We've processed your Assignment extension request FORM-AEX-147939

no-reply@qut.edu.au <no-reply@qut.edu.au>
Sun 6/12/2022 5:47 PM
To: Sasha Le <sasha.le@connect.qut.edu.au>

Hi Sasha,

Thank you for your assignment extension request **(FORM-AEX-147939)**.

We have approved your request and the due date for your assignment **Assignment 2**, for unit CAB420 has been extended by 48 hours from the original due date. If your unit outline does not specify that your assignment is eligible for an extension, this confirmation email is not valid and unless you submit by the original due date, the late assessment policy will apply.

You are responsible for ensuring that this assignment is eligible for extension before submitting it after the original due date. Check your unit outline for eligibility.

As you indicated this is a group assignment, you are also responsible for informing other members of your group of this extension.

Be aware that a copy of this email is kept on file. You should not alter this email in any way. Email notifications that have been altered or differ in any way from the original may result in an allegation of student misconduct as set out in the Student Code of Conduct.

Need extra support? You can access free, confidential counselling with qualified professionals. We also offer planning and support if you have a disability, injury or health condition that affects your ability to study. If you are struggling to meet your university commitments, academic support is also available.

Have a question? You can contact us by email or phone. We're also available to help you on campus or via online chat. Visit the HiQ website for our contact details and opening hours.

Ref ID: 10091378 FORM-AEX-147939

ASSIGNMENT 2: ENRON EMAIL CLASSIFICATION

CAB420, Machine Learning

Group 43
Sasha Le, N10091378
Ash Phillips, N10477659
Raymond Thach, N08584605

1.0. Introduction and Motivation

The problem and motivation outlined for this project is to, using the Enron Email dataset, classify the text of an email and predict who sent it. To do so, the raw Enron Email dataset would need to be converted into a useful numeric form to then predict their senders. The three methods chosen to perform the classification task were K-Nearest Neighbours (KNN), Long Short-Term Memory (LSTM), and Latent Dirichlet Allocation (LDA) with Support Vector Machines (SVM). Multiple related works were analysed to support this method selection and processes to solve the problem. By using these methods, a further understanding of machine learning approaches was reached.

2.0. Related Works

Multiple methods to classify emails have been used throughout literature. Ahlstrand and Rosander (2017) outlined multiple deep learning and non-deep learning methods that can be used for this task. They stated that the methods selected are often chosen for their “group, diversity, and acceptance in the machine learning community”. They also outlined a solution to email classification using Bag of Words (BoW) to pre-process the dataset. They implemented BoW using scikit-learn to reduce the number of features and improve the quality of the classification process (Ahlstrand & Rosander, 2017). Using this method, they were able to cleanup the email data to only keep the features they required for classification, as we have attempted to do through our project.

2.1. K-Nearest Neighbours

The chosen KNN method has been used to classify emails throughout the literature, mostly focusing on spam detection. One example of this is an article in the *Journal of Soft Computing*, *K-Nearest Neighbour Classification of E-Mail Messages for Spam Detection* (Murugavel & Santhi, 2020). In this article they used density-based clustering, then the KNN classification algorithm to classify the selected attributes in the dataset that determine whether an email was spam. They also explored the computational requirements and performance of other methods; these were random forests, K-Means, decision trees, Naive Bayes, and Perceptron Rule Base. Through their testing, they found KNN had the shortest execution time (0.96 secs), lowest computational overhead, and the lowest energy consumption. They also found some weaknesses of this method as KNN had the lowest detection rate and the highest memory requirement (Murugavel & Santhi, 2020).

Another article, *Spam Image Filtering Using K-NN and SVM*, used KNN to again classify spam (Khalid et al., 2019). They investigated both KNN and SVM for the classification task and outlined strengths for KNN, that its calculation ability is powerful and easy to actualise, but also some weaknesses, that the method is

inefficient for large-scale, high dimensional data, and due to its lack of learning stage there is a high computational cost (creating a high memory requirement).

In the article *Spam Detection Using KNN, Back Propagation and Recurrent Neural Network*, three methods were analyzed for spam detection; KNN, Back Propagation (BPNN), and Recurrent Neural Network (RNN) (Kaur & Kumar, 2015). When looking at the results it was found the KNN method had an accuracy of 75%, lower than the RNN method (79.54%) but higher than the BPNN method (59.09%), showcasing the ability of KNN for classification.

2.2. Long Short-Term Memory

It is inefficient for users to open each email to manually check their intent, when there's an overwhelming amount. Therefore, Alibadi & Vidal (2018) applied a LSTM model to the Enron Email dataset for classification by "to read" and "to do", as an approach to resolve this email overload problem, by labelling each email to bring their intent to the surface automatically to distinguish them. They used a baseline LSTM model consisting of an embedding layer, two LSTM layers (256) each accompanied by a dropout layer (0.5), which was pre-trained on 80% of the data and tested on 20%. This baseline model was then used to fine-tune another LSTM model of similar structure producing 87.5% accuracy, which is 2.5% more than the initial model (Alibadi, 2018). They found that using pretrained models to train other LSTM models outperform using only a two LSTM layer model (Alibadi, 2018). While this is a strength in their approach, since they're using the entirety of the Enron Email dataset, training times may be long as there's a large number of samples, so training times may be a limitation to their approach if they're to consider training for longer. It's worth noting that both LSTM layers have 256 units executing, so running this model on Google Colab may be limited by GPU memory when fitting.

Building upon the previous approach, Alibadi et al. (2019) improves the performance of the pre-trained LSTM models in this problem space by using already trained word embeddings such as Word2vec, GloVe, and ELMo as well. They found that in comparison to a word embedding trained on the entire Enron Email dataset with 80.91% accuracy, GloVe performs worse at 77.82%, but Word2vec is better at 82.02%, and ELMo is best at 90.1%(Alibadi, 2019). This suggests that the pre-trained word embedding, ELMo, should be used with a pre-trained 2-layer LSTM model to yield the highest accuracy. However, this approach still uses the entirety of the Enron Email dataset, which means that training times should still be taken into consideration as the sample size is large.

To further build upon the previous approaches, Alibadi (2021) combines CNN with LSTM using pre-trained Word2vec and GloVe embeddings, and word embeddings trained on the Enron Email dataset. This time, the model consists of the input layer feeding into the embedding layer, followed by a convolution layer with 32 size 3x3

filters, accompanied by a maxpooling size of 2, 1 LSTM (100 units) layer then a dense output layer using the “sigmoid” method (Alibadi, 2021). They found that the model with word embeddings trained on the entirety of the dataset had the highest performance with 89% accuracy, while ones with the Word2vec and GloVe embeddings had 86% accuracy (Alibadi, 2021). However, the previous approach using pre-trained ELMo embeddings and pre-trained 2 layer LSTM models had a higher accuracy of 90.1%, so it has slightly higher performance than this approach. This approach still uses the entirety of the Enron Email dataset, so training time may restrict the number of epochs, and it’s also difficult to compare to the previous approaches as its model structure is significantly different.

