Dissertation Type: software development

DEPARTMENT OF COMPUTER SCIENCE

The Language of Scam-Baiting

Thomas Armson

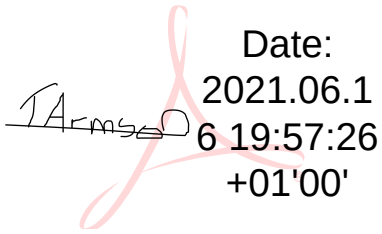
A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Bachelor of Science in the Faculty of Engineering.

Wednesday 16th June, 2021

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of BSc in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Thomas Armson, Wednesday 16th June, 2021



Contents

List of Figures

List of Tables

Executive Summary

Email scams are still a part of modern life and falling victim to a scam is at least disheartening and at most deadly. In response to this reality, scam-baiters are a community of people who have formed to combat this blight by purposefully engaging scammers to waste the scammer's time, reducing their effectiveness at scamming legitimate victims.

My project will use data sourced from the scam-baiting community to train a series of machine learning models for two purposes:

1. To distinguish between a scammer and a scam-baiter in scam-baiting conversations.
2. To distinguish between a conversation occurring between a scammer and scam-baiter and a regular non-scam email conversation

Furthermore, my project will analyze how the use of language in scam conversations changes over time. These tools and analysis will add to the tools available to the cyber security industry for identifying scam email conversations so that email filtering may be more effective and email providers better able to intervene ongoing scam solicitations.

My contributions and achievements are as follows:

- I spent 10 hours researching the various scam-baiter communities and their documented interactions.
- I wrote 1500 lines of code, performing the tasks of: data collection and cleaning; feature extraction for machine learning models; training and testing models; using said models to analyze how language changes over time in scam conversations.
- I trained models for distinguishing between scam and non-scam conversations.
- I trained models for distinguishing between scammers and non-scammers.
- I trained an LDA model for topic clustering, using said topics to plot language use over time in scam conversations.

This project did not require ethical review, as determined by my supervisor Dr. Matthew Edwards.

Supporting Technologies

I used the Python programming language and many of its supported modules and classes to achieve my goals, in particular:

- Python 3.7 was my programming language of choice, and all my code is implemented in this language.
- The Python modules SKlearn and Gensim were used for training and testing my machine learning models.
- The Natural Language Toolkit (NLTK) Python module was used for tokenization and stemming.
- The Matplotlib Python module was used for graphing my results and tests
- The Numpy and Pandas Python modules were used for data storage and handling

Notation and Acronyms

419DB	:	419Eaters Database
LDA	:	Latent Dirichlet allocation
ML	:	Machine Learning
LR	:	Logistic Regression
SVM	:	Support Vector Machines
CV	:	Cross-Validation
W2V	:	word2vec
BOW	:	Bag of Words
FCO	:	Frequency Cut-Off
EEC	:	Enron Email Corpus
NLP	:	Natural Language Processing

Acknowledgements

I would like to acknowledge and thank my Supervisor Dr Matthew Edwards for his guidance throughout this project.

I would also like to thank my Mother and Sister, whose help and support has kept me sane throughout the difficult time this project was completed.

Chapter 1

Contextual Background

1.1 Email Spam and Scams

Email spam, also know as junk mail is defined as:

> "Unsolicited, unwanted email that was sent indiscriminately, directly or indirectly, by a sender having no current relationship with the recipient." [10]

Email spam makes up more and more of email traffic as the technology required to produce the spam increases in availability and simplicity [18]. In fact, some estimates put the percentage of emails that are spam at 73% [16].

Not all spam emails are malicious (ie intend to do harm to the receiver). A non-malicious spam email may be an unwanted email from your local supermarket advertising their newest deals. Malicious spam however is a considerable proportion of received spam and various methods exist for a malicious individual to harm the emails recipient. Most of these methods are included under the umbrella of 'phishing' scams [6], where a malicious individual will attempt to gain valuable information about the emails recipient through trickery or coercion. Valuable information could be an individual's social network login information, credit card numbers, banking details etc.

The scam of interest to this project is the Nigerian Prince Scam (also referred to as the 419 scam, a reference to the criminal code for this type of scam in Nigeria, where the modern version of the scam is said to originate [20]). The scam has existed in one form or another however for a vast amount of time, with one early documented example dating to the late 18th century [3]. The basic principles of the scam are as follows:

The scam target will receive an email from an individual pretending to be a figure of prominence or vast wealth. This figure will usually claim to be from an area of enough mystery or anonymity to the target that the events laid out next might appear plausible. The figure might claim that they are in some form of legal trouble, for example, they may have a large fortune stashed away and want to move it out of the country but can't due to incompetent / corrupt officials. For the relatively low cost of a bribe or a legal

processing fee the scammer promises to provide the target with a significant share of the newly liberated wealth. Once the scammer has managed to receive payment for whatever proposed purpose, they are rarely heard from again and vanish as quickly as the appeared. Details may vary such as the location, amount required or circumstances causing the figure to need help but the basic principle is the same, "Give me some money now and I will give you more money later".

This 419 style scam has been in use for some time now and still sees considerable use and moderate success, with amount stolen equally hundreds of millions of dollars per year [9] and people going as far as to commit suicide once falling victim to the scam [4]. This displays how serious a problem this scam is to the modern individual and is the main moral motivation behind my project.

A more concrete example of a 419 scam solicitation email may be seen here **??**.

1.2 Scam-Baiting and 419Eaters

The 419 scam has become so well known that an online community has formed around the idea of engaging with the 419 scammers so as to waist the scammer's time. This, they reason, will reduce the likelihood of the same scammer successfully scamming a genuine victim, and as such, these scam-baiters are doing good work.

'419Eaters' [1] is one of the larger scam-baiting forums on the internet. The members of this community have taken to documenting their conversations with scammers as they pretend to be failing victim to the scammer's tricks. This has caused there to be created, a relatively large database of currently 658 scam-baiting conversations, that their forums curate and provide available for further use and study. This database will be the main resource used in my project to analyze language use and train models to identify scam conversations. An example of a scam-baiting conversation from the 419Eaters forum may be seen in Figure 3.1.

The 419Eaters database (419DB) isn't perfect however. I plan to return to this point in my critical evaluation however its worth noting at this stage that a conversation taken from the 419 Eaters database will not 100% match a conversation between a scammer and a genuine victim. This is in part because a scam-baiter is aware of the real situation they are communicating in whereas a genuine victim would not be, however the scam-baiters make a concerned effort to appear as genuine victims for as long as possible to better their goal of wasting scammer's time, and as such conversations from the 419Eaters database can be considered a strong facsimile.

1.3 Current Scam Defense Measures

The next question to address is: 'Why are models trained using this 419Eaters Database necessary?".

Current email spam filters can implement many different methods to protect an in-

dividual. These include challenging an email origin to respond to a query and analysing and data-mining many sections of the email including the header and other meta data [7].

These methods do however contain a shortfall. These methods work on the idea that blocking the initial solicitation email from a scammer is sufficient to protect their client. This idea, however only provides two outcomes:

1. The solicitation email is intercepted and never shown to the client.
2. The solicitation email is not caught by the email filter and the client is now left to their own devices and free to engage the scammer and potentially fall for there scam.

My project proposes a third outcome: one where the initial solicitation email slips past the email provider's spam filter, but a second alarm is tripped as the conversation between scammer and victim continues because now the email provider and their filtering software can recognize not just initial scam emails, but the scam conversations themselves. This outcome heavily motivates the work done in this project and may be seen as the driving force behind most decision making where goals are concerned. I want to show a proof of concept and the methodology with which an email provider may better protect their customers from 419 scams.

1.4 Previous Work

Previous work exists studying the 419 scam. Some work provides and insider's perspective [8], some provides a detailed analysis on the scam as it appears on an individual platform such as Cragslist [17] and some looks at how the 419 scammers organise [14].

A paper has previously been published using the 419DB to train distinguishing machine learning models [11], however its effectiveness was limited by the then relatively small size of the 419DB. Previously the database had contained 57 conversations whereas now it contains 658 messages. This I hope will increase the effectiveness of the models I am intending to train.

1.5 Intended Audience

As mentioned previously my project aims to show that models trained on the 419DB can be effective at distinguishing between 419 scam and non-scam conversations. This methodology may then be used by email providers or email spam filter creators to better train their filters and better protect their clients.

Further, my project intends to train models that may be used to distinguish between scammers and scam-baiters on a message-by-message basis within the 419DB. Similarly trained models should aid in the future labeling of scam-baiting conversations, allowing

larger datasets to be used to provide better results for future work. The 419Eaters dataset is advantageous because it helpfully labels who in the conversation is the scammer and who is the scam-baiter. If scam conversations not from the 419Eaters dataset may be easily labeled, then they too could be used in future work to increase the size of available training data beyond the 10x increase in dataset size seen between [11] and the current size of the dataset as mentioned in 1.5.

