

CAB 420 Assignment 2

Enron Email Classification

Group 27

Due date: 6th June 2021

Group Member:

	Name	Student Number
1	Shu Du	N10505024
2	Haonan Jiang	N10533001
3	Zhiyuan Jiang	N10541683
4	Jiyan Zhu	N10415483

Table of Contents

Title Page	1
Table of Contents	2
Introduction / Motivation	3
Related Works	4
LSTM.....	4
Naïve Bayes Classifier	4
Support Vector Machine.....	5
Random Forest.....	5
Transformer	5
Data.....	6
The data	6
Data processing.....	7
Methodology.....	9
SVM	9
Bidirectional RNN.....	10
Transformers.....	14
Evaluation / Discussion.....	16
Conclusion	18
Bibliography	19
Appendix.....	20

Introduction / Motivation

As the fast growth of modern technology, machine learning has already become an indispensable part of people's everyday life. By successfully integrating machine learning techniques into a range of different areas (medical, financial, etc.), it can not only simplify the complex tasks, but also bring an enormous number of benefits to the modern society. One typical topic in machine learning is classification, which refers to categorize a dataset into a different number of classes. This technique has also been widely applied in different research areas and industry projects to solve problems such as identify spam emails, face recognition etc.

In this project, an investigation will be conducted into the Enron email dataset, which contains around 0.5 million messages sent by 150 senior management users from the Enron company. The dataset was initially generated by the US Federal Energy Regulatory Commission (FERC) according to the Enron bankruptcy investigation and has been used for academic research extensively in the past. *(Purtill, C. (2019)).*

During the investigation of this project, it aims to classify the text contains in the email by discovering existing patterns such that the sender of the email will be detected. This is a representative classification problem that can be further used in various real-life applications to benefit society. For instance, by training a model to classify the sender of the emails accurately and concisely, it can help the organizations manage the employees more effectively and reduce the risk of spamming and phishing such that it further protects the confidentiality of the sensitive data.

As the dataset was originally generated in 2004, many machine-learning-based research, such as text classification and spamming identification, has already been conducted. However, the goal of this project is to explore different classification algorithms to bring people a better understanding of the significant practices of machine learning. In the meantime, it is further expected to discover new perspectives for improvements of classification tasks using machine learning and further benefits the society potentially.

Therefore, by utilizing the Enron Corpus dataset, a range of different algorithms will be developed to train the models for predicting the email senders according to the text contained. Furthermore, comparisons and evaluations based on different factors such as accuracy and efficiency will also be conducted to different developed approaches throughout the investigation. In this way, it could help select the best solution to apply in real-life situations under different circumstances.

Related work

In order to perform the investigation successfully, it is essential to explore the existing approaches on this topic such that it will further help to design the solutions in a more effective way. Hence, several approaches that specific to this investigation will be explored and evaluated in this section.

Approach 1: Word embedding and Long Short-Term Memory (LSTM) network

One suitable approach found is to apply word embedding to process the text stream and then use a sequential model that contains LSTM layers to train the model from the processed data. Word embedding is a widespread technique for text representation which has the capability of capturing the semantic and synaptic similarity within the text. This can further actively position the words with a similar meaning closer spatially (*Karani, D. (2020)*).

A sequential model refers to a particular plain stack of layers network structure where each layer has one input and output tensor explicitly. However, this further results from the model with the limitation of not having the ability of training with multiple inputs or outputs (*Keras. (2020)*). LSTM is a type of recurrent neural network that can actively bring long-term dependency, which refers to using information learned previously during the loop in the current task.

Thus, after applying Google's pre-defined word embedding function `word2vec` and then train the model using a LSTM network, an accuracy of 0.7367 has been achieved successfully using this approach (*Ankur, K. (2020).*).

Tokenization is another typical method for process text streams. The tokenizer is used to convert the text stream into tokens by using whitespace as the separator. Then by converting it into numerical representation and inputted into a pre-defined sequential model, a model is trained successfully with the capability of classified the sender according to the body part of the email (*Analytics Vidhya. (2020).*).

Approach 2: Naive Bayes Classifier

Naive Bayes classifier is also a potential technique that has been found to determine the sender of the emails. In the approach, the data was first loaded by using the `Email.Parser()` function, which allows extracting email data into different fields such that the unstructured email data can be handled in a more organized way.

Then the extracted data are processed using the `CountVectorize()` function, which can effectively convert the text into a vector representation of tokens. In this way, the raw data is well-prepared to be used for model training.

Then by training the data using the Naïve Bayes algorithm, a typical text classification technique developed based on Bayes' Theorem. This particular method works under the assumption of each pair of variables within a class is conditional independent. Overall, after the model trained using the Naïve Bayes algorithm, a relatively low accuracy of 0.46 was achieved (*nowshad-sust. (2017)*).

There could be many possible reasons for this low achieved accuracy. One potential reason could be the variables are dependent on each other and further affect the performance on Bayes' Theorem of

conditional probability. In such case, Naïve Bayes classifier may not be an optimal algorithm for this project.

Approach 3: Support Vector Machines (SVM)

Another powerful approach that can be used to classify the senders is SVM, which is a classic supervised machine learning technique that works by separating classes using a hyperplane. This approach also handles data by using the *CountVectorize()* function to convert the text stream into a vector of tokens. Then by further using grid search to find the best parameter for SVM specific to the dataset, a hyperplane has been found to separate the classes, and resulted an accuracy of 0.85 (*nowshad-sust. (2017)*).

However, one limitation that can be found for this approach is that instead of using validation data to select the best hyperparameter, grid search is only capable of using training data to evaluate the parameters that have the best performance. Thus, if validation data could be used to select the best model parameters, a better accuracy could be achieved potentially as the parameters are chosen according to the unseen data.

Approach 4: Random Forest

Random forest is also a classic technique that can be used for text classification such that the email senders can be determined according to the body text of the emails. Random forest is a combination of decision trees where each tree will generate a set of rules to classify the input data, then by combining all the rules from each tree together, the random forest can be used effectively to predict the email senders (*Javed, M. (2020)*).