2.3. Latent Dirichlet Allocation

Latent Dirichlet Allocation method has been used on the enron email dataset to analyse and explore key topics discussed in these emails. LDA is a topic modelling tool used for finding patterns in documents and grouping these patterns by topics that are characterised by distribution of words (Blei, Ng and Jordan, 2003). In this article, LDA was used to determine the topical distribution of each email which was then used as features along more direct features in the clustering of emails via K-means clustering (Harwath et al., 2022). This exploration allowed for further investigation of individual emails by viewing found topics and then searching for emails that scored highly on each of the topics. This may provide useful insights into the data however it is still unclear if the method can be used as investigation tools. This analysis can be further improved by focusing on users with higher roles in the company or users who have larger directories (Palaniswamy, 2015).

In the second article, this method was explored and used on emails for classification of email subjects to handle important information more effectively and quickly. Automatic email classification in this paper is based on the LDA topic model and SVM classifier. The LDA model is used to model text sets and the latent relations in text sets are mined to build the corpus, the SVM then classifies these features into the e-mail subject classification (Gong et al., 2018).

In the third article, LDA has been employed for fraud detection in electronic communication. This method was implemented on both phishing and non-phishing data as features which were then classified with Adaboost . This method also utilizes Conditional Random Field to discover the impersonated entity as the final process. This had a 88.1% impersonated discovery rate (Ramanathan, Venkatesh & Wechsler, Harry, 2013).

3.0. Data

3.1. The Dataset

The Enron Email dataset csv used for this project was the 2015 version. This dataset contained over 500,000 emails from approximately 150 Enron Corporation

employees. Due to the size of the dataset and what resources were available to us, a random sample of 12,000 emails were selected to use throughout the project. This small selection was taken to keep our training and inference times at a suitable length for this project.

3.2. Pre-Processing

The Enron email dataset was split into two columns; the 'From' data to represent the senders and the 'Body' data to encapsulate the content of the emails. The emails were randomly extracted for our 12,000 email sample containing the top 20 senders to provide enough data for each class to avoid underfitting the model. The "stratisfy" parameter on the train_test_split function was used as an attempt to stop class biasness. This selected dataset was split into 60% training, 20% validation, and 20% test sets.

For the K-Nearest Neighbours and Latent Dirichlet Allocation methods, the raw data was processed into Bag of Words using the default 'English' stop words dictionary. We chose to exclude any words that occur fewer than 0.01% and over 25% of the documents. This is to exclude rare words as well as very common ones that may result in inaccurate clusterings. CountVectorizer was used for this as it provides pre-processing functions such as punctuation removals and updating all values to lowercase. This method resulted in 78289 features for our training data and 73289 for both validation and testing data.

Stop words were not used for LSTM to avoid disruption to the order of words in emails. The classes (y) were converted into one-hot vectors via LabelBinarizer, and body (x) was tokenized using Tokenizer and padded to be the same lengths using pad_sequences.

4.0. Methodology

4.1. K-Nearest Neighbours

KNN is the simplest non-deep learning method used for classification. As this method does not include an independent training step the computation requirements for this method only occur when the classification is underway (Kaur & Kumar, 2015). This allows the method to execute quickly and use the least amount of energy to complete (Murugavel & Santhi, 2020). Though, this method still requires a larger amount of memory as everything is being computed during one step. The "nearest neighbour" is determined by its euclidean distance; the shorter the distance, the more similarities are present (Kaur & Kumar, 2015). This method is appropriate for the classification of emails as, due to these distances, the similarity of emails can be determined and which sender they belong to can be predicted.

We used the Grid Search selection method to determine the best hyperparameters for our KNN model. By using this method all possible combinations of the

hyperparameters *k* and *weight* could be searched through to determine which values would be best for our model. The range used for the possible *k* values was 1-200 to capture a large range of possibilities, and the options for the *weights* were either *'distance'* or *'uniform'*. Through this selection method it was found the best values were a *k* of 4 and a *weight* of *'distance'*.

4.2. Long Short-Term Memory

LSTM is an unsupervised Recurrent Neural Network classification method that processes sequences, prioritising order. Since text data such as emails are sequences of words, this method was considered. However, as order is a priority, no stop words were removed. The email bodies were tokenized using Keras' Tokenizer() to convert the words into sequences of integers each representing that memorised word in a dictionary of size 50000 (max_words). It was found that changing the maximum number of unique words didn't affect performance, as long as there's enough to record all the valuable ones, which 50000 seems enough. The class labels or senders' email addresses were converted into one-hot vectors of size 20 using Sklearn's LabelBinarizer() for multi-class classification. The ideal maximum length of sequences was found empirically, where 250 seems to provide best performance. This parameter wasn't set to the longest sequence of the datasets combined, because 15986 is too long, and will result in very long and impractical training times, especially when all other sequences will be padded to match that length. The end of sequences were padded with 0's so that they're all the same length, and these will be the inputs given to the model's input layer.