Finally my project should expand the understanding of academics seeking to understand the 419 scam in the modern day. I intend to use Latent Dirichlet Allocation (LDA) to identify topics in the language used by both the scammers and the scam-baiters as observed from 419DB. These topics will then be graphed in regards to time, to see how the language used by both parties changes over the course of the conversations. This should hopefully inform future work that wishes to understand how either the scammer or scam-baiter uses language to achieve their goals.

1.6 My Approach

In conclusion my aims for this project are as follows. I will:

- Train machine learning (ML) models for distinguishing between 419 scam and non-scam conversations.
- Train ML models for distinguishing between scammer and scam-baiter messages within the 419Eater conversations.
- Train LDA models for topic extraction from the course of the 419Eater conversations
- Analyze how use of language changes over time in the 419Eaters conversations

Chapter 2

Technical Background

2.1 Machine Learning

My project will train ML models at various points throughout its execution, and therefore some explanation on machine learning is needed. Machine learning is a field of computer science that studies a particular style of algorithms. These algorithms are able to make decisions. Supervised models do this after being trained on a problem with training data. Models used in this project will be both supervised and unsupervised and for the purposes of both classification and topic clustering.

2.1.1 Supervised Models

Supervised machine learning models work by taking training data as input. The importance of this training data is that it provides the model with sample inputs and crucially the data is 'labeled'. Labeled data is data that also notes what output the model should produce when provided the original data as input. Put more concretely, and in the context of emails and email spam filters, a piece of labeled data may be the content on an email and a data point declaring whether this email is spam or non-spam. This labeled training data allows the model to tune internal parameters that mean when the model is tested, by being given unlabeled data as input, it is able to make an accurate prediction as to the correct output.

2.1.2 Classification

Classification is a common task asked of supervised learning models. Classification asks the model to decide what group the input data belongs to, based on the information it has derived from the training data. An example may be: a model that has been trained to distinguish between two types of fish, Tuna and Salmon. As input this model is given the length, width, color and weight of many individual fish measured in the wild and when given this information in a testing environment, the model is expected to be able to return, as output, the species of the fish, be that Tuna or Salmon.

In more complex models more that 2 groups may be provided as sample outputs. For example a model may be able to determine which of 10 species of fish the input belongs to, or perhaps even 100. However in the case of this project, the supervised classification models trained will need only distinguish between scam and non-scam or scammer and scam-baiter, so models that best classify data into 2 groups are acceptable.

2.1.3 Unsupervised Models

Unsupervised ML models in contrast to 2.1.1, are models that find patterns and structure within input data without being provided labels denoting the correct outputs for the data. These models are typically used when less concrete answers and decisions are required from the model and instead these models tend to excel at finding compact and imaginative representations of the input data and the relationships they share.

2.1.4 Topic Modeling

Topic Modeling (or topic clustering) is a task that may be undertaken by ML models. Topic clustering is when a model analyses input data and identifies topics within the documents. In the case of textual data these topics may literally be topics in language however the term topic can also be more generally understood to be any sensible feature of data that allows it to be grouped. Textual topic models locate topics by finding links and similarities between the words in a corpus, grouping these words into topics. These topics may then be viewed by a human and labeled (as will be the case in this project). In the context of natural language processing a document is a collection of words, and a corpus is a collection of documents.

Latent Dirichlet Allocation (LDA) is the statistical model being using to cluster topics. It works by first imagining the process through which a document in the corpus was created, and using this process to reverse engineer and discover topics in the document.

It imagines the document is created by first sampling a topic from available topics and then from this topic a word is sampled and placed in the document. Now it imagines that the entire document was created this way and sees what topics it has created to allow this document to form. This process is done in such way as to produce soft clustering, that being that a word may appear as a member of more than one topic.

The topics found in the corpus may then be printed, along with the words that constitute them and to what degree does each word influence/define that topic (called a weight).

2.2 This Project's Chosen Models

Now it is understood what models are expected to do and how we may group and define them by tasks they perform and under what circumstances, it makes sense to explain

what concrete implementations will be used and how these work in more detail. The only model not explained below is the LDA model as is explained above in 2.1.4

2.2.1 Naïve Bayes Classifiers

Naïve Bayes Classifiers are models that perform the task of classification by utilizing Bayes' Theorem. Bayes' Theorem [15] is a mathematical formula used to calculate conditional probabilities, summarized below:

> For two events, A and B, if the probability that B occurs given A is know, as is the probability of A occurring and the probability of B occurring then the probability of A occurring given B may be calculated.

$$P(A|B) = P(A)\frac{P(B|A)}{P(B)} \tag{2.1}$$

Naïve Bayes classifiers expand upon Bayes' theorem by making certain assumptions about the data they receive as input. These assumptions are: [12]

- That all features are independent
- That all features make an equal contribution to the outcome

In most contexts these assumptions are not 100% true however they still work well enough in practice that this is overlooked, hence the name naïve. This naïve assumption of independence allows Equation 2.1 to be expanded with the use of the following:

$$P(X, Y) = P(X)P(Y) \tag{2.2}$$

And these two equations combine to create:

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)} \tag{2.3}$$

Equation 2.3 means that if a piece of data, for example, a text document, may be expressed as a set of features x_1 to x_n, then probabilities may be calculated expressing how likely a new document is to belong to the specified class based on the probability that those features individually would appear in a said class. These individual probabilities may be calculated via observing training data and as such the model may begin to classify testing data accordingly.

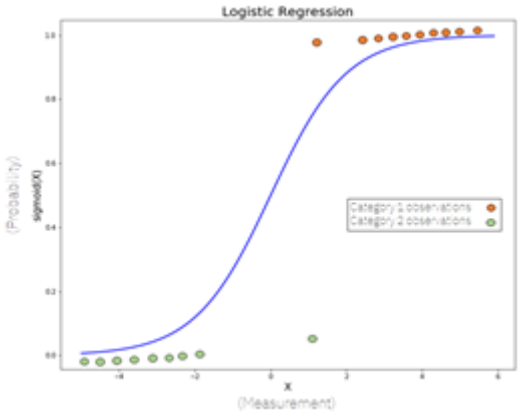


Figure 2.1: An example plotted sigmoid curve with corresponding data points

2.2.2 Logistic Regression

Logistic Regression ML models are a style of model used uniquely for binary classification. That means, unlike the Naïve Bayes model explained above a Logisitc Regression classifier, may only distinguish between 2 classes, this however is perfectly adequate for use in this project.

Logistic Regression models work by using the training data provided to the model to fit a sigmoid function to the data. The generic sigmoid function may be defined as follows:

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

Equation 2.4 provides Logistic Regression (LR) with a nice advantage in that all outputs of the sigmoid function will lie between 0 and 1, and this may immediately be read as the probability of the input data point belonging to the positive class, allowing a result of >0.5 to be classified as belonging to class 1, and <0.5 as belonging to class 0.

Most LR models will now use maximum likelihood to train the parameters of the sigmoid function to best fit the training data. Maximum likelihood works by calculating the likelihood that the observed data was produced from the current function with its current parameters. It is then able to search all available versions of the function resulting from all available set of parameters and find the function with the maximum likelihood (hence the name).

In the case of LR, this allows θ and b to be calculated for:

$$x = \theta \cdot features + b \tag{2.5}$$

This allows us to plot our data points resulting in classification as observed in Figure 2.1 [19]:

Please note the data seen in Figure 2.1 is only one dimensional, for larger dimensional data (ie data corresponding to multiple features) both θ and $features$ seen in 2.5 must

be converted to arrays of values.

2.2.3 Support Vector Machines

Support Vector Machines (SVM's) [13] are the last type of model that will be trained for distinguishing between the origin of texts in this project. Support Vector Machines work by attempting to calculate soft margins between data. These margins are what would normally be called decision boundaries. The side of a decision boundary that data can be determined to reside upon, determines the resulting classification. An optimal margin would normally be calculated by attempting to find a plane that has maximum distance between the edges of the two classes, (ie its equally far away from each class). This however creates a problem in that this margin would be incredibly sensitive to ouliers.

What makes these margins 'soft' is the fact that they allow misclassification. This means that the margin calculated by the SVM allows small proportions of the data to fall on the incorrect side of the margin. This margin is tuned via cross validation (CV) and CV will be explained further in Section 2.4.1.

Another key feature of an SVM is its use of kernels to artificially increase the dimensionality of the input data so as to find a more optimal margin or classifier. These kernels allow the SVM to simulate providing more dimensions to the data to find a more accurate classifier. Kernel choice is therefore important when training an SVM, however fortunately for this project, when preforming text classification a linear kernel is acceptable, as it best handles data with an already high dimensionality, as is the case in text classification.