As there is an enormous amount of data stored in the Enron email dataset, this further suggests it requires a large amount of trees in order to achieve a reasonable performance and avoid the problems such as overfitting. Thus, time efficiency could be a considerable factor for implementing a random forest algorithm.

Approach 5: Transformer

A particular type of deep neural network known as the transformer is also a potential technique that could be applied for classifying the email senders. It combines convolutional neural network together with the attention models where attention can measure the importance of each word and further focus on the more significant words. However, as this technique was introduced recently, it was not able to find an approach that is specific to the Enron email dataset. Instead, an example approach of classifying free text into 25 possible classes was found. After fitting the data into a transformer model, the free test data was classified successfully with a reasonable accuracy (*Zhanna, T. (2020)*).

Although it has not found a transformer approach for this particular Enron email dataset, the example found still has a decent level of similarity with the Enron dataset. Thus, it further proves that the transformer is a valid technique that could be investigated for this project.

Assignment 2

Overall, after a thorough research on the different existing approaches that can be applied to the Enron email dataset, a better understanding has been gained successfully on different ways to perform the investigation for this project. In the above five different approaches researched, the first three approaches are the previous works that have been applied on the exact same dataset, while the last two approaches are the text classification works that was performed on the dataset with a certain level of similarity.

It can be found that some of the approaches have a relatively high accuracy; this could be potentially due to these approaches has narrowed down the dataset to the top 10 most frequent email senders. Hence, a relatively better accuracy was generated as the model only dealt with a small portion of the dataset in these cases.

Throughout the research on different previous works, different data processing techniques were also found together with the algorithms and can be used effectively for the later developments. For instance, the *Parser()* function is an excellent technique for loading the email data while word embedding, tokenization, and *CountVectorize()* are the valid techniques that can be applied to convert the text streams when training the model. Therefore, in this section, both data processing techniques and algorithms have been investigated successfully and can be further extended in the following sections for this project.

Data

The dataset

The Enron Email dataset used for this project is the version released on May 7th, 2015, and contains around 0.5 million of emails over the 150 users. Emails are categorised into different folders according to sender's name and each email is stored separately in a different file. Apart from the sent emails of the users, this dataset also has many other files contains information such as contacts, customer, invoice and etc. As the goal of this project is to investigation different algorithms for classifying the email senders, only the sent emails data within the dataset will be considered in this case.

```
'Message-ID: <1812944.1075857796215.JavaMail.evans@thyme>\n',  
'Date: Sat, 9 Sep 2000 15:01:00 -0700 (PDT)\n',  
'From: davidpsmith@att.net\n',  
'To: jamills@storm.ca\n',  
'Subject: new email address\n',  
'Cc: kzweifel@indiana.edu, john.zufferli@enron.com, kyounger@lmumail.lmu.edu  
'\tryan.wormley@ceridian.com, mike.wolfe@ernexinc.com, \n',  
'\tbrett.x.vowles@msg.ameritech.com, dvvalenti@aol.com, jtower@istar.ca, \n'  
'\tcrikstephens@hotmail.com, tsmith@agraee.com, swbb@nwlinc.com, \n',  
'\tjoshsilver@canada.com, jdsilver1@mail.sprint.ca, \n',  
'\tjoanna.silver@redklay.com, syscokid@bellsouth.net, \n',  
'\tseansh@attachmate.com, darcy@dialin.net, dshaver@pacbell.net, \n',  
'\tjrosen@cadiesel.com, bmraylor@mq.psd.k12.ca.us, john@poyser.com, \n',  
'\tkerry.pond@edu.gov.on.ca, lpeden@westarinsurance.com\n',  
'Mime-Version: 1.0\n',  
'Content-Type: text/plain; charset=us-ascii\n',  
'Content-Transfer-Encoding: 7bit\n',
```

Figure 1-example of email format

The figure on the right is a screenshot on how the emails data are stored in this dataset. It can be found that the email contains a range of different information such as ID, date, sender, receiver, content and etc. This particular consistent format is used for all the email data within this data. By considering the purpose of this project, it is decided to only use the contents of the emails to classify the senders. Although information such as date, time, receiver

could potential help to improve the classification tasks as each user may has its own preferences such as time, receivers when sending the emails, it is still decided to use contents only to perform the sender classification for the purpose of reducing the computational demands. Also, it is further expected that the contents of the emails can efficiently provide enough information to classify as each person should has its own word preferences when composing the emails.

Data Processing

After a thorough exploration of the dataset, the next step is to load all the sent email data into python for further processing. As mentioned, all the emails are stored in different files within different folders. A function called *get_files()* was built to get all the files in the given folder and all its subfolders. With all files found, we can then read each of them and extract the contents.

A single file is first to read in as a sample. By observation, it can be found that all the email senders are leading by 'X-From:' and tail by '\n'. While all the email contents leading by the end of the line 'x-FileName', and tail by either Forward by or Original message. Thus, two functions called *extract_sender()* and *extract_content()* was created explicitly to extract the sender and the contents of an email by using the regular expression according to the patterns found.

Integrating the three functions: *get_files()*, *extract_sender()* and *extract_content()* together, a function called *extract_data()* was built to process all the files with a given path. The function integrates each file and extracts the data. Consider the scale of the dataset, data from each file is stored in a dictionary then converted to a pandas data frame after all files processed. This increase the performance of the function significantly.

	sender	message
0	Phillip K Allen	Dave, \n Here are the names of the west desk m...
1	Phillip K Allen	Paula,\n 35 million is fine\nPhillip
2	Phillip K Allen	Brenda,\nPlease use the second check as the Oc...
3	Phillip K Allen	I think Fletch has a good CPA. I am still doi...
4	Phillip K Allen	Brenda,\n Please use the second check as my Oc...