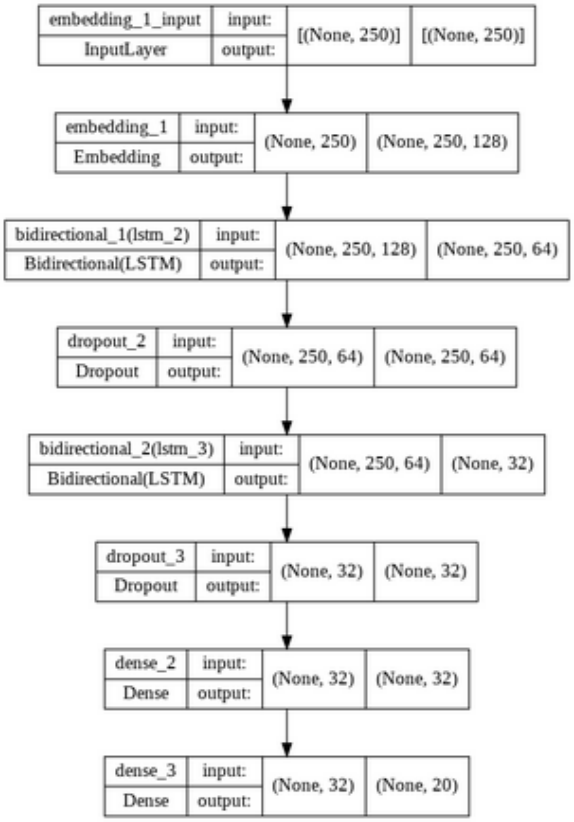


Figure 1: LSTM Model Diagram

It can be seen, in *Figure 1* above, that the model's structure starts with an embedding input layer receiving input sizes of 250, which are passed onto the first bidirectional LSTM layer of 32 units. The first LSTM layer returns sequences (True), and is accompanied by a dropout layer of 0.2 to avoid overfitting early on. The outputs of this layer become the inputs of the second bidirectional LSTM layer of 16 units, and doesn't return sequences (False) as it's the final LSTM layer, and it's also followed by a dropout layer(0.2). A dense layer of size 32 is used to convert the vector sizes down, and is followed by an output dense layer of size 20 using the sigmoid method to predict the sender or class based on the one-hot vectors of size 20. Using this model, a batch size of 128 was used for a slow training rate, given that there's 7200 samples in the training set, to avoid the model skipping over important information at the start. The model was trained for 350 epochs using the training set, where this parameter was found empirically and tuned with the validation set, so that convergence is reached for the training loss, indicating that there isn't much left to learn on the training set. After completion of training, the model is evaluated on the testing set, producing one confusion matrix for training, and one for testing. A loss graph was also drawn, and these were used for analysis.

Unlike the related works for LSTM, Bidirectional LSTM layers were chosen over regular LSTM layers, as empirical testing has shown that B-LSTM layers produce a higher accuracy on the validation and test set. This may be due to backpropagation, resulting in important information being recycled for further learning. Use of B-LSTM layers result in longer training times though, as data is moved back through the unit chains. These LSTM layers are stacked as this has shown better performance empirically, which is similar to what the related works' found. This may be due to the input into the second LSTM layer being the output of the first, allowing the model to create a more complex representation of the input sequences. Hence, better feature extraction or learning predict the sender, but also at the cost of training time.

4.3. Latent Dirichlet Allocation

Latent Dirichlet Allocation is an unsupervised topic modelling method for classification of documents. LDA is a generative probabilistic model of a corpus where documents are represented as a random mixture over latent topics.Each of these topics are characterized by a distribution over words(Blei, Ng and Jordan, 2003). The method is a technique used to classify text in a document to a particular hidden topic known as latent. The algorithm cycles through each document and randomly assigns each word in a document to one of the specified number of topics.

The hyperparameters for this model is the number of topics that is used for fitting the model, cross-validation can be used here to find the most optimal topic number based on the dataset perplexity value. Perplexity is a measure of how well a probability model predicts a sample, choosing the number of topics that produces the smallest value of perplexity will result in better generalization and results ("Evaluate Topic Models: Latent Dirichlet Allocation (LDA)", 2022). As the dataset involves emails, there is a large distribution over words in the dataset. The initial value of topics was chosen at 300 to explore how much perplexity increased and decreased as the topics increased in value. Referring to appendix 3.2.A, this graph shows that the most optimal number of topics that results in low perplexity value is around 70 and 80 topics, however there is a bug with the sklearns perplexity code where the value goes up when it should go down. With this in mind, the method has been explored with 10 topics, 70 topics and 300 topics.

The final number of topic values chosen to train this model is 300 as this number gave the best results for both training and testing.

This method can be explored further by utilizing LDA to extract topics then using this as features to fit a SVM for document classification. Grid Search was used to determine the most optimal hyperparameters for this dataset and the final values for the SVM model are C = 1200, Gamma = 0.1 using the RBF Gaussian kernel.

Latent Dirichlet Allocation is an appropriate approach for working with emails as a dataset. The method has been used for email subject classification as well as

exploring fraud detection based on the latent topics found in the data. For the project problem of finding sender from the email message, a supervised method such as k - nearest neighbours or SVM should be used to classify the senders based on terms found by LDA term extraction. For this method, we have chosen to explore SVM with LDA.