2.3 Methods Of Feature Extraction

Feature Extraction describes the method by which an algorithm selects features to be provided as input for a machine learning algorithm. For example, when attempting to train a model to distinguish between negative and positive reviews on a website, the input intended to be provided to the model is the reviews (ie a collection of text). ML models require inputs to be numerical in nature because as seen throughout Section 2.2, their decision boundaries and classifiers and internal logic are determined by mathematical models.

In order to transform textual data (or any data) into model readable numerical format feature extraction must be performed. Feature extraction is therefore a very important steps as it entirely determines how a ML model will be presented data. For textual data (as is the type of data in this project) there are a variety of well documented feature extraction methods. In this project three such methods shall be used:

I	ran	on	the	beach	Sunday
1	1	2	1	1	1

Table 2.1: Table displaying BOW feature extraction on the sentence, "I ran on the beach on Sunday"

2.3.1 Bag of Words

Bag of Words (BOW) is a feature extraction method that takes textual input and for each unique word in the original text (referred to from here on as corpus) computes a count of how many times that word appears in the corpus. For example, the sentence "I ran on the beach on Sunday" may be represented by Table 2.1

Notice how beneath each word is the number referring to how many times the word appears in the corpus. Before this table is supplied as input to a ML model, the words may also be removed, simply replacing them by their index, as to the model it does not matter what the word is, only that it is a feature we wish to quantify. As long as all occurrences of the word when fed into the model are accompanied by the same index, the model will be able to perform its job of classification accurately.

Bag of words is not however a perfect system, a point motivating the next method used.

2.3.2 Frequency Cut-Off

Frequency Cut-Off (FCO) is a method of feature extraction that is similar to BOW in how it collects features, with each word in the corpus being provided its own count, however with FCO an extra step is implemented. This extra step is: each word must appear a minimum number of times or in a minimum percentage of all documents on which the feature extraction is performed.

This extra step means there are some trade-offs between FCO and BOW. FCO allows the ignoring of rarely used or perhaps misspelled words from being considered by the model. This often will improve the models performance, as it is now only trained with impactful features. This is because any word in the corpus appearing an extremely low amount of times is normally not very important or key in decision making (ie names of places or people, misspelled words). These words can also be referred to as noise, as they add unnecessary extra detail for the model to consider.

FCO however may also produce negative results compared to BOW. In the situation where a series of seldom used words where to help with decision making (ie the words 'not-scam' may only appear in a very small number of documents because what scammer would like to bring the idea of a scam to someone they are scamming, but may be a strong indicator that indeed the email where a scam). In this situation FCO removes useful information from the corpus and therefore both BOW and FCO will be implemented in this project and there results compared.

2.3.3 Word Embeddings (word2vec)

Word Embeddings are the final feature extraction method that will be explored in project. Embeddings (in the context of text classification and word2vec) convert a word into a vector representing the word's semantic meaning. These embeddings are quite interesting in how effectively they can identify the semantics of a word. For example, some word embedding schemes create results where King – man = queen. This means embedding of king, minus the embedding of man, using elementwise subtraction, equals the embedding of queen.

Word2vec [2] is the embedding scheme chosen for my project. This is because it is widely used and has many pre-trained embedding schemes already created for it and published open-source on the internet. Word2Vec works by training a model. This model is tasked with predicting a word from its context (the words surrounding it). As it repeats this task the model internally creates and tunes a series of parameters (weights) as it attempts to get better at its job of predicting. Once the model is fully trained the model itself is discarded and the weights now form the basis of the embedding.

2.4 Model Evaluation

2.4.1 Cross Validation

Cross validation (CV) is a widely used method across ML to more accurately evaluate how well a model does at performing its chosen task on unseen data. In the case of text classification CV allows more accurate scoring metrics to be calculated for the performance of the models at classifying the texts. When normally evaluating the performance of a classification model, the programmer implementing the tuning and testing of the model must determine what data should be used to train the model and what data should be used to test the model and ascertain its effectiveness.

This method is flawed because depending on the way the data is split, the model may run into some problems. For example, if the data is split so all the training data is from class X and all the testing data is class Y, the model wont have been able to effectively learn the differences between the two sets and therefore wont be able to classify effectively. Similarly, if the training set it too small and the testing set is too large, the model will have seen insufficient data to be well trained. Finally, if the training data is heavily biased towards a certain class, the model may tune itself in such a way as to heavily favor identifying unseen data as that class, and if the unseen data is of fact heavily biased towards a different class the model will perform poorly.

Cross Validation works by taking the human error out of the train and test set splitting process. It does this by splitting the data into k equally sized folds, where k is determined by the user. Each fold is then used once as the testing data whilst the other k-1 folds are used as training data. This process is repeated until each fold has had its turn at being the testing data, the individual metrics the model achieved at each fold may now be averaged or returned separately for analysis.

This method succeeds because it allows a non-biased shuffling of data, and all data may act as testing data so the resulting averages may be trusted to be a better representation of how the model may perform on truly unseen data in the future.

2.4.2 Scoring Metrics

The last topic to be discussed surrounding the technical background of this project is the scoring metrics that will be used to evaluate the classification models trained, they shall be as follows:

2.4.3 Accuracy

Accuracy is perhaps the simplest metric. Its calculated as the number of correct predictions, divided by the total number of predictions.

$$Accuracy = \frac{CorrectPredictions}{TotalPredictionsMade} \tag{2.6}$$

This metric is simple and therefore also flawed. As mention as a motivating factor for implementing CV in Section 2.4.1, if a large proportion of the testing data belongs to one class, and the model exclusively classifies data as belonging to this one class, accuracy would still return a high performance and the incorrect classifications would be of too small a size effect the score.

2.4.4 Recall

Recall is calculated as the number of correctly identified positive data points (ie, 1s that should be 1s), divided by correct positives + false negatives (ie 0s that should be 1s).

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{2.7}$$

This metric quantifies how well the model identifies positives in a dataset. This can be a useful metric to maximise therefore, however, over tuning recall may impact precision, seen in the case where if all data were classified as 1s (positive), the metric would return 100%, when this would obviously not be an ideal behaviour.

2.4.5 Precision

Precision is calculated as the number of positives classified as positives, divided by the number of positive classified as positives plus the number of negatives classified as positives (is that should be 0s).

$$Recall = \frac{TruePositive}{TruePositive + FalsePositive} \tag{2.8}$$

While recall scores how well the model performed at finding all the positive values within the dataset, precision scores how many of the positive data points the model found, were correct.

2.4.6 F1-Score

F1-Score is seen as a middle ground between precision and recall. Its calculated as:

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.9}$$

Chapter 3

Project Execution

3.1 Goals

As seen in Section 1.6 the goals of this project are as follows:

- Train machine learning (ML) models for distinguishing between 419 scam and non-scam conversations.
- Train ML models for distinguishing between scammer and scam-baiter messages within the 419Eater conversations.
- Train LDA models for topic extraction from the course of the 419Eater conversations
- Analyze how use of language changes over time in the 419Eaters conversations

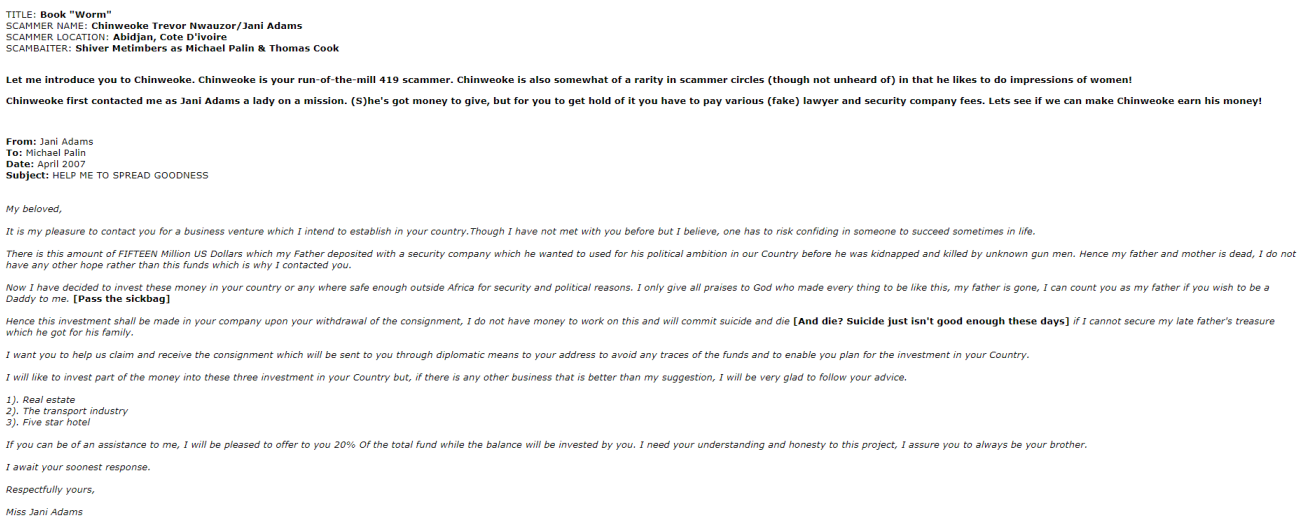
3.2 Data

I will be using two databases in order to achieve my project goals. These are the 419Eaters database (419DB) and the Enron Email Corpus (EEC).