Figure 2-Top of the data

	sender	message
475431	Zufferli, John </O=ENRON/OU=NA/CN=RECIPIENTS/C...	Some of my position is with the Alberta Term b...
475432	Zufferli, John </O=ENRON/OU=NA/CN=RECIPIENTS/C...	2\n
475433	Zufferli, John </O=ENRON/OU=NA/CN=RECIPIENTS/C...	Analyst\t\t\t\t\tRank\nStephane Brodeur\t\t\t1...
475434	Zufferli, John </O=ENRON/OU=NA/CN=RECIPIENTS/C...	i think the YMCA has a class that is for peopl...
475435	Zufferli, John </O=ENRON/OU=NA/CN=RECIPIENTS/C...	I will have 4 books:\nCAND-MGMT-BAS for all AE...

Figure 3-Bottom of the data

With all the files have been extracted successfully, we then further observed the extracted data. It shows in the above figure that 475436/517401 files are recorded in the data frame since we have filtered the emails with empty content already when processing. Content of emails is acceptably clean after cutting of the *Forward*, *Original*, and *HTML* contents.

Assignment 2

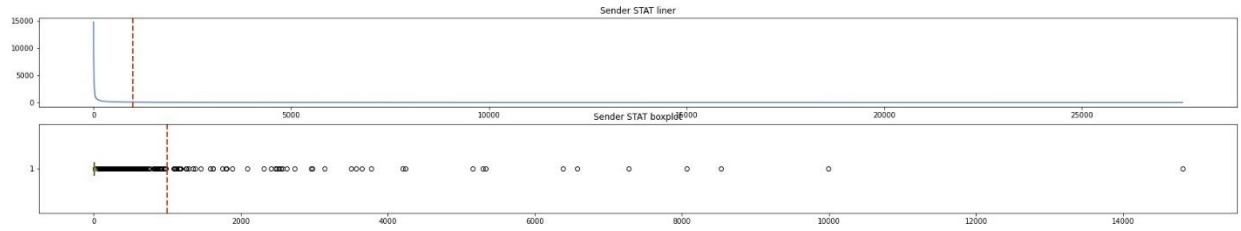


Figure 4-Box plot of senders' email number

However, by counting the number of emails from each sender, it can be observed that there is an enormous amount of sender hold very few emails. It is then decided to filter those senders to ensure the performance of future training. By plotting the box plot of each sender's email number, a threshold of 1000 is determined. Therefore, it further decided only to use the top 51 senders who sent the most amount of emails and then randomly draw 1000 emails from each sender. It further reduced the data to 51000 rows. In this way, it has effectively saved the computational demands when training the models.

The filtered data is saved to an HDF5 format for training use. As for the division of training, validating and testing data, a ratio of 6:2:2 is set to split them randomly.

Methodology

In this section, an extension of different algorithms will be investigated on the Enron dataset i

SVM

One effective technique that can be applied to classify the email sender is Support Vector Machines (SVM), which can separate all the text contents into different classes by finding a hyperplane. Each sender would use certain words more often or less than others, so this allows SVM to classify senders based on the occurrence of words in a message.

To successfully investigate the SVM, it is essential to perform word embedding on the text contents to convert texts into numerical vectors since the SVM algorithm does not support string format data. Bag of Words is further chosen as the word embedding technique which can encode the text as a histogram, measured by the word occurrences. This will guarantee sequences with exact size representation, but it will destroy all the information related to the order of the data in the meantime. The data contains rare words and everyday words that everyone uses, and those texts will give no help on the model; therefore, dropping words that only occurred 0.01% and words that occurred more than 25% will make the model easier to train. The purpose of this cluster approach is because it is sparse data with high dimension.

```
print(X_train_counts.shape)
print(train_X[0])
print(X_train_counts[0])
```

```
(30600, 19064)
```

```
I think he will just tell us that chicken nuggests are vegetables.
```

```
(0, 4244)    1
(0, 8801)    1
(0, 10119)   1
(0, 17144)   1
(0, 17298)   1
(0, 18066)   1
(0, 18194)   1
```

Figure 5- example of word emedding

SVM uses a hyperplane to separate all the data into classes, and those data points are each vector. SVM only uses a few vectors that support it to separate the points into classes; thus, if it can identify those few vectors, it will ignore all other dimensions, which makes SVM very efficient to handle sparse data. So, BOW embeddings are ideal for SVM classifier as it performs well on such a high dimensional sparse data.

In order to find the best parameter for SVM, a nested for loop was generated to fit the model with training data and used validation performance to determine the best parameters. This method is relatively better than the grid search technique as it uses training data itself to select the best parameter. Thus, it is further expected the model to gain a better accuracy by using the unseen data to determine the best hyper parameter.

Assignment 2

As a relative large dataset has been used, only a small range of parameters was used to find the best parameter in order to reduce the computational demands.

```
C = 0.1 class_weight = None kernel = linear val acc = 0.7559803921568627
C = 0.1 class_weight = None kernel = rbf gamma = 0.1 val acc = 0.22872549019607843
C = 0.1 class_weight = None kernel = rbf gamma = 0.01 val acc = 0.3949019607843137
C = 0.1 class_weight = None kernel = rbf gamma = 0.001 val acc = 0.26362745098039214
C = 1 class_weight = None kernel = linear val acc = 0.7427450980392157
C = 1 class_weight = None kernel = rbf gamma = 0.1 val acc = 0.5542156862745098
C = 1 class_weight = None kernel = rbf gamma = 0.01 val acc = 0.6806862745098039
C = 1 class_weight = None kernel = rbf gamma = 0.001 val acc = 0.566764705882353
C = 10 class_weight = None kernel = linear val acc = 0.7233333333333334
C = 10 class_weight = None kernel = rbf gamma = 0.1 val acc = 0.5755882352941176
C = 10 class_weight = None kernel = rbf gamma = 0.01 val acc = 0.7368627450980392
C = 10 class_weight = None kernel = rbf gamma = 0.001 val acc = 0.7181372549019608
C = 30 class_weight = None kernel = linear val acc = 0.717156862745098
C = 30 class_weight = None kernel = rbf gamma = 0.1 val acc = 0.5677450980392157
C = 30 class_weight = None kernel = rbf gamma = 0.01 val acc = 0.7340196078431372
C = 30 class_weight = None kernel = rbf gamma = 0.001 val acc = 0.7471568627450981
C = 50 class_weight = None kernel = linear val acc = 0.7170588235294117
C = 50 class_weight = None kernel = rbf gamma = 0.1 val acc = 0.5663725490196079
C = 50 class_weight = None kernel = rbf gamma = 0.01 val acc = 0.7244117647058823
C = 50 class_weight = None kernel = rbf gamma = 0.001 val acc = 0.7549019607843137
C = 100 class_weight = None kernel = linear val acc = 0.7161764705882353
C = 100 class_weight = None kernel = rbf gamma = 0.1 val acc = 0.5658823529411765
C = 100 class_weight = None kernel = rbf gamma = 0.01 val acc = 0.7156862745098039
C = 100 class_weight = None kernel = rbf gamma = 0.001 val acc = 0.7574509803921569
C: [0.1, 1, 10, 30, 50, 100]
class_weight: [None]
kernel: ['linear', 'rbf']
gamma: [0.1, 0.01, 0.001]
degree: [2, 3, 4, 5, 6]
Best hyper-parameters: C = 100 class_weight = None kernel = rbf gamma = 0.001
```