5.0. Evaluation and Discussion

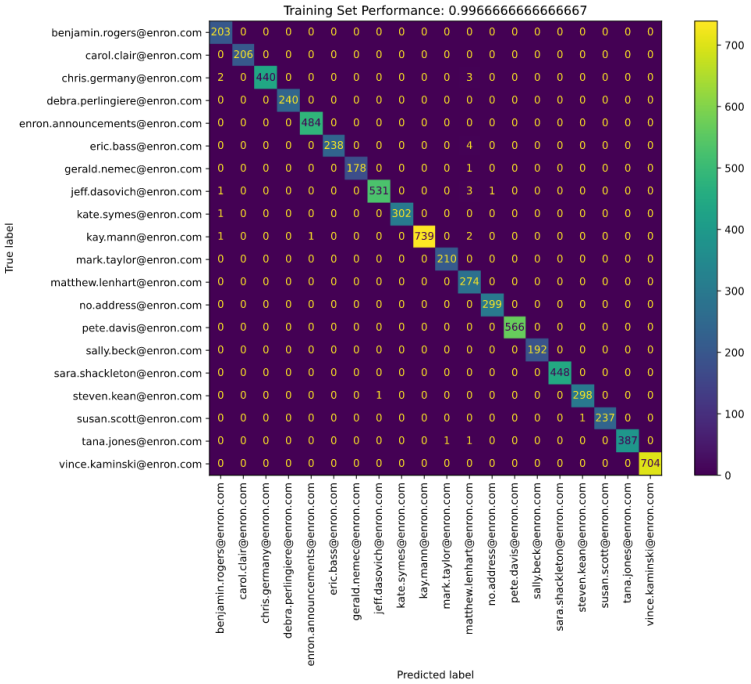
5.1. Results

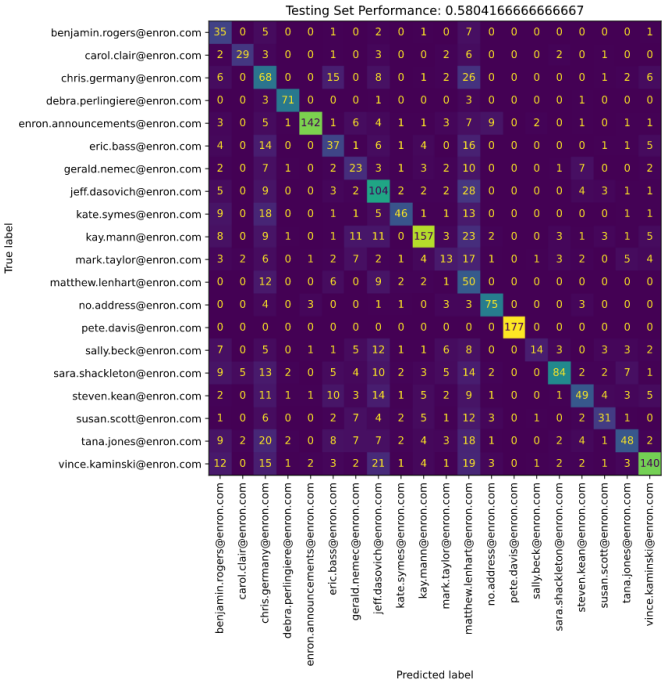
KNN Results

The final confusion matrices produced from the KNN method can be seen in *Figure 2* below. Here, it can be seen that the overall accuracy for the training set was approximately 58.04% accurate. The sender that was predicted correctly the majority of the time was Pete Davis at 177 times (100% accuracy, as shown in the normalised confusion matrix). Enron Announcements followed that at 142 times (90% accurate), then No Address at 75 times (81% accurate). The sender predicted with the lowest accuracy was Mark Taylor at 13 times (18% accurate). A possible reason for this range in accuracy could be how these senders write and sign off their emails. It could also be due to a class imbalance in the data. Effort was taken when processing the data to ensure there was no class imbalance but when looking at the confusion matrix the least accurately predicted sender, Mark Taylor, sent 210 emails while the most accurate, Pete Davis, sent 566. There were also senders, such as Kay Mann who sent up to 739 emails. Due to these imbalances the KNN method does not accurately portray the dataset.

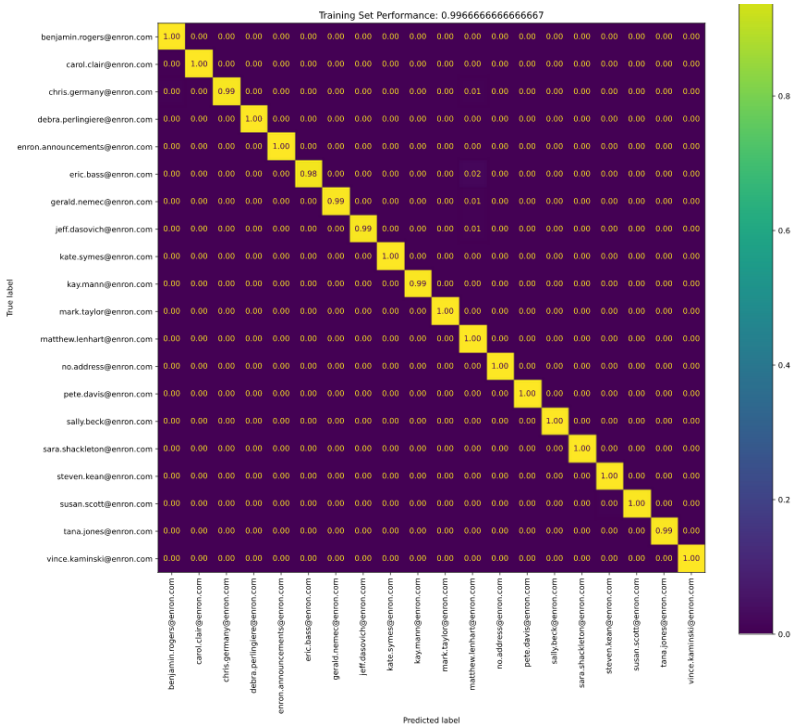
ASSIGNMENT 2: ENRON E-MAIL CLASSIFICATION

Non-Normalised





Normalised



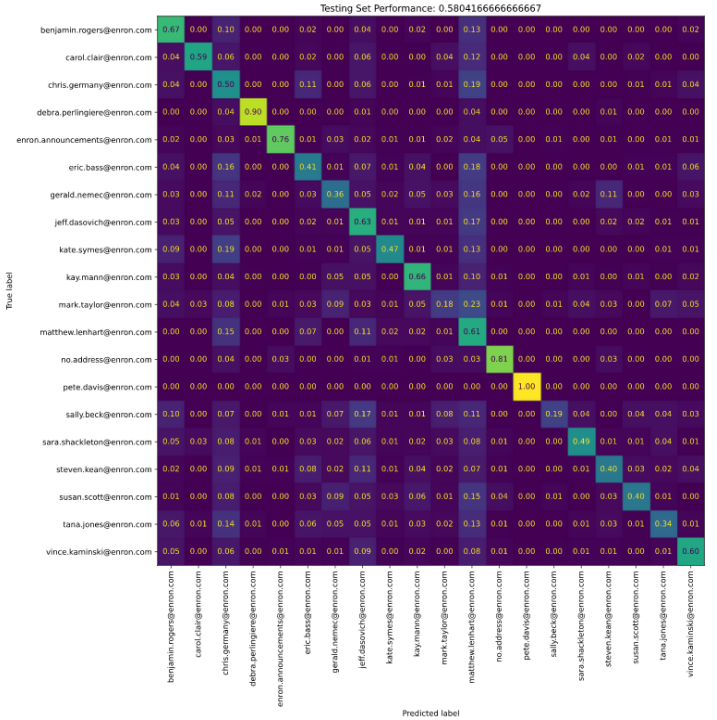


Figure 2: The confusion matrices displaying the results non-normalised and normalised of the KNN method.

LSTM Results

The results in *Figure 3* below show that the LSTM model trained until ~99.7% accuracy, but it could imply overfitting as a diagonal line from top left to bottom right is shown in the confusion matrix, so an analysis of the loss graph should be done. The Loss-Accuracy graph, in *Figure 4* below, where the rising trend for val_loss starts around 10 epochs indicating when overfitting begins. At 250 epochs convergence has been reached for the training loss, suggesting there isn't much left for the model to learn. The val_loss curve appears to be sporadic, suggesting that the model has learnt noise details from the training data, as it's overfitting.

From the confusion matrix for the testing set in *Figure 3*, the model identifies Vince Kaminski as the sender correctly the most at 188 times, followed by Kay Mann 182 times, then Pete Davis 177 times. Mark Taylor is identified correctly the least at 14 times. It's possible that these senders sign off their email with their name at the end often. The model confuses Jeff Dasovich and Steven Kean the most, this could be due to their emails being similar, forwarding of emails from each other, or just similar writing habits. Overall, the testing set has an accuracy of 70.3%, meaning that it correctly predicts the sender 70.3% of the time. It should also be taken into consideration that not all classes are evenly balanced, so some classes may have more correctly predicted than others, even though those other classes do not have many incorrect classifications.