3.2.1 The 419Eaters Database

The 419Eaters Database has previously been mentioned in Section 1 however now it must be brought up and analysed in a new context, one where it will help achieve this project's goals.

The 419Eaters database (419DB) is a collection of email conversations that have occurred between scammers and scam-baiters. These conversations exist online as a downloadable archive where each file represents one conversation. These files are of the JSON format, which allows for easy manipulation inside Python, my chosen programming language.



TITLE: **Book "Worm"**
SCAMMER NAME: **Chinweoke Trevor Nwauzor/Jani Adams**
SCAMMER LOCATION: **Abidjan, Cote D'ivoire**
SCAMBAITER: **Shiver Metimbers as Michael Palin & Thomas Cook**

Let me introduce you to Chinweoke. Chinweoke is your run-of-the-mill 419 scammer. Chinweoke is also somewhat of a rarity in scammer circles (though not unheard of) in that he likes to do impressions of women!

Chinweoke first contacted me as Jani Adams a lady on a mission. (S)he's got money to give, but for you to get hold of it you have to pay various (fake) lawyer and security company fees. Lets see if we can make Chinweoke earn his money!

From: Jani Adams
To: Michael Palin
Date: April 2007
Subject: HELP ME TO SPREAD GOODNESS

My beloved,

It is my pleasure to contact you for a business venture which I intend to establish in your country.Though I have not met with you before but I believe, one has to risk confiding in someone to succeed sometimes in life.

There is this amount of FIFTEEN Million US Dollars which my Father deposited with a security company which he wanted to used for his political ambition in our Country before he was kidnapped and killed by unknown gun men. Hence my father and mother is dead, I do not have any other hope rather than this funds which is why I contacted you.

Now I have decided to invest these money in your country or any where safe enough outside Africa for security and political reasons. I only give all praises to God who made every thing to be like this, my father is gone, I can count you as my father if you wish to be a Daddy to me. **[Pass the sickbag]**

Hence this investment shall be made in your company upon your withdrawal of the consignment, I do not have money to work on this and will commit suicide and die **[And die? Suicide just isn't good enough these days]** *if I cannot secure my late father's treasure which he got for his family.*

I want you to help us claim and receive the consignment which will be sent to you through diplomatic means to your address to avoid any traces of the funds and to enable you plan for the investment in your Country.

I will like to invest part of the money into these three investment in your Country but, if there is any other business that is better than my suggestion, I will be very glad to follow your advice.

1). Real estate
2). The transport industry
3). Five star hotel

If you can be of an assistance to me, I will be pleased to offer to you 20% Of the total fund while the balance will be invested by you. I need your understanding and honesty to this project, I assure you to always be your brother.

I await your soonest response.

Respectfully yours,

Miss Jani Adams

Figure 3.1: An example of an email existing in the 419DB

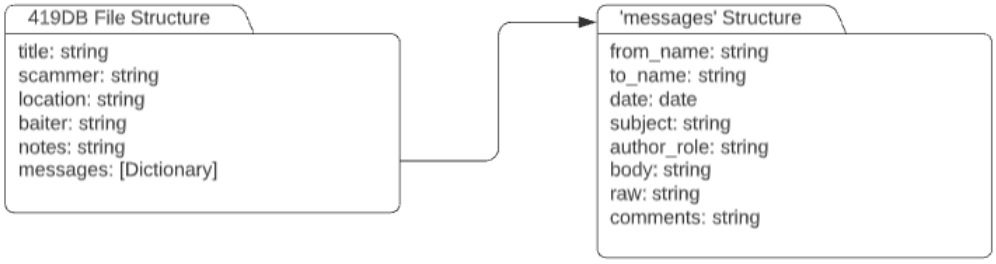


Figure 3.2: The internal structure of JSON files in the 419DB

An example of a conversation may be seen in Image 3.1 and an example of the structure of the JSON file may be seen in 3.2. Please notice the many fields that exist within the file, such as 'title', 'location', 'scammer' etc. Many of these fields will be removed in the data pre-processing steps as they do not aid in achieving this project's objectives.

The 419DB will be used to train models that require access to scam conversations, to both distinguish between them and normal conversations, and for the purpose of distinguishing between scammer and scam-baiter.

3.2.2 The Enron Email Corpus

The Enron Corpus is a collection of over 600,000 emails generated by over 100 employees of the Enron Coorporation and was generated in its current format by the Federal Energy Regulatory Commission (FERC) when they conducted a corruption investigation into the Enron Corporation shortly before its termination in 2001. The Enron Corpus was later released to the public and is today regarded as one of the best sources of real emails [5] as it is not bound by any legal or privacy restrictions and is therefore freely used in research.

```
<Date: Mon, 1 Oct 2001 15:04:03 -0700 (PDT)>
<From: monika.causholli@enron.com>
<To: cecil.stapley@enron.com>
<Subject: Pulp facts>

Cecil,
can you please take a look at this presentation
and see if numbers make sense?
thanks,
Monika
```

Figure 3.3: An example email from the Enron Email Corpus

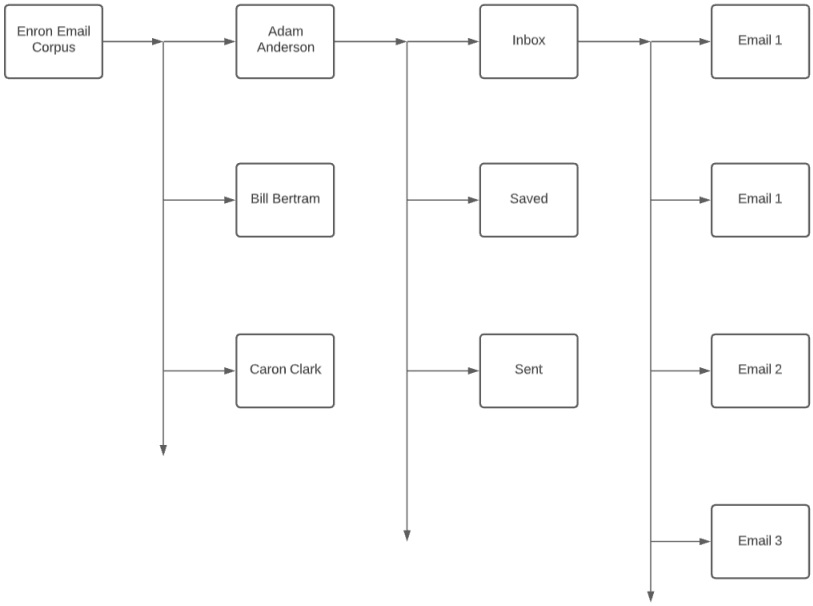


Figure 3.4: The Directory/File Structure of the EEC

An example email from the Enron Email Corpus (EEC) may be seen in Figure 3.3 and to better understand some of the re-structuring performed on the EEC during this project; the file structure of the EEC downloaded for use in this project may be seen in Figure 3.4

3.3 Design

The overview of the design of this project may be seen in Figure 3.5. This diagram breaks down the steps required to achieve this project aims into a step by step series of processes and stages, that will be discussed individually and in more detail as this section continues.

Figure 3.5: A Flowchart detailing the design of this project

Figure 3.6: The proposed file structure of the 419DB post 'Trim 419DB' step

3.3.1 Data Pre-Processing

The data pre-processing step may be seen as how 2 individual databases are made ready for use by the ML models used in this project. I can also be viewed in steps being applied to each database:

- Manipulating the database, removing unnecessary information.
- Processing and tokenizing the text of the database.

Manipulating The 419DB

As seen in Figure 3.2, the 419DB contains some information not relevant to this project. There are many fields that make up parts of the JSON object that is these files that are unnecessary for the projects intended tasks. The fields required for this project are 'messages' $->$ 'body' and 'author-role'.

To remove the remaining unwanted fields I intend to load the files from the 419DB before stripping away the unnecessary fields from the JSON files and re-saving the files in a new directory for prosperity. This step is also referred to in parts of this project as 'Trimming' and can be seen written as such in Figure 3.5. After this process is complete the file structure of the 419DB should resemble that seen in Figure 3.6.

Manipulating the Enron Corpus

For ease of use I intend for the EEC to be formatted into a similar shape and style as the 419DB, this will allow the functions that handle the text processing steps of my design to be implemented more easily, as they need only handle one style of input. This will involve scanning through the EEC (currently storing each email from an individual's inbox as a single file, and each inbox as its own directory) and collecting together all emails that share a similar 'subject'. This step is referred to in Figure 3.5 as 'Group Enron via Subject'.

