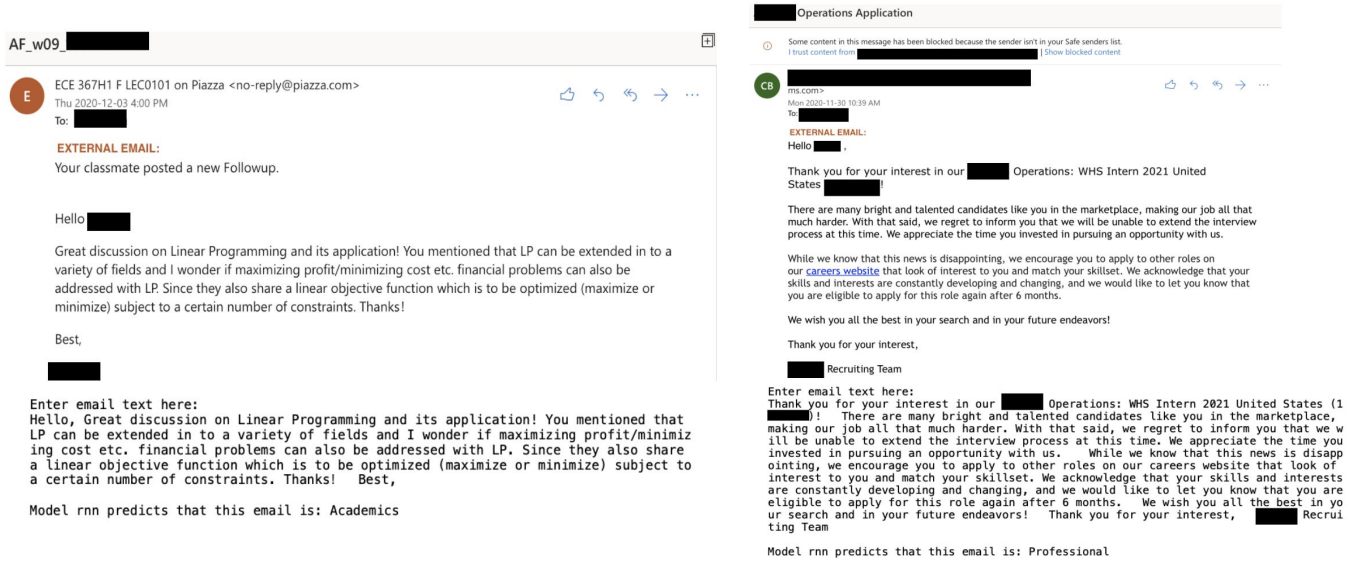

Figure 8*: *Two correctly predicted email samples.

From this bot, there were also incorrect predictions, as seen in **Figure 9**. A potential reason may be due to the nature of the email itself. For instance, the course evaluation email (left) matches the style of typical survey advertisements, which normally belong to the "Promotions and Events" class. Alternatively, the interview follow-up email (right) has few professional-related keywords (from **Table 1**) and rather contains mostly factual information, typical of promotional info-letters.