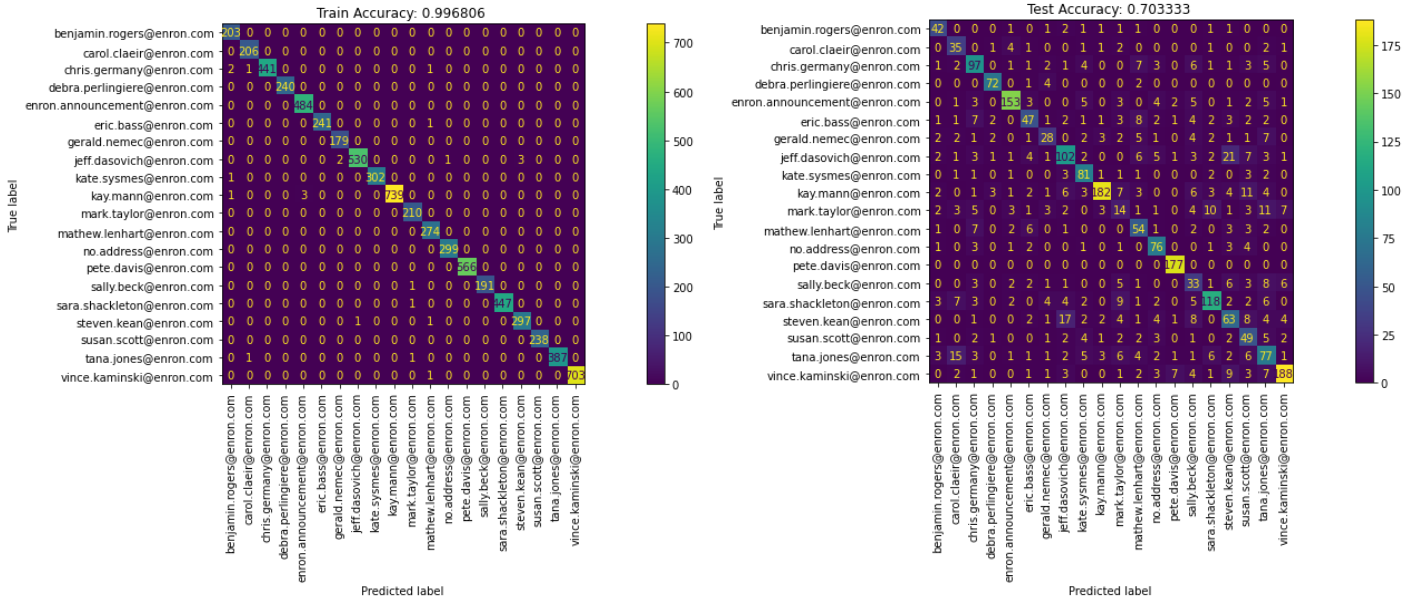


Figure 3: The confusion matrices displaying the results of the LSTM method with 128 batch size and 350 epochs.

Training Set performance: ~0.997, Testing Set Performance: ~0.703

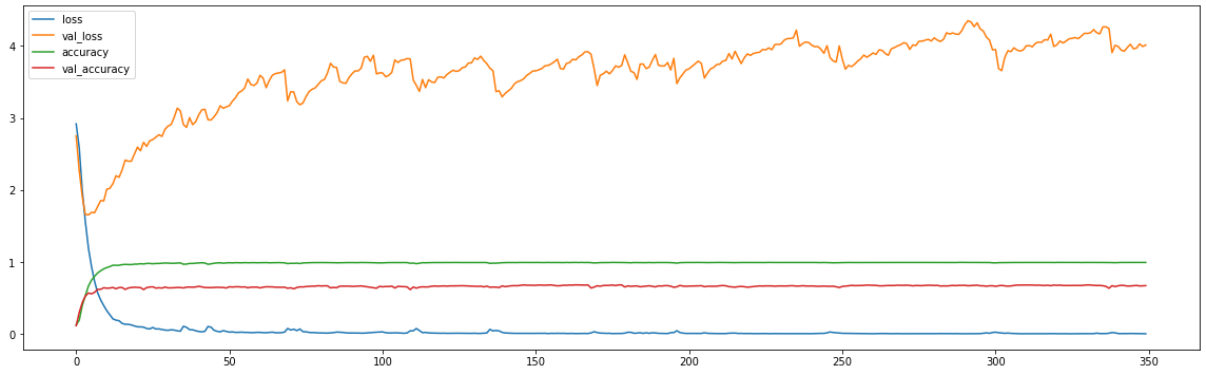


Figure 4: LSTM Loss - Accuracy Graph

LDA\SVM Results

The results of the LDA and SVM method in *Figure 5* shows that the dataset may be overfitting as the testing set performance is around 61.9% and training set performance is around 84% accuracy.

There are senders that the model is able to differentiate better from the rest of the dataset such as Pete.Davis@enron.com. Looking at both results, the model was able to process new data and predict the sender as Pete.Davis@enron.com correctly with no confusion with the other senders. Debra.Perlingiene@enron.com is another sender that seems to perform well, in the training results, the accuracy predictions for this sender may be low however this could be due to class imbalances where there was less data for this user compared to the others. The testing results for debra.perlingiere@enron.com predicted only 76 emails that belonged to the sender however there is less confusion with this sender compared to the others, however not as accurate as Pete.Davis@enron.com.

Pete.Davis@enron.com has a f1-score of 1 and Debra.Perlingiene@enron.com has a f1-score of 0.89 and a precision score of 0.87 on the testing data set. The assumption can be made that the accuracy for these users is high due to the way they write their emails and that it may be consistent for each email they send. It can also be assumed that they may sign their name off at the end of each email which LDA has picked up as a topic feature for the documents.

Matthew.lenhart@enron.com seems to be the most difficult to predict and differentiate as this sender gets predicted as the other users a lot more. This sender has a F1-score of 0.27 and a precision score of 0.18. With these results, we can assume that this sender does not follow a consistent way of writing up their emails and that they have very broad topics which makes it hard for the model to group their

emails together as well as differentiate from the other senders. The samples for this user may also be lower compared to the others which could also mean that the model did not have enough or valuable data to train on to find key topics for this user to allow for more accurate predictions.

The model has an overall F1-score accuracy of 0.62. This method does not perform badly however at this stage, it is still not very accurate. The model can be further improved by fine-tuning the parameters even more as well as data preprocessing can be further explored before it is used to train on the LDA model.

Compared to the third article, the current model has lower accuracy then a model that is trained on phishing and non phishing data. This existing model was trained to predict impersonators and had a 88.1% impersonated discovery rate (Ramanathan, Venkatesh & Wechsler, Harry, 2013) while the model to predict senders is only 62% accurate. We can assume it is a lot easier to classify emails as either fraudulent or not as the key topics found are not as broad as trying to find key topics for a person.