The next step is to convert 'Grouped Enron' into 'Enron Conversations'. This will involve a complete scan of 'Grouped Enron'. The purpose of this scan will be to sort all

Figure 3.7: The proposed file structure of 'Enron Conversations'

emails 'subject' line and place these emails and their contents in date order in a single JSON file, creating 'Enron Conversations'. These files will now resemble the 419DB files as both will now be a sequence of messages in time sequential order that in total form a conversation. The new files should have a similar structure as seen in Figure 3.7.

Tokenizing Both Datasets

With both the 419DB and EEC now in similar, simple to understand formats, the next step will be to pross their textual content so they will be ready for feature extraction. Tokenization is the system by which a text document is converted into a list of words. As this conversion is taking place I intend to also perform standard pre-processing steps to make the data ready for feature extraction. These steps include:

1. Removing excess white-space and non-alphanumerical symbols from the texts
2. Removing all references to Enron from the Enron Corpus
3. Making all text lowercase
4. Replacing sequences of the letter X (used to denote omitted information) with a unique token indicating omission
5. Replacing all numbers with a token indicating a number was present
6. Performing tokenization to convert from string to list of tokens
7. Removing stop words from the tokens
8. Apply Porter's Stemmer algorithm to the tokens

In the above sequence of steps some points may require further explanation. Numbers and sequences of Xs are provided with new tokens as in the context of Natural Language Processing (NLP) they have only one semantic meaning. Xs mean omitted information, regardless of length, and numbers mean a number was present, regardless of what the number actually was. These steps allow a more concise set of tokens to be produced at this stage.

Note that in many NLP contexts, when performing tokenization and the data pre-processing steps that surround it, many individual will correct misspelling of words. This works as above to create a more concise set of tokens and allows the semantic meaning of the documents to be clearer, however, in the case of this project misspellings have been allowed and kept within the tokens as it is believed that such misspelling may help better identify the correct class to which the data should belong. For example, due to human error, a scammer may be more likely to misspell certain words rarely misspelled in legitimate official documents. This misspelling could therefore better identify a scammer and scam conversations, so misspelling have been allowed to remain.

The final stage of pre-processing noted in the above steps makes use of the Porter's Stemmer algorithm [21]. Stemming algorithms are algorithms that transform to better capture their semantic meaning. For example the words run and running are semantically similar yet in a pure string representation vary by some amount. Porter's Stemmer algorithm removes morphological and inflectional endings from words in the English language and is used commonly in NLP, and will be used here for the same reasons.

3.3.2 Forming The Corpus'

The next step in creating the models required to achieve this project's goals is to use the data resulting from Section 3.3.1 to train ML models. However before this step may be completed the data must be 'labeled' or 'tagged'. This involves collecting the data from the pre-processig steps and supplying data to note its correct classification. Two different tasks are to be completed by classification ML models trained in this project, they are:

1. To distinguish between scammers and scam-baiters within the 419DB data
2. To distinguish between scam and non-scam conversations

Each task will require different data and a different labeling scheme. For the first task, messages will be taken from the 419DB and provided the label of 0 if the $author_role$ of the message is scam-baiter, 1 if the $author_role$ is scammer. In this resulting corpus each document shall represent a single message. The ratio of scammer to scam-baiter messages in the 419DB is very close to 1:1, and the total number of messages forming the new corpus equals 37488.

For the second task both the 419DB data and EEC data shall be required. For this new corpus entire conversations shall be extracted from both databases. A conversation shall be provided the label of 0 if it is from a non-scam conversation (ie the EEC) and 1 if it is a conversation involving a scammer (ie from the 419DB). The ration of scam to non-scam conversations in this new corpus shall be roughly 10:175 to mirror the ratio used in [11]. To form this corpus, the entirety of the 419DB shall be used (658 conversations) and therefore 11540 conversations shall be sampled from the EEC. This means the newly created corpus shall contain 12,198 conversations.

These two newly created labeled corpus' shall from this point on be referred to as the 'Scam Baiter Vs Scammer Corpus' and the 'Scam Vs Non-Scam Corpus'.

0	1	2	...	n_words
4	7	2	...	
3	2	6	...	
⋮	⋮	⋮		
n_docs				

Table 3.1: Sample Array of size (n_docs x n_words)

3.3.3 Feature Extraction

At this stage both corpus's are ready for feature extraction using the three methods mentioned in Section 2.3. These being Bag of Words (BOW), Frequency Cut-Off (FCO) and word2vec (W2V). The sample outputs of the feature extraction methods methods may be seen in Figure 3.1.

Please note for BOW, n_words shall equal the number of unique words found. For FCO, n_words should equal a lower number than for BOW, this number is determined by how many words meet the cut-off frequency. Finally, for w2V, n_words is equal to the dimensionality of the w2v model used. Common number are 50, 200 and 300. In all cases, n_docs is the number of documents being used to train the models, either 12,198 or 37,488 as per 3.3.2.

3.3.4 Training the Models

At this stage the models may be provided with their data and begin training. All together this will produce 18 models trained for the purpose of text classification: 9 for distinguishing between scam and non scam conversations (3 feature extraction methods * 3 models), and 9 for distinguishing between scammer and scam-baiters within scam conversations (again 3 FE methods * 3 models).

Parameters influencing the success of the models will be tuned as the models are iteratively tested. This is documented in 3.4.1 and 3.4.2.

Results should then be produced from each model after performing 10-fold cross-validation. This cross-validation will help create a more balanced and bias free set of evaluation parameters. The Python module scikit-learn (SKLearn) has many robust and well tested model implementations and these shall be the basis of the models' implementation.

Topic Clustering and Language Analysis

So far the only models mentioned in much of this projects design documentation has been regarding the models that shall be trained for the purpose of text classification. This project does however intend to use one more type of ML model to achieve its aims. LDA models will be used to perform topic extraction on the scam conversations found in the 419DB. This model does not require labeled data to perform its task as its goal is to find

topics within an unlabeled collection of data. This unsupervised model shall therefore take as input the 419DB from directly after the data pre-processing stage seen in Section 3.3.1 and directly before the data is transformed into the corpus' seen in Section 3.3.2.

The 419DB shall be split into three variants, each variant being supplied to an instance of an LDA model:

- Variant A: The the entirety of the 419DB
- Varient B: Only the messages from scam-baiters
- Varient C: Only the messages from scammers

These three different document corpus' will each train different instances of an LDA model until each model provides a human understandable set of topics extracted from its corpus. These topics for example may be:

- Intimidating Language
- Formal Language
- Language implying urgency
- Language about finance

Each corpus variant will then be binned into deciles (10 sections), where each section contains the corresponding messages from every conversation in the variant. For example, if a conversation is 20 messages long, the first two messages will be placed in decile 0, the second two in delice 1 etc.

With each variant split and organized into deciles, the deciles may now used to query their respective LDA models. There respective LDA model will then provide as output a vector describing how much the topics that the model had identified appear in the documents provided from the deciles

This will allow a visual representation of how the use of the various topics the LDA models are able to extract changes of the course of scam conversations, allowing the language analysis required by this projects aims and objectives.

3.4 Implementation

To enable easier replication of this project's work, summaries of each step on the project design steps and how they were achieved in practice shall now be detailed. Note once again that the chosen language with which to implement was Python version 2.7. Any areas that deviated from the originally suggested design will be noted and the reasoning behind the deviation explained.

3.4.1 Data Pre-Processing

Manipulating the 419DB

As mentioned in the design section, the 419DB was stripped of unnecessary fields using a ¡dict¿ class to store an internal representation of the JSON files during runtime.

Manipulating the Enron Corpus

As per the design documentation the Enron corpus was first re-structured so that individual emails existed in folders corresponding to the subject field of the email. A scan was then performed over this newly formed copy of the corpus and email threads were subsequently formed into a single JSON file using a custom class for this conversion.

These two individual scans proved to be very necessary due to the large nature of the EEC. The grouping by subject was done by placing each email in a folder corresponding to the initials of the subject. For example, emails with the subjects "I am Groot" and "I am Green" would appear in the same folder upon which the second scan would be performed. This was done as folder names have a limited length and naming rules so a direct copy of the subject couldn't be used.

Special care was also taken to ensure suffixes such as Re, Fwd and Fw were removed from the subject when they are moved and or compared. Originally only Re was accounted for, causing a fix to be implemented at a later date.

Tokenizing Both Datasets

Both datasets were then tokenized, ensuring all tasks mentioned in 3.3.1 where performed. Originally for the 419DB an older implementation of Porter' Stemmer algorithm was implemented, this added lots of excess file handling steps to the process. When the process was repeated for the EEC, a Python module was found to exist, allowing the algorithm to be applied in the same instance as the rest of the tokenization and pre-processing steps.