Figure 6-Nest for loop search

As shown in the above figure, after applying the search on different combinations of C, class_weight, kernel and gamma, it has been found that the model has achieved the highest validation accuracy (0.7574) when C= 100, class_weight = None, kernel = rbf and gamma = 0.001.

Therefore, after further fit the the model by using the best hyper parameter, a training accuracy of 0.93 and a testing accuracy of 0.75 was achieved successfully.

```
Training accuracy = 0.9277777777777778
Testing accuracy = 0.7481372549019608
```

Figure 7-SVM model accuracy

Bidirectional RNN

Deep learning classifiers can learn and understand the sequence of an input. Therefore, it captures the words each sender often uses and learns how they use the words in a sentence. In other words, it learns the writing style of each person. Recurrent Neural Network (RNN) is a class of artificial neural network where connections between nodes form a directed graph along a sequence, and it has been proved to be a robust neural network for sequenced text classification(Liu.P). The most advanced RNN architecture for sequenced text classification is the Bidirectional RNN architecture, a sequence to sequence model (Vaswani.A), superior to the Unidirectional RNN mentioned in the Related Work section (approach 1).

RNN processes information from inputs that have already passed through the model using the hidden state to learn a sequence. Unidirectional RNN only runs the sequence forwards. It stores information of the past because the only inputs it has seen are from the past. Furthermore, Bidirectional RNN runs the

sequence in a forward direction and backwards so that the information from the future can be preserved. By combining the two ways hidden states, the model can learn the information from both past and future to better capture the relationships within the sequence. Generally, Bidirectional RNN shows a better result as they understand the context better.(Junyoung.C)

Traditional RNN would experience the problem of vanishing gradients if the input sequence is getting longer and longer. However, using long-short term memory (LSTM) and Gated Recurrent Unit (GRU) can solve the gradient vanishing even if the sequence is extensive.

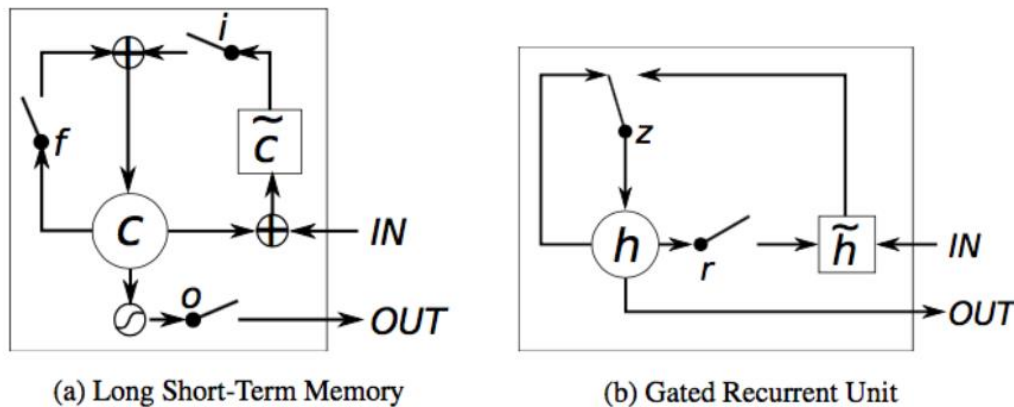


Figure 1: Illustration of (a) LSTM and (b) gated recurrent units. (a) i , f and o are the input, forget and output gates, respectively. c and \tilde{c} denote the memory cell and the new memory cell content. (b) r and z are the reset and update gates, and h and \tilde{h} are the activation and the candidate activation.

Figure 8 - LSTM and GRU

Both LSTM and GRU have the same goal of tracking long-term dependencies effectively while mitigating the vanishing/exploding gradient problems. The critical difference between a GRU and an LSTM is that a GRU has two gates (reset and update gates), whereas a LSTM has three gates (input, output and forget gates). The GRU controls the flow of information like the LSTM unit, but without using a memory unit. It just exposes the full hidden content without any control. When comparing GRU with LSTM, it performs good but may have a slight dip in accuracy. Nevertheless, still have a smaller number of trainable parameters, which makes it advantageous to use. So, we decided to use a stacked Bidirectional GRU as our RNN model.

Word Tokenization

Word tokens can be used as inputs for a LSTM RNN. Unlike BOW, which summarize how often a given word appears in a text, Tokenization turns a text or sentences into a sequence of integers so that it remembers the position of each word within a text (figure 9). First, we used TensorFlow tokenizer to break a stream text into a list of words or sentences. Then, we limited the dataset to the top 50,000 words and set the maximum number of words of each email message at 300 as the majority of the message are less than 300 (figure 2). Finally, truncated and padded the input sequences to be all in the same length for modelling.