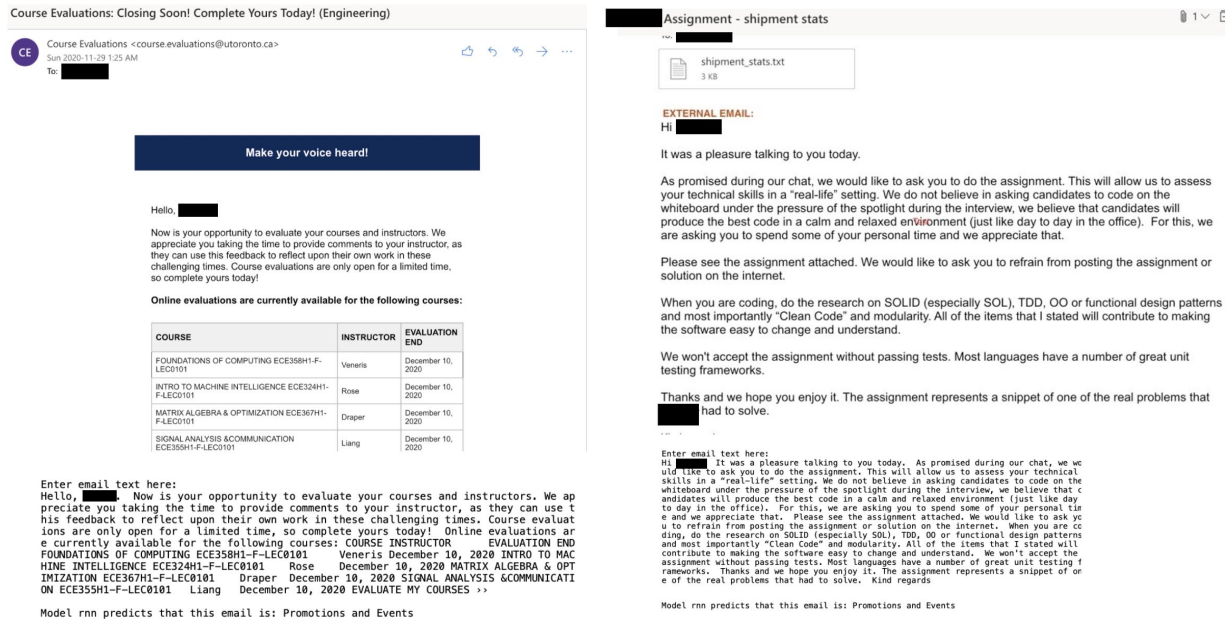

Figure 9*: *Two incorrectly predicted emails.

Discussion and Learnings

Based on the results above, we can conclude that our model performs well for classifying our team's emails. Our confusion matrix was strong and confused classes fell safely into a small margin of error. However, upon Email Bot's additional testing with unseen emails, incorrect predictions still occurred. This brings forward a few discussions.

First, it is important to note that our data source was very restricted. Approximately 75% of our data was collected from our team's personal and school mail inboxes, while 25% was from a student in Psychology. It has not been trained on a more diverse dataset, and hence we do not know how well it generalizes.

Second, our method of data augmentation may have been suboptimal. The process of random data augmentation and random splitting may have caused undesired similarities across the three datasets. The result would be a high test accuracy that might not reflect general performance. Hence, this could justify why the Email Bot makes mistakes more often than expected given our high test accuracy. For future iterations, a solution is to be more selective on how data is distributed between the sets to avoid repetition. The resulting test accuracy from this method may be more comparable to how the model generalizes.

Finally, with this project, we gained many valuable insights. Most importantly, data collection is associated with high amounts of time and labour. Our team manually collected the data without using an email-scraping API, resulting in an unnecessary time-consumption. The second key insight relates to the subjectivity of text categorization. We had many cross-categorizations that had to be fixed with strict, specific rules for particular email categories. At a larger scale, this level of specificity is unfeasible. A

potential solution could be to implement multiclass labels or unsupervised learning, which lets the model decide the number of labels. For future exploration, the changes discussed can help uncover a universal solution for a wider community.

Ethical Framework

ClutterCutter was developed with the objective of providing an efficient email organization tool. The main stakeholders to be discussed are users of ClutterCutter and us, the project team. Other stakeholders include email service providers and the instructors of ECE324. We analyze the ethical implications of implementing ClutterCutter, according to the four principles of reflexive principlism:

1. **Beneficence:**
   The project practices beneficence by improving user experience with an efficient tool to organize emails. It benefits us, the project team, by providing a unique opportunity to implement our knowledge from ECE324 into a real-world application.

2. **Respect for Autonomy:**
   The project team has the autonomy to create and launch this product. Users of ClutterCutter must also have the autonomy to choose whether they use it. However, user privacy may be violated if they do not recognize the permission they are giving ClutterCutter to access their email account.

3. **Justice:**
   ClutterCutter is trained on emails from students at the University of Toronto. It may perform better for users of similar backgrounds, providing them with higher benefits over other users. All team members will equally distribute the benefits or consequences resulting from this project.

4. **Non-Maleficence:**
   ClutterCutter could potentially cause harm to users by incorrectly sorting emails, causing further disorganization. Further, poor performance by the software could damage our reputation as developers.

References

[1] Richard B. Segal, Jeffrey O. Kephart, *MailCat: An Intelligent Assistant for Organizing E-Mail*, pp.1, 1999. Available: https://www.aaai.org/Papers/AAAI/1999/AAAI99-141.pdf.

[2] Alghoul, Ahmed, Ajrami, Sara Al, Jarousha, Ghada Al, Harb, Ghayda & Abu-Naser, Samy S. (2018). *Email Classification Using Artificial Neural Network*. International Journal of Academic Engineering Research (IJAER)_ 2 (11):8-14.

[3] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, Quoc V. Le. *Unsupervised Data Augmentation for Consistency Training*.

[4] Wei, J., & Zou, K. (2019). EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. doi:10.18653/v1/d19-1670

[5] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.