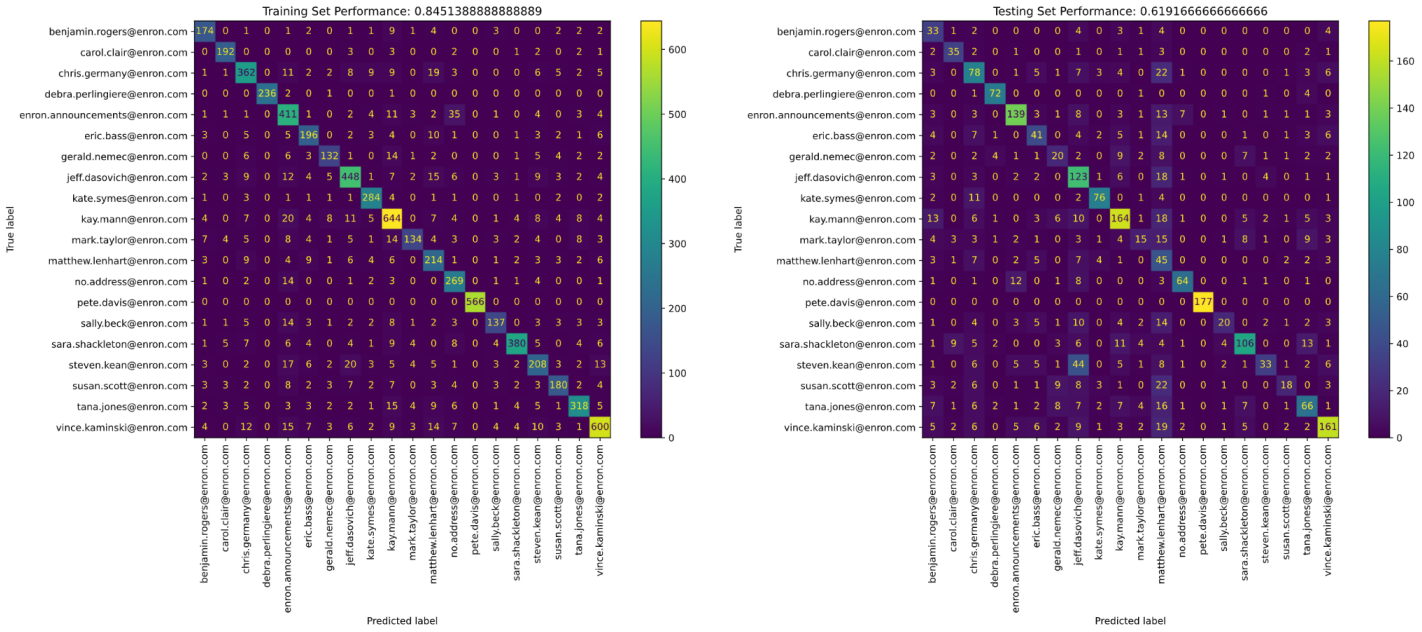


Figure 5: The confusion matrices displaying the results of the LDA method utilizing SVM with 300 Topics. Training Set performance: ~0.845, Testing Set Performance: ~0.619

5.2. Performance

The KNN model performs worse than the existing approach seen in *Spam Detection Using KNN, Back Propagation and Recurrent Neural Network* (Kaur & Kumar, 2015). As stated, the overall accuracy of our KNN model was 58.04% where the overall accuracy of Kaur and Kumar's model was 75%. This is likely due to the dissimilarities of these methods as our method used BoW where they used a step-by-step process to extract features, perform stemming, remove stop words, remove noise, and establish word frequencies. They also used a different dataset which may have caused even further differences in the methods.

This problem's LSTM model performs worse than the existing approaches, possibly due to a number of reasons. The existing approaches' problem space involves binary classification of emails into "to do" and "to read" classes, while this model classifies emails into 20 classes, one for each sender as it's a multi-class classification. The existing approaches may have less vocabulary of words to track for identification of "reading" and "doing" tasks, while this method needs to use the entirety of each sequence to predict who sent email, especially when it isn't signed off by the sender's name. The existing approaches also use the whole Enron Email dataset, while this method only uses 7200 samples for training, 2400 each for validation and testing sets, which is 12000 samples total compared to their 500000 samples.

The LDA\SVM method extends on existing approaches to try and explore latents topics related to a sender from an email. The goal was to use LDA to discover these topics then an SVM for classification of these topics to a sender, however the accuracy compared to the existing model is lower.

LDA is a popular topic modeling technique that finds latent topics in a dataset and can be utilized with lemmatization to group together different inflected forms of words into a single word with the same meaning. This is a simpler method as contents of an email can be processed and lemmantisated into a one key topic which then can be used as the classification. Our approach shows that it's more difficult to classify topics found in a document to a sender as the patterns used for a person are a lot more broad then the subject or intended goal of an email. This model may require more data to be trained on for each user to be able to discover the latent topic patterns related to how a user writes emails. However this is a constraint as the team does not have the right resources to perform training on larger datasets.

Overall, LSTM has the highest performance at 70.3% accuracy on the testing set, followed by LDA with 61.9%, then KNN at 58.04% LSTM performed the best as it's a deep learning method that is capable of propagating important information of each email up the cell chains, as well as returning such information backwards through the cells as well. It can also update and forget features, allowing it to only retain the most

important information to produce better outputs. It's also capable of creating complex representations of the emails, since it uses a deep neural network, and its training parameter is adjustable. In comparison, LDA\SVM has better accuracy score on the testing dataset compared to KNN as it uses two methods to extract latent topics from patterns in the emails and then passed on as features to the SVM model to classify these into senders. However this method heavily relies on emails having good data that can relate back to a sender by key topics.

KNN performs the worst, possibly due to its simplicity where its classification is based entirely on its calculation for Euclidean distances between the data points. Therefore, it is prone to classifying incorrectly as there are points that could be edge cases or fall between more than one class.

5.3. Computational Demands and Other Considerations

Method	Training Time	Inference Times (seconds)
KNN	1 mins 38 secs	4.81
LSTM	19 mins 7 secs	14.68
LDA \ SVM	11 mins 5 secs	25.00

Table 2: The training and inference times for each method.

Comparing the three methods, KNN trains the fastest, followed by LDA/SVM, and LSTM trains the slowest, as can be seen in *Table 2* above. KNN is not a deep learning method so it doesn't have an independent training step nor does it need to train for a particular number of epochs. LDA is slower than KNN, because training times are related to how large the corpus size that is being used. The corpus is the set of documents and for the current dataset that is 7200 emails. LDA model takes a parameter that indicates a number of features that should be returned. This parameter is important for training this model as an optimal value should be used to see accurate results. This adds on to the training time of the model. LSTM is the slowest in this case, because it's stacking two bidirectional LSTM layers. Stacking LSTM layers already makes training time longer, but using bidirectional LSTM layers will result in even longer training times especially when they're stacked, as a result of data propagation and backpropagation. It's also a deep learning method, where it trains a larger number of epochs in this case to ensure better performance for predicting. Therefore, LSTM may be the most computationally demanding, in fact there were situations where Google Colab would crash as it runs out of memory. This occurrence happens most often when using too large or a dataset, too large of sequences, and having too many units in the LSTM layers.