Assignment 2

```
print(train_X[1])
print(train_padded_X[1])
```

I was wondering if you were there. Hope the weather is better there than in Houston. Is there any water in the bay?
Love,
Kay
Bill and Rita McCall <bmccl@wcnet.net> on 10/10/2000 11:06:58 AM
To: Kay.Mann@enron.com
cc:
Subject: Palacios calling
Hi Kay,Kay. Hope all of you are staying warm. We are looking for Jo and Rich in a few minutes. Just wanted you to know we are here. Love to all,Mother

```
[ 7 58 1988 26 9 170 85 327 1 867 12 569
 85 176 8 123 12 85 45 1727 8 1 4341 656
402 438 3 3152 11549 15437 15438 457 13 63 63 47
80 347 1266 39 2 402 1091 6 27 41 37 10015
1769 642 402 402 327 38 4 9 24 2857 4437 20
24 563 10 4310 3 3212 8 5 517 921 99 525
9 2 53 20 24 147 656 2 38 3658 0 0
0 0 0 0 0 0 0 0 0 0 0 0]
```

Figure 9- Sample output of tokenized email message

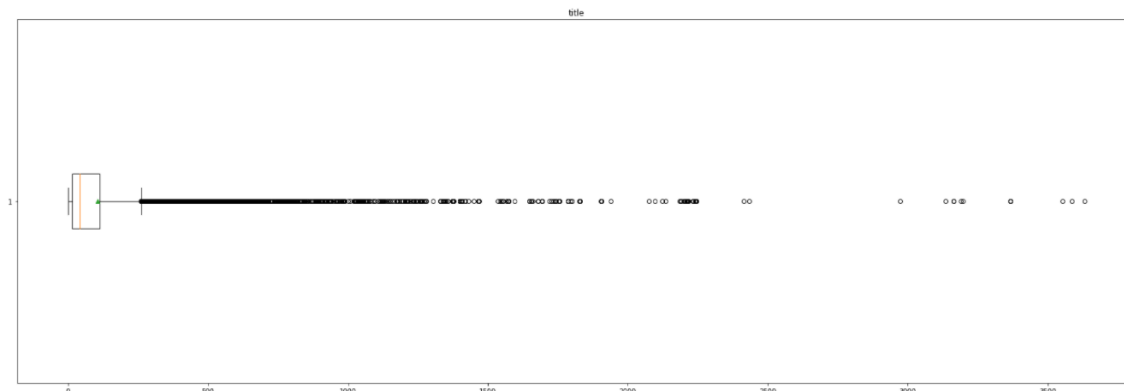


Figure 10- Boxplot shows the distribution of message lengths

Model Architecture

The model shows as below. In the input layer, the inputs are the tokenised email message in 300 lengths. In each Bidirectional layer, each sequence is analysed forwards and backwards and then passed to the fully connected dense layers with 51 units output using SoftMax activation. Two dropout layers are used to reduce overfitting. Finally, we compile the model with sparse categorical cross-entropy loss function and trained 20 epochs with batch size 64. The final test accuracy was 81% and 1.2 for test loss.

Assignment 2

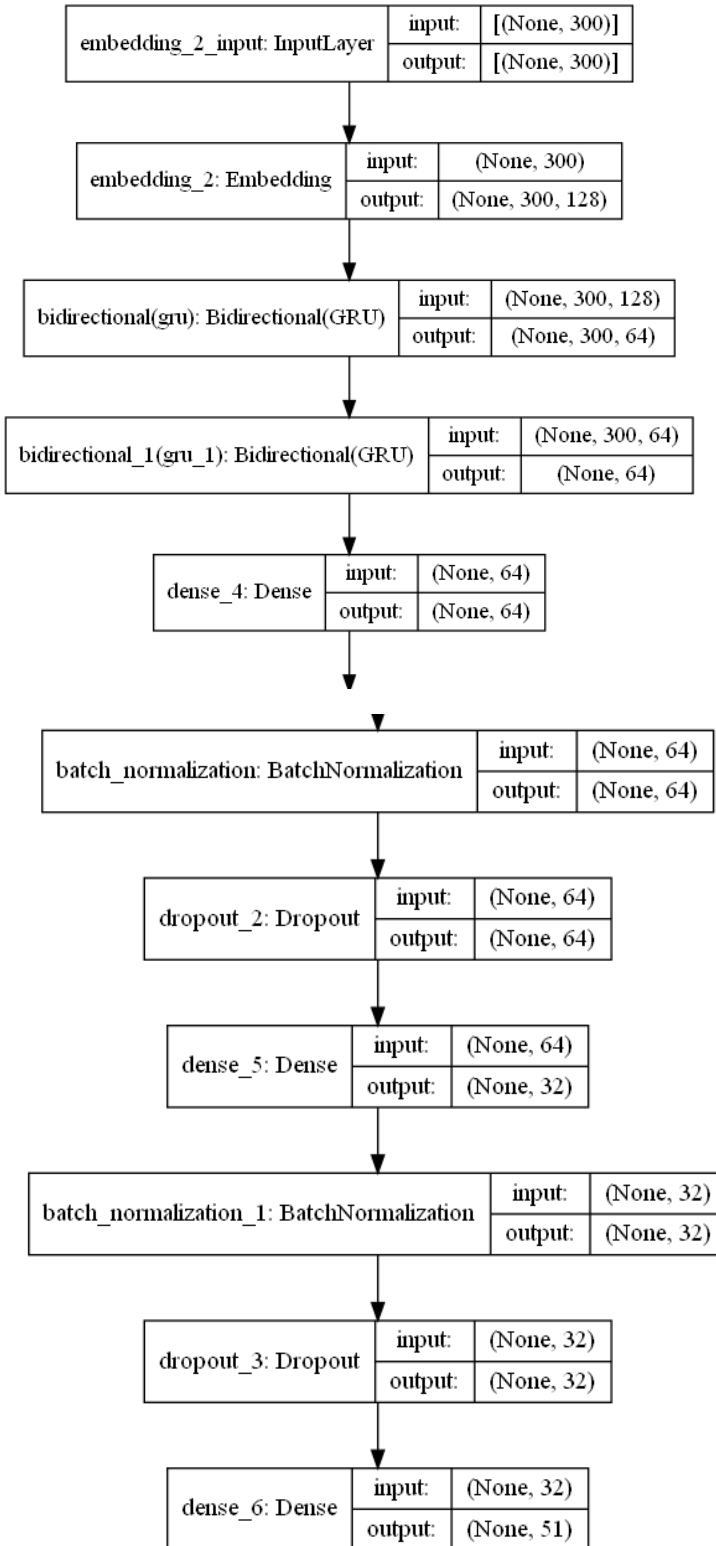


Figure 11-Bidirectional GRU model architecture

Transformers

The major drawback of the sequence-to-sequence model is that the performance decreases as the input space gets larger though LSTMs and GRUs are meant to deal with this problem. Also, the model cannot achieve parallelization. While these problems can be overcome by applying Transformer architecture. The Transformer model is state of the art for sequence and language tasks where its architecture is solely based on attention technique without using any RNN but often lead to increased accuracy. Moreover, the Transformer architecture is also parallelizable. (Amal N)