Forming the Corpus'

Both corpus' were formed, being held during runtime in the 'Bunch' data structure provided by the sklearn module. Here however a change had to be amde from the original design. The Enron corpus was found to be too large to store effectively in memory prior to feature extraction and as such the corpus upon which feature extraction would be performed was formed incrementally, each increment then being saved to permanent memory and removed from allocated runtime memory. This was necessary as the corpus to which the Enron Corpus was added contained roughly 12,000 conversations of on average 10 messages each and at times this proved to be too much for a 32-bit run time environment with running with 8GB of RAM to handle, causing crashes and errors in processing.

Performing Feature Extraction

Performing feature extraction on the 419DB was simple however due to the edited implementation for storing the Enron Corpus a new function was designed that incrementally read from disk sections of the Enron Corpus. Feature Extraction was then immediately performed on the whole Enron Corpus and the corpus was immediately removed from run time memory once feature extraction was complete.

This implementation does still need the entirety of the Enron Corpus to be loaded into memory, however it was now done so in a more ergonomic manner due to the later stage of processing and for a much shorter period of time. This was sufficient to allow feature extraction to occur and for the task to progress. Once feature extraction had been performed the pyhsical memory footprint of the data was massively reduced as representation of the data changed from arrays of string and integers, to concise arrays of just integers. These arrays were more concise due to the counting/embedding methods used reducing the dimensionality of the data.

Training the Models

The models trained mostly without incident. There are however two things to note. The first is that the SVM models, even when using only a linear kernel to aid efficiency, took a long time to tune parameters and fit to the training data, the second being that the W2V embedding took a long time to embed the entirety of the two corpus' it was provided and as a result models requiring the W2V Embedding on the larger corpus (Scam Vs Non-Scam) where unable to be collected as the feature extraction still hadn't been successful after 2 days of processing.

3.4.2 Evaluating the Models

Implementing 10 fold cross-validation did cause some of the models to perform irregularly. For example, the original implementation of logistic regression took to crashing whilst fitting whilst using cross-validation, however these issues were soon dealt with and results produced by using a variety of cross-validation implementations, avoiding the situations causing errors. The results will be discussed in a later Section 4.2

3.4.3 Topic Clustering and Language Analysis

Creating topic clustering's using the LDA model proved an arduous task as often the topic clustering provided as output provided little insight to the data, or the topics found were near meaningless and a lot of the topics suggested proved difficult to name. This will be discussed in more detail in Section 4.3.1.

Restructuring the 419DB into deciles required a significant amount of coding to create a robust solution but the basic principles were simple in nature and therefore don't bare mentioning in detail. The graphs produced by the graphing of these topic densities over

time however did produce some interesting results, explained further below in Section 4.3.1 also.

Chapter 4

Critical Evaluation

4.1 Code Testing

The vast majority of testing conducted during the course of the completion of this project was for the purpose of confirming data integrity. This is because the majority of functions created transform large amounts of data, be that by reorganizing data into new structures, altering data with word embeddings or appending 'labeling data and so on.

In order to confirm data integrity at various stages of this project multiple testing strategies were used. One such strategy was to confirm the shape and size of the handled data at all points throughout the data's use in the project. For example, after each transformation of data is complete, the shape of the returned data is checked to ensure it has been transformed as intended.

4.1.1 Data Shape Tests

Shape Testing was performed at multiple points during the transformation of data, the outline and summarised results of these tests may be seen in Table 4.1

Note-worthy Tests

4 & 5: These functions did not have an intend output shape due to the fact that the output was subjective on the input and how they were grouped, more shared subjects would equal fewer resulting conversations, but the conversations themselves would be longer.

6: Enron Tokenize also performed the task of sampling from the Enron Corpus for the required number of email conversations, hence the drop in returned email conversations from the whole corpus to this new number.

7: Scam Vs Non-Scam Corpus combined the email threads from Enron with the files from 419DB, hence the shown mathematics.

Test ID	Transformation	Shape before	Intended shape after	Observed shape
1	Trim 419DB	658 Files	658 Files	658 Files
2	Tokenize 419DB	658 Files	658 Files	658 Files
3	S Vs SB Corpus	37448 Docs	37448 Docs	37448 Docs
4	Enron Group	150 ppl, 517401 Files	517401 Files	517401 Files
5	Enron Convert	517401 Files	NA	159282 Threads
6	Enron Tokenize	159282 Threads	11540 Threads	11540 Threads
7	S Vs NS Corpus	11540 + 658 Convs	12198 Docs	12198 Docs
8	BOW FE (S vs SB)	37448 Messages	(37448x1000) Mat	(37448x1000) Mat
9	BOW FE (S vs NS)	12198 Documents	(12198x1000) Mat	(12198x1000) Mat
10	FCO FE (S vs SB)	37448 Messages	(37488xN) Mat	(37488x148) Mat
11	FCO FE (S vs NS)	12198 Documents	(12198xM) Mat	(12198x392) Mat
12	W2V FE (S vs SB)	37448 Messages	(37448x200) Mat	(37448x200) Mat
13	W2V FE (S vs NS)	12198 Documents	(12198x200) Mat	(12198x200) Mat

Table 4.1: Table containing outline of shape tests: Plan and Results

Model	Feature Extract	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Naïve Bayes	BOW	69.2	73.6	69.4	67.8
Naïve Bayes	FCO	63.7	67.0	64.0	62.1
Naïve Bayes	W2V	69.5	70.1	69.4	69.2
Logistic Regr	BOW	85.1	85.3	85.1	85.1
Logistic Regr	FCO	77.7	78.4	77.8	77.6
Logistic Regr	W2V	79.8	80.0	80.0	80.0
SVM	BOW	81.3	83.7	78.0	83.7
SVM	FCO	77.3	81.4	71.4	76.1
SVM	W2V	78.5	78.1	80.0	80.0

Table 4.2: Results obtained from the Scammer Vs Scam-Baiter Corpus

8 & 9: For spatial efficiency the BOW FE method limited the number of stored counts to the most numerous 1000, hence the shape.

10 & 11: FCO further reduced the size of the returned array by removed all words from the corpus that don't meet the cut of frequency of appearing in 20% of the documents.

12 & 13: The chosen W2V Embedding loaded and used in this project had a complexity of dimensionality of 200, meaning it used 200 data points per word to store its embedding.

4.2 Classification Model Results

The results obtained from the models trained in this project to distinguish between scammers and non-scammers inside the 419DB can be seen in Table 4.2

The results obtained for the similar models trained instead distinguish between scam and non-scam email conversations may be seen in Table 4.3

Model	Feature Extract	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Naïve Bayes	BOW	82.5	62.6	90.7	65.7
Naïve Bayes	FCO	86.0	64.6	92.4	68.2
Naïve Bayes	W2V	NA	NA	NA	NA
Logistic Regr	BOW	99.8	98.6	99.3	98.1
Logistic Regr	FCO	99.9	97.1	99.1	98.1
Logistic Regr	W2V	NA	NA	NA	NA
SVM	BOW	99.8	99.9	99.9	99.9
SVM	FCO	99.7	99.9	99.9	99.9
SVM	W2V	NA	NA	NA	NA

Table 4.3: Results obtained from the Scam Vs Non-Scam Corpus

4.2.1 Commentary and Evaluation

Scammer Vs Scam-Baiter Classification

Provided in Table 4.2 are the results for the models trained to distinguish between the scammer and the scam-baiter inside the 419DB. For these models the best performing model across all scoring metrics is the Logistic Regression model that used BOW for feature extraction. In terms of expected results, 85.1% is a relatively low achievement, especially with the volume of data involved as can be seen when the results are compared to the results obtained in [11]. This could however indicate that the language used by the scam-baiter has become more similar to the scam-baiter as the 419DB has expanded. This could be the direct result of an effort to better engage and therefore better inconvenience the scammer, or it could instead be a result of more informal language becoming more commonplace in the modern online space.

This set of low results may also be caused by parts of the projects methodology. Removing the maximum number of features from the generic BOW method and also lowering the minimum frequency required for a word to be allowed by FCO could increase the results seen. These changes should provide more information to the models and if improvements can be made by increasing the amount of data seen, this should trigger this improvement.

A reason perhaps why the models using W2V extraction scored poorly could be that the dimensionality of the pre-trained embedding used was only 200. This pre-trained embedding was trained on tweets from the social media platform twitter. This could also explain the W2V using models poor performance as its not unlikely that the language used in the 2 environments (twitter vs scam conversations) may be substantially different, causing the embedding the W2V model used to be poor embeddings for the 419DB.