Similarly, KNN has the fastest inference time as it has less to account for, followed by LSTM, then LDA/SVM. LDA is the slowest as it uses the SVM for the final step of classification, this contributes to the inference time as SVM is slower to train depending on the size of the training sample and dimensions.

6.0. Conclusions and Future Works

KNN did not prove to be a viable method to use for the objective of predicting sender from emails, however it can be used in a multilayer soft classification method with the latent dirichlet allocation to further improve the overall accuracy.

After exploring our current dataset with LDA and SVM produced okay results however the accuracy can be further improved. It may be beneficial to train this model on a larger dataset to extract key topics per sender. As SVM is not a suitable method for large training datasets, KNN may be more appropriate or the random forest classifier can also be explored. LDA will be used as the initial model to compute and extract key topics for each sender and the results will be used to train either KNN and Random Forest.

From these methods, it's clear that LSTM performed the best, and that it's ideal for text classification, since it uses sequences and prioritises order which is perfect for an analysis of emails. In the future, we should look to use this LSTM model with a pre-trained word embedding like ELMo as it may further improve its performance. It may produce better performance if the LSTM model is pretrained. This was found by the existing approaches, despite them using regular LSTM layers. Those regular LSTM layers should be swapped for bidirectional LSTM layers, as this approach has found that the latter has better performance. However, it's computationally heavier, meaning longer training times, so that should be considered.

7.0. Bibliography

Ahlstrand, J., & Rosander, O. (2017). *Email Classification with Machine Learning and Word Embeddings for Improved Customer Support*. Blekinge Tekniska Hogskola. http://www.diva-portal.org/smash/get/diva2:1189491/FULLTEXT01.pdf?fbclid=IwAR0_7buh3AaIcTxFwFDq5JCDwAyp1CysbhCIwdNGYHqyk85iVxykeLunOnM

Alibadi, Z. Vidal, J. (2018). To Read or To Do? That's The Task: Using Transfer Learning to Detect the Intent of an Email. Retrieved 13 June 2022 from

https://ieeexplore.ieee.org/abstract/document/8947885

Alibadi, Z. Du, M. Vidal, J M. (2019). Using Pre-trained Embeddings to Detect the Intent of an Email. Retrieved 13 June 2022 from

https://dl.acm.org/doi/abs/10.1145/3325291.33253

Alibadi, Z. (2021). Detecting the Intent of Email Using Embeddings, Deep Learning and Transfer Learning. Retrieved 13 June 2022 from

https://scholarcommons.sc.edu/etd/6384/

Blei, D., Ng, A. and Jordan, M., 2003. *Journal of Machine Learning Research 3 (2003) 993-1022*, [online] 3, pp.993-1022. Available at: https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

Evaluate Topic Models: Latent Dirichlet Allocation (LDA). (2022). https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0

Gong, H., You, F., Guan, X., Cao, Y., & Lai, S. (2018). Application of LDA Topic Model in E-Mail Subject Classification. Advances In Intelligent Systems Research, 161, 1-7. https://www.atlantis-press.com/article/25907231.pdf.

Harwath, D., Johri, N., & Yin, E. (2022). AN UNSUPERVISED APPROACH TO EMAIL LABEL SUGGESTION, 1-5.

http://cs229.stanford.edu/proj2010/HarwathJohriYin-AnUnsupervisedApproachToEmailLabelSuggestions.pdf

Kaur, K., & Kumar, M. (2015). Spam Detection Using KNN, Back Propagation and Recurrent Neural Network. *International Journal of Engineering Research & Technology, 4(9)*, 557-564. https://www.ijert.org/research/spam-detection-using-knn-back-propagation-and-recurrent-neural-network-IJERTV4IS090492.pdf

Khalid, Y., Ali, S. A.,& Naser, M. A. (2019). Spam Image Email Filtering Using K-NN and SVM. *International Journal of Electrical and Computer Engineering, 9(1)*, 245-254. https://www.researchgate.net/publication/332546034_Spam_image_email_filtering_using_K-NN_and_SVM

Murugavel, U., & Santhi, R. (2020). K-Nearest Neighbor Classification of E-Mail Messages for Spam Detection. *ICTACT Journal on Soft Computing, 11(1)*, 2218-2221. http://ictactjournals.in/paper/IJSC_Vol_11_Iss_1_Paper_5_2218_2221.pdf

Palaniswamy, H. (2015). Exploratory Data Analysis of Enron Emails. Retrieved 13 June 2022, from https://www.stat.berkeley.edu/~aldous/Research/Ugrad/HarishKumarReport.pdf.

Ramanathan, Venkatesh & Wechsler, Harry. (2013). Phishing detection and impersonated entity discovery using Conditional Random Field and Latent Dirichlet Allocation. Computers & Security. 34. 123–139. 10.1016/j.cose.2012.12.002. https://www.researchgate.net/publication/257006827_Phishing_detection_and_impersonated_entity_discovery_using_Conditional_Random_Field_and_Latent_Dirichlet_Allocation