The Transformer uses self-attention to compute the query, key and value of each word from an input sequence. Then, to measure the importance of each word with respect to each other, it maps to a new sequence by taking the dot product of query and key vectors and apply SoftMax to normalize the result. Moreover, Transformer can have multiple versions of query, key and value to achieve multi-headed self-attention. This allows the model to learn separate weight matrices and combine them into the final output such that each head can learn a different aspect of the mapping directly from the data.

Model Architecture

The Transformer has six layers. The first layer is the multi-head self-attention layer that takes an input of sequence and output a new sequence to pass through a fully connected feed-forward network. Each sequence will get standardized by normalization layers twice, one is prior to the feed-forward network, and the other is prior to the output. So, both the multi-head self-attention system and feed-forward network have a residual connection and a normalization layer.

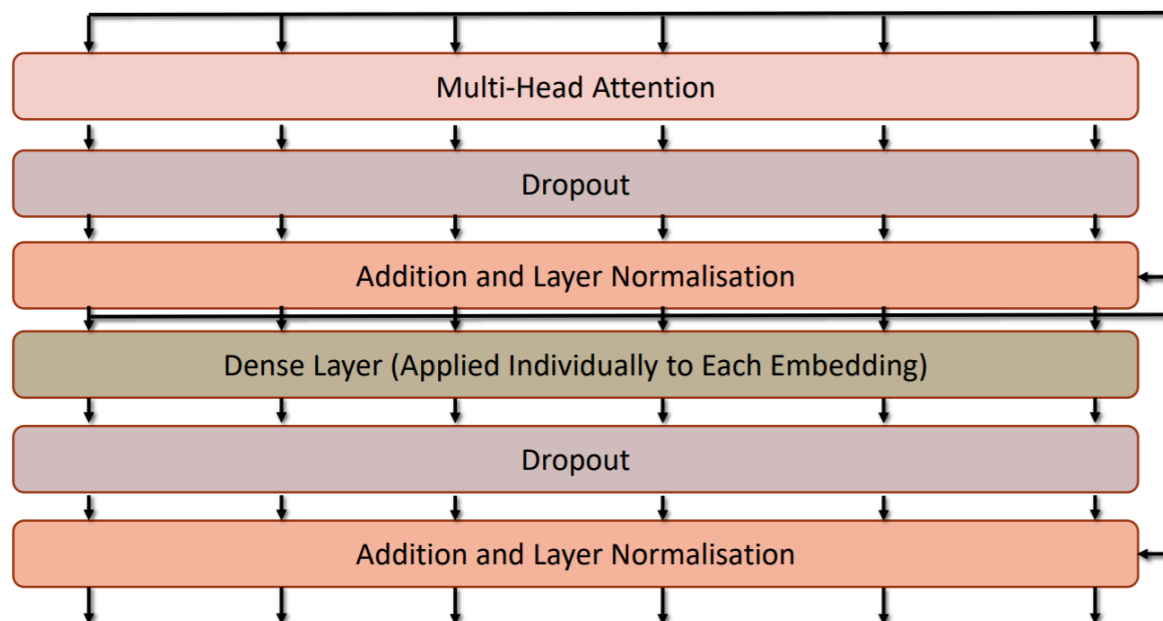


Figure 12 4- Layers within a transformer block

As the figure below shows, we used the same tokenized word embeddings from the previous RNN model and applied two stacked transformer blocks before the max pooling and the final dense layers. What needs to be noticed is that Transformer not only uses the word embeddings but also generates position embeddings to encode position information in the word embeddings. The final input is the sum of two embeddings, thus captures the word and its location.

Assignment 2

We compile the model with sparse categorical cross-entropy loss function and trained 20 epochs with batch size 64. The final test accuracy was 77% and 1.8 for test loss.