Scam Vs Non-Scam Classification

As mentioned in the Section 3.4.1 of the project implementation, creating the dataset corresponding to W2V feature extraction performed on the scam Vs Non-Scam Corpus proved too time consuming for results to be collected for this project. Upon further in-

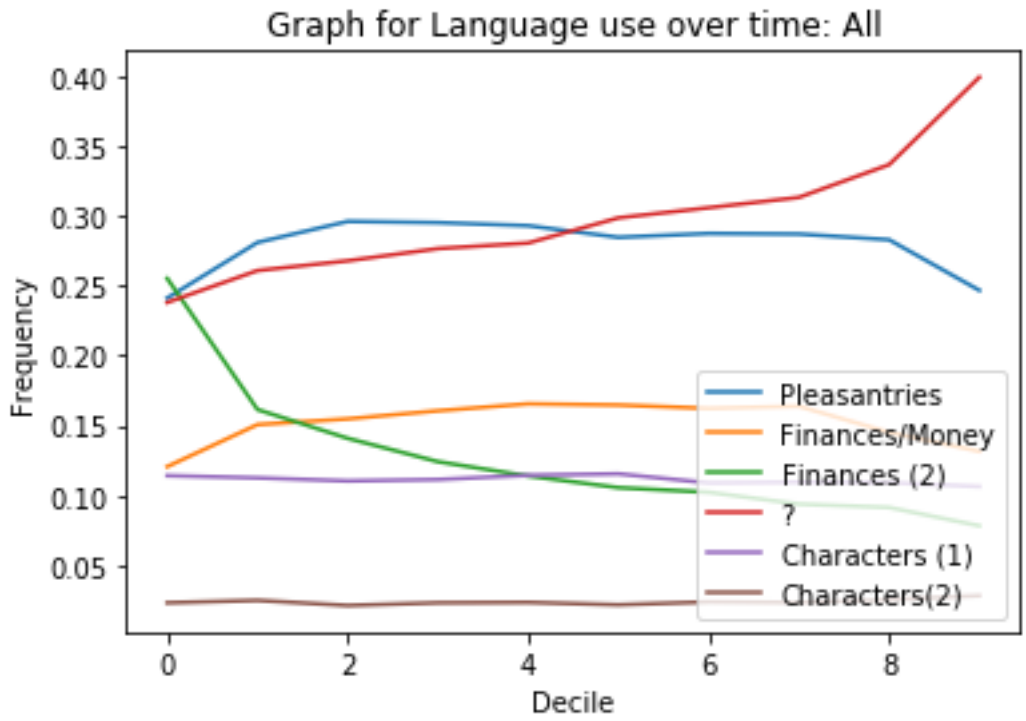


Figure 4.1: Graph of topic use from both authors over time

spection the algorithm was computing the required embeddings, so with a large amount of time or perhaps some more powerful hardware, these models could be trained as intended.

The best results seen here were very strong at 99.9%, mostly achieved by the SVM models. This could suggest that where the SVM excels (at creating soft margins) is exactly what is needed for effective classification in this context. This could mean that outliers are a common and otherwise damaging part of the combined Scam Vs Non-Scam email corpuses.

This task and the classifiers it has trained can be seen as significantly more successful than the classifiers trained in for the previous task. This could be in part to the larger amount of data available when including the Enron Corpus, or perhaps the task is significantly easier for ML models to perform, with a greater distinguishing boundary between the two classes than in the previous problem.

4.3 Language Analysis

4.3.1 Results

This results section shall provide analysis of the language used by scammer and scam-baiters by analysing the results return from three LDA models. One model was trained from each author, with the final model being trained from messages from the two authors combined.

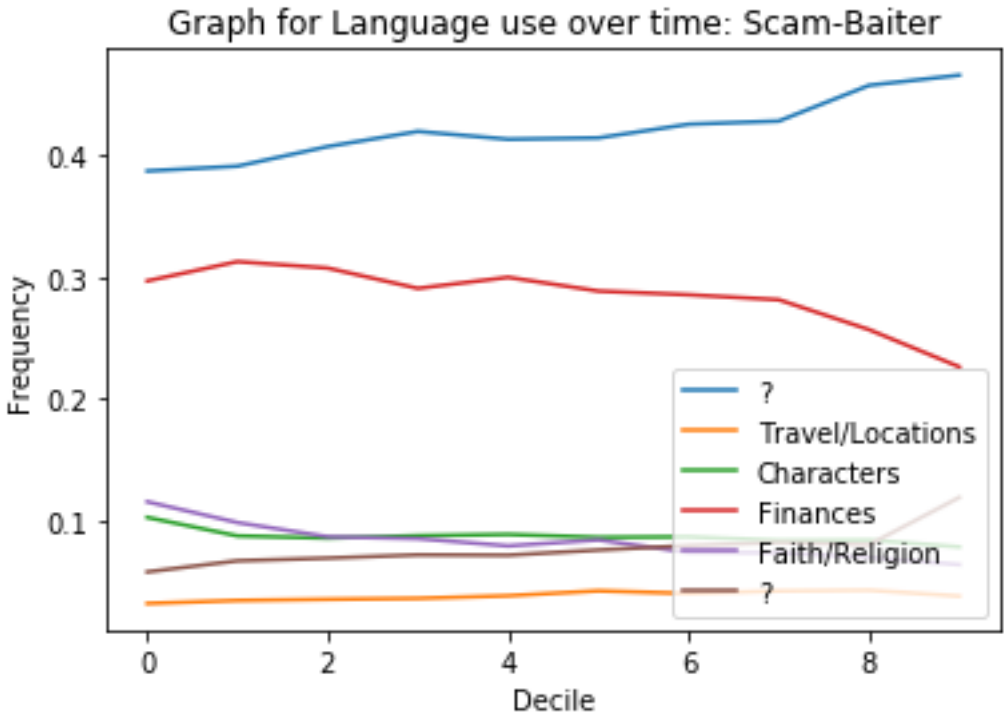


Figure 4.2: Graph of topic use from Scam-Baiters over time

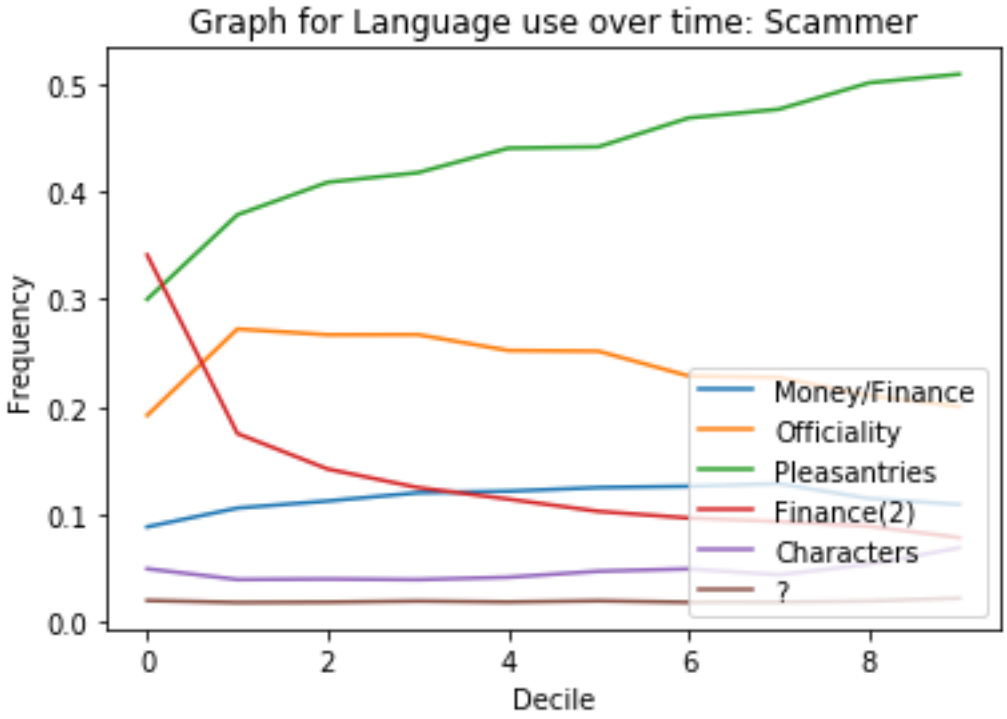


Figure 4.3: Graph of topic use from Scammers over time

Language and Topic Use from both author types

In Figure 4.1 it can that perhaps contrary to popular belief, the use of pleasantries throughout a scam conversation remains mostly constant, with only a small down tick near the end of a conversation. This is perhaps explained by the fact that both the scammer and scam-baiter are aiming to keep the other party engaged in the conversation for a long as possible until it is clear that the scam will not succeed, when we then see the use of pleasantries fall.

Note however that one may expect a sharper drop off where this case. We can also determine from the above graph that certain words relating to finance as use extensively in the beginning of conversation but then decrease in occurrence/use. This is perhaps because once a payment method has been discussed it need no longer be a concern. Also note that finance(2) here included more slightly more formal language such as 'company' and 'document' so again, perhaps once credibility had been stressed at the beginning of a conversation, the conversation moved on to other topics.

Unfortunately, the topic marked '?' in this instance was not a human interpretable topic and included words such as 'get', 'know', 'time' and 'like'. If I were pushed to make a judgement I may call this topic "short filler words" and unfortunately this doesn't inform the viewer on any particular behavior.