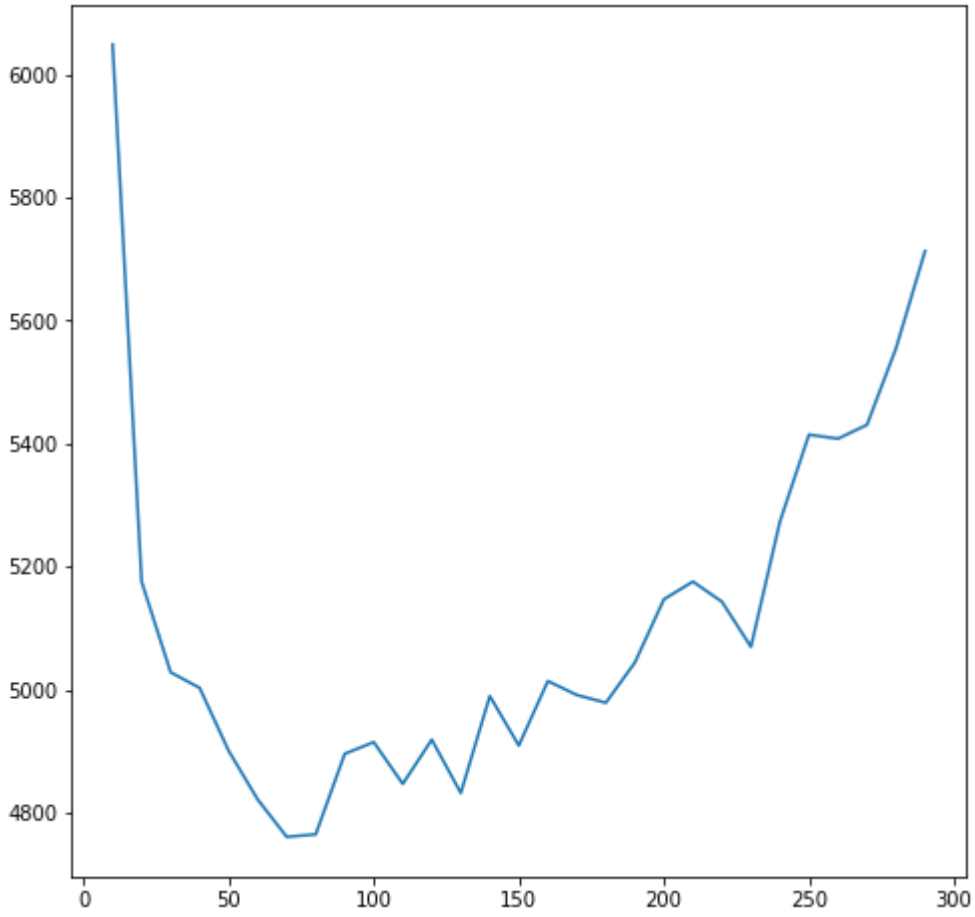
8.0. Appendix

8.1. Team Contribution

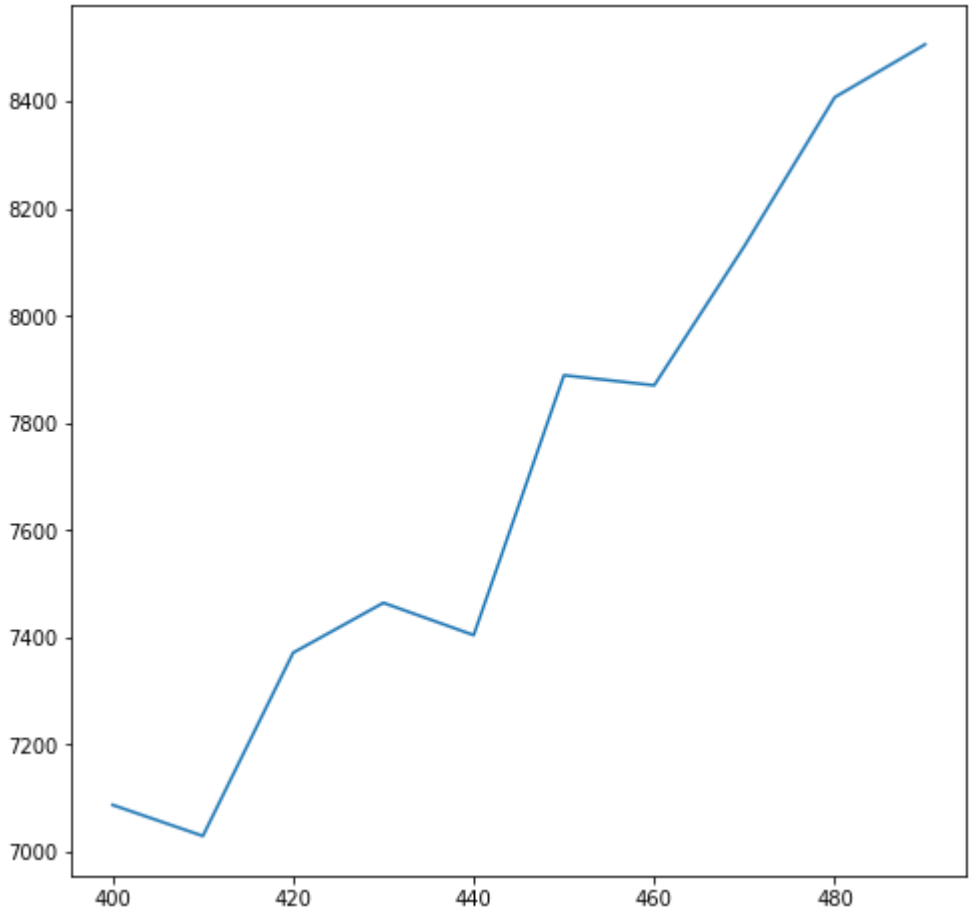
Student Name	Their Contribution	Contribution Percentage	Signature
Sasha Le	LDA	33.33%	S.L
Ash Phillips	KNN	33.33%	A.P.
Raymond Thach	LSTM	33.33%	R.T

8.2. Latent Dirichlet Allocation

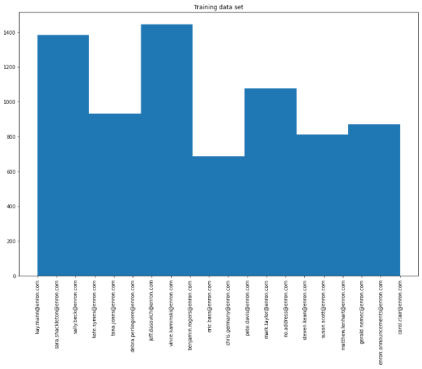
8.2.A Perplexity Graph - 300 topics

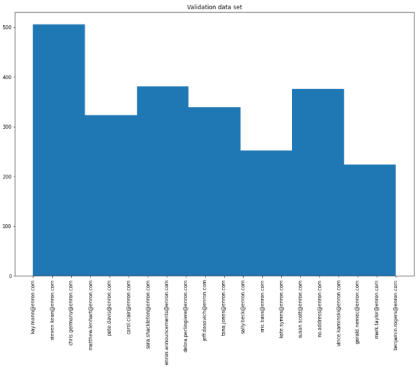


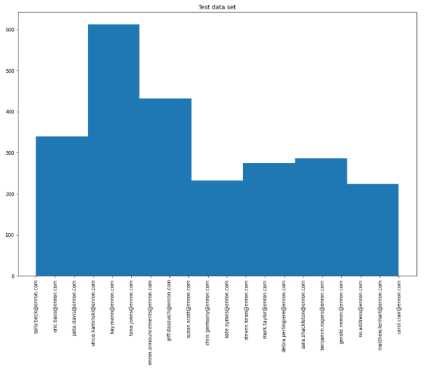
8.2.B Perplexity Graph - 500 topics



8.3. Class Imbalance







8.4. Code

All code used was built upon using code from Lecture examples and Tutorial solutions.