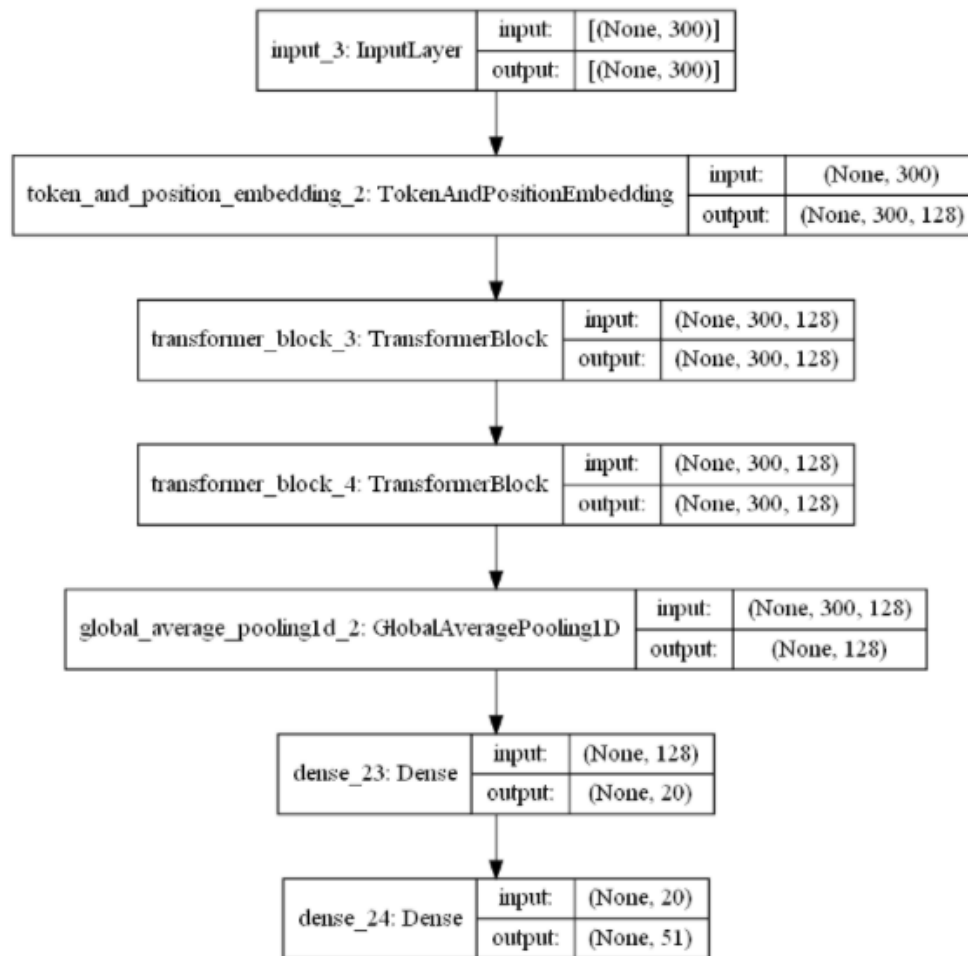


Figure 13 5- Stacked transformer model architecture

Assignment 2

Evaluation and Discussion

Followings are the loss and accuracy results of each approach we used:

```
C: [0.1, 1, 10, 30, 50, 100]
class_weight: [None]
kernel: ['linear', 'rbf']
gamma: [0.1, 0.01, 0.001]
degree: [2, 3, 4, 5, 6]
Best hyper-parameters: C = 100 class_weight = None kernel = rbf gamma = 0.001
```

Training accuracy = 0.9277777777777778

Testing accuracy = 7481372549019608

Figure 14-SVM(0.75 test accuracy)

Epoch :13
Training Loss: 0.25057774782180786
Validation Loss: 1.1220972537994385
Training Accuracy: 0.943790853023529
Validation Accuracy: 0.8136274218559265

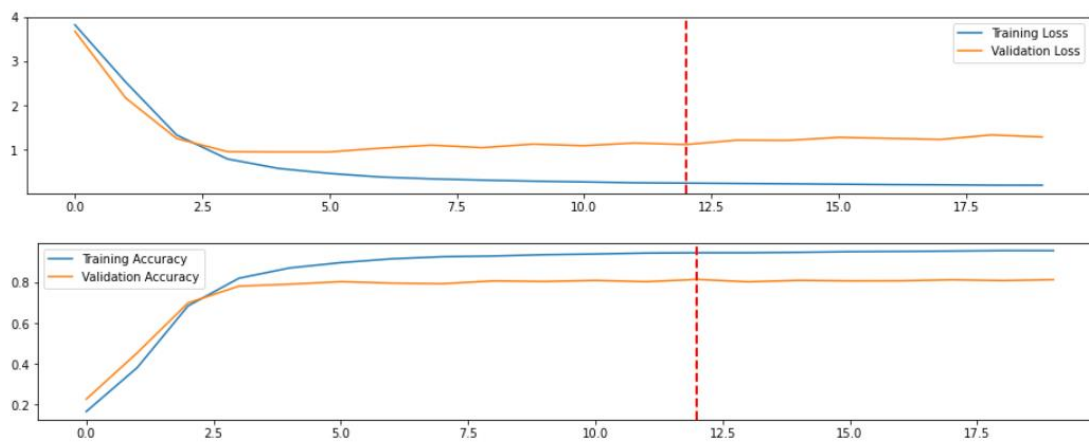


Figure 15-Bidirectional GRU(0.81 test accuracy)

Epoch :13
Training Loss: 0.19863539934158325
Validation Loss: 1.6176252365112305
Training Accuracy: 0.9434313774108887
Validation Accuracy: 0.7849019765853882

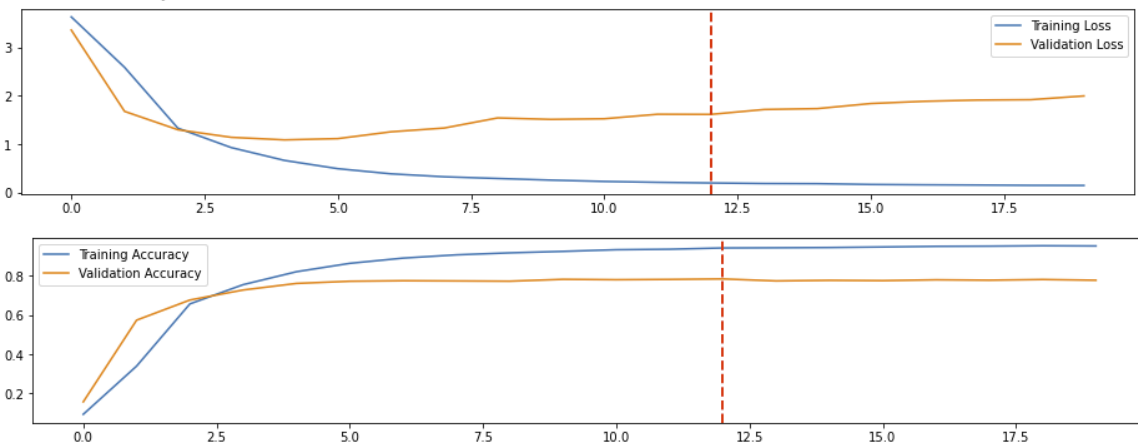


Figure 16-Stacked Transformer Model(0.78 test accuracy)

Due to the large number of classes we have (51 classes) for classification, the confusion matrix plot is too huge to display and hard to read, so we will only take the overall accuracy as the accuracy metric for evaluation.