Most other topics remain fairly constant in frequency of occurrence and therefore don't inform the viewer of much either.

Language and Topic Use from Scam-Baiters

Figure 4.2 displays use of language by the scam-baiter over time and does provide some useful insight. For example, one topic identified by the topic clustering process was "Faith/Religion". Use of this topic appeared relatively consistently by the scam-baiter throughout their conversations.

From context and study of the original conversations its understand that this is because scam-baiters are seen to proport their innocence and righteousness to the scammer throughout their conversations, both to make the scammer feel guilty, and to assert their naivety when they have to ask the scammer for more time to complete their requested tasks, in an effort to make the delays they are reporting believable.

Similarly to the previous figure, it can be seen that use of financially connected terms drops as the conversation progresses. I posit that if is for the same reason as discussed previously, that conversation related to finances is established at the beginning of the conversation and then allowed to drop in frequency as the conversation progresses.

Language and Topic Use from Scammers

In Figure 4.3 the most notable trends seen in topic frequency are seen in Pleasantries and Official language. Perhaps counter intuitively the use of pleasantries increases as a

proportion of language used by the scammer as the conversation progresses. The reasons as to why this may be the case may be viewwed as a combination of factors. One is that as the scam-baiters ruse progresses, the scammer, believing this scam is taking too long, tries to increase the pressure applied to the scam-baiter by use more pleasant language to increase social pressure. Another reason may be that, it is not uncommon that if a scammer gets called out by the scam-baiter and it is revealed the scammer is being toyed with, they will occasionally resort to pleading with the scam-baiter, claiming to be a poor or desperate individual.

The other interesting change is the change in frequency of official language. This, I believe, may be easily explained. In the beginning of a conversation, it is seen that the scammer makes a great effort to appear formal and official, to win their targets trust, however as the conversation progresses and the scammer must produce more organic and original responses their trained aura of officiality drops and more natural language takes its place.

Similarly to the previous two graphs we also observe in this data that the use of financially connected terms steadily decreases as the conversation progresses.

4.3.2 Commentary and Evaluation

The topic clustering task performed by the LDA models trained in this project were of mixed success. The LDA models were able to effectively cluster and extract topics from within the provided texts, however the plotting of these topics over time did not reveal many unique or new insights into the language used by the individuals involved. The lack of interesting language changed over time is not a fault of the LDA model, as it is not designed to distinguish topics based on how these topics change over time.

This means the fault perhaps lies with the choice of model used by this project and assuming that any topics an LDA model was able to distinguish from a document would in-fact have varying densities over time. Upon reflection it seems somewhat obvious that a topic that remains consistently prominent through-out the series of messages that make up a conversation would be more likely to be identified as a topic by LDA. LDA will better identify a topic, if it can see this topic used all through the conversation. This means that perhaps leaving this task to a fully unsupervised ML model may always encourage mixed results. Then again, perhaps with a different model used for topic clustering, the need for consistency may be eliminated and therefore topics with more interesting variations found.

There also remains the question of the un-tagged topics that the LDA model identified, but which I was unable to successfully provide a topic description. A greater understanding as to the intricacies of language may have allowed a more accurate description.

Chapter 5

Conclusion

5.1 Summarising of Project Goals

In conclusion, this project aimed to do 4 things:

- Train machine learning (ML) models for distinguishing between 419 scam and non-scam conversations.
- Train ML models for distinguishing between scammer and scam-baiter messages within the 419Eater conversations.
- Train LDA models for topic extraction from the course of the 419Eater conversations
- Analyze how use of language changes over time in the 419Eaters conversations

5.2 Evalution of Project Status

All of these proposed aims have been achieved albeit to varying levels of success. The models used to distinguish between Scam and Non-Scam email conversations where able to score as highly as 99.9% accuracy. This definitely confirms the motivating use case, that these models be used to better protect email clients from scammers once the initial solicitation email has already reached the client, as a real and achievable goal for industry leaders to implement.

The models trained to distinguish between scammers and scam-baiters within the 4129DB met with limited success compared to the previous examples, achieving scores from varying scoring metrics around the 85% mark at best. This is still a perfectly acceptable score, however more effective models may need to be created before the methods and models displayed in this project for that tasks can become widespread.

The LDA models were able to perform topic clustering on the messages used by Scammers and Scam-Baiters, both individually and combined. This process however took much

trial and error and the resulting topics were not easy to label, nor were they particularly detailed topics.

These topics were also not as informative as first hoped, especially once plotted over time. A proposed reason for this may be the contradicting nature of how LDA models determine topics, and the variations this project hoped to find in the usage of said topics over time.

5.3 Open Problems and Possible Extensions

Results and analysis from Section 4.2 indicate that the method via which the models for the purpose of distinguishing between Scammer and Scam-Baiter were trained and implemented may be flawed. This motivates further study into the field of natural language processing and training text classification models as these results can be improved upon.

Another area worthy of further exploration is the method by which this project identified topics within the 419DB. These topics were intended for use as markers that would allow commentary on how the use of language changed over time in scam conversations. In practice however, primarily due to the model used to perform topic clustering, the results found provided less insight than was hoped. Experimentation with different models to perform this topic clustering could produce more interesting and comment-worthy clusterings that better provided insight into how the use of language changed.

As mentioned previously in this Section 1.2, the 419DB has some flaws. These conversations are not 100% accurate scam conversations as they are not scam conversations with genuine victims. This leads to they idea that more accurate data and therefore more accurate models could be created if a database of genuine scam conversations could be formed. These conversations however would be much harder to source and therefore this task could take a large amount of time, perhaps too large to be worthwhile.

As an addition to the previous point, models could be trained using scam conversations originating from different forms of scam. The work done in this project only demonstrates the ability for ML models to identify scam conversations where the scam is the 419 (Advance Fee Fraud) scam. This scam is one of the most prevalent but it is not the only existing scam. This projects methodology could be replicated with data originating from different scams, to better train an all purpose scam filter.

Finally, one last possible extension of this work could be to demonstrate the models trained in this project functioning as actual email filters. So far the proposed use case of these models is a hypothetical. Further work that demonstrated these models and how they could be practically implemented to function as email filters would make the proposed use case of these models and insights being adopted by the cyber security industry, closer to reality.

Bibliography

[1] https://www.419eater.com/.

[2] word2vec code and documentation archive. https://code.google.com/archive/p/word2vec/.

[3] An old swindle revived; the "spanish prisoner" and buried treasure bait again being offered to unwary americans. *The New York Times*, 1898.

[4] Suicide of internet scam victim. *BBC News*, 2004.

[5] Armies of expensive lawyers, replaced by cheaper software. *The New York Times*, 2011.

[6] Adam Binks. The art of phishing: past, present and future. *Computer Fraud & Security*, 2019(4):9–11, 2019.

[7] Enrico Blanzieri and Anton Bryl. A survey of learning-based techniques of email spam filtering. *Artificial Intelligence Review*, 29(1):63–92, 2008.

[8] Richard G Brody, Sara Kern, and Kehinde Ogunade. An insider's look at the rise of nigerian 419 scams. *Journal of Financial Crime*, 2020.

[9] Joshua Chang. An analysis of advance fee fraud on the internet. *Journal of Financial Crime*, 15:71–81, 01 2008.

[10] Gordon Cormack and Thomas Lynam. Trec 2005 spam track overview. 01 2005.

[11] Matthew Edwards, Claudia Peersman, and Awais Rashid. Scamming the scammers: towards automatic detection of persuasion in advance fee frauds. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 1291–1299, 2017.

[12] Geeks for Geeks. Naive bayes classifiers, 2020. [Online; accessed 10-June-2021].

[13] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[14] Jelena Isacenkova, Olivier Thonnard, Andrei Costin, Aurélien Francillon, and David Balzarotti. Inside the scam jungle: A closer look at 419 scam email operations. *EURASIP Journal on Information Security*, 2014(1):1–18, 2014.

[15] James Joyce. Bayes' Theorem. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Spring 2019 edition, 2019.

[16] Wazir Khan, Muhammad Khan, Fahad Muhaya, and Mohammed Aalsalem. A comprehensive study of email spam botnet detection. *IEEE Communications Surveys Tutorials*, 17:1–1, 01 2015.

[17] Youngsam Park, Jackie Jones, Damon McCoy, Elaine Shi, and Markus Jakobsson. Scambaiter: Understanding targeted nigerian scams on craigslist. *system*, 1:2, 2014.

[18] Heloise Pieterse and Martin S Olivier. Android botnets on the rise: Trends and characteristics. In *2012 information security for South Africa*, pages 1–5. IEEE, 2012.

[19] Towards Data Science. Logistic regreesion explained, 2020. [Online; accessed 10-June-2021].

[20] Charles Tive. *419 scam: Exploits of the Nigerian con man*. iUniverse, 2006.

[21] Peter Willett. The porter stemming algorithm: then and now. *Program*, 2006.