Comparison among the 3 approaches

The highest test accuracy among the three approaches is Bidirectional GRU (0.81) followed by Stacked Transformer (0.78), Grid search SVM(0.75). The SVM model has the lowest accuracy because it only counts the words appeared most in each sender's message but does not learn how the sentences are written by everyone or the sequences in every email. Both of the other two deep learning method has taken sequences into consideration but the Bidirectional GRU outperformed Stacked Transformer by 0.03. Based on the research has been done, Transformers has advantages over RNN because they can process a sentence or a sequence as a whole using multi-headed attention thus avoid recursion and gradient vanishing problem. So, the Stacked Transformer approach is expected to have a better performance than Bidirectional GRU. However, our results show GRU is slightly better than Transformer. This might be because the max length of words we used for our input is 300 and it may not be long enough to highlight Transformers' advantages in solving sequences with long dependencies over RNNs.

Comparisons with previous existing approaches

In the Related Work approach 3, it used SVM with 'gridsearchcv' to classify all Enron email POI and reached a high accuracy as 0.85. A key difference between this approach and ours is that it has used all the word contents within each sender's folder as inputs. In other words, it tried to classify an email that comes from a sender's inbox rather than an email written by the sender. So, it is normal to get a higher accuracy in this way, because each sender will have their own frequent contact list, SVM can classify a message from an inbox simply based on the contact names appeared in the email. However, during our data processing, we extracted the messages that only written by the sender and excluded the others such as response emails and forward emails. Therefore, our task is more challenging which explains why we have a lower accuracy as 0.75.

In the Related Work approach 1, a LSTM RNN was trained with word2vec embeddings to classify contents from 10 folders in Enron dataset. The best test accuracy was 0.76 which is lower than our Bidirectional GRU approach. Our approach extended the previous approach in several aspects. In terms of the model complexity, we used a more advanced sequence to sequence Bidirectional RNN instead of a basic LSTM RNN. It allows the model to learn the context better by considering sequence in two directions. Moreover, our task is more difficult than the existing task where we are identifying 51 people's email message rather than 10 folder's texts. Even though, we still achieved a higher accuracy as 0.81.

In the Related Work approach 5, Transformer model was used to classify a range of text with 25 labels. Because the dataset it used was not the Enron email, so we cannot compare the results directly. What we have done was implementing another Transformer block on top of the existing Transformer model thus to make the architecture deeper and expected to improve the performance by any chance.

Conclusion

Overall, in general, this investigation has been conducted successfully by first research on different existing approaches, then exploring the dataset to further process the data in a more efficient way, three text classification algorithms are further developed to classify the senders based in the email contents, in particular, SVM, RNN and Transformer.

Each method has been compared and evaluated based on its advantages and disadvantages. Throughout the three approaches, it has been found that Bidirectional GRU brings the best model performance while the SVM brings the lowest accuracy amongst the three. However, there is not a huge accuracy different among the three different algorithms. Thus, it further suggested that all the approaches are valid with a reasonable accuracy.

Throughout the whole investigation, there are a range of factors such as time, computational expenses have limited the conduction of this project. For instance, in order to reduce the computational demands, only top 51 most frequent email senders were used to train the model for classification. Therefore, in future investigations, more data should be used to help the model to learn better and avoid overfitting problem to further increase the model performances. Another improvement can be made to this project is to explore more algorithms to investigation if a better performance can be found. Yet, due to the time limitation, only 3 algorithms have been investigated in this project.



Therefore, it can be concluded that the investigation of exploring different algorithms to classify the email senders has been conducted successfully although there are several limitations and further improvements can be made. These investigated algorithms are not restricted to be effective on this particular dataset, but also can be applied for other similar text classifications tasks. Thus, it further expected to help to gain a better understanding of the text classification approaches that can be applied in various real life projects.

Bibliography

1. Purtil, C. (2019). *The emails that brought down enron still shape our daily lives*. Quartz. <https://qz.com/work/1546565/the-emails-that-brought-down-enron-still-shape-our-daily-lives/>
2. Karani, D. (2020). *Introduction to word embedding and word2vec*. Medium. <https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa>
3. Keras. (2020). Keras documentation: The sequential model. https://keras.io/guides/sequential_model
4. Ankur, K. (2020). Enron email classification using lstm. <https://www.kaggle.com/ankur561999/enron-email-classification-using-lstm>
5. Analytics Vidhya. (2020). *What is tokenization in nlp? here's all you need to know* <https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/>
6. nowshad-sust. (2017). Nowshad-sust/enron-sender-detection. GitHub. <https://github.com/nowshad-sust/enron-sender-detection>
7. Javed, M. (2020). *The best machine learning algorithm for email classification*. Medium. <https://towardsdatascience.com/the-best-machine-learning-algorithm-for-email-classification-39888e7b1846>
8. Zhanna, T. (2020). *Text classification using a transformer-based model*. Medium. <https://medium.com/the-center-for-social-media-and-politics/text-classification-using-a-transformer-based-model-1bed832f22fe>
9. Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014, December 11). *Empirical evaluation of gated recurrent neural networks on sequence modeling*. arXiv.org. <https://arxiv.org/abs/1412.3555>
10. Liu, P., Qiu, X., & Huang, X. (2016, May 17). *Recurrent neural network for text classification with multi-task learning*. arXiv.org. <https://arxiv.org/abs/1605.05101>
11. Nair, A. (2020, October 27). *Transformers simplified: A hands-on intro to text classification using simple transformers*. Analytics India Magazine. <https://analyticsindiamag.com/text-classification-using-simple-transformers/>
12. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (n.d.). *Attention is all you need*. List of Proceedings. <https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>

Appendix

Statement of contribution

	Name	% of contribution	List of works	Signature
1.	Shu Du	25%	<ul style="list-style-type: none"> • Bidirectional RNN • Transformers • Evaluation and discussion 	
2.	Haonan Jiang	25%	<ul style="list-style-type: none"> • Introduction • Related works • Svm parameter search • Conclusion 	
3.	Zhiyuan Jiang	25%	<ul style="list-style-type: none"> • Data • Data Processing • Code generating 	Jerry
4.	Jiyan Zhu	25%	<ul style="list-style-type: none"> • SVM's documentation • Tokenizer's training • Presentation's PPT 	