Email Authorship Identification

Mohammad Ali Khaksar mkhaksa
Sumitosh Pal spal3
Juan Jose Rivera jrivera5
Preya Shabrina pshabri

Abstract

With the growing usage of email as a medium of communication for official and unofficial purposes, their illegal usage has become a major concern for the users. Everyday millions of anonymous emails or online messages are exchanged that sometimes have the aim to trap naive users. Many terrorist societies hide their identities and share their views with mass people via email and a lot of times succeed to induce people to join their society. For these reasons, researchers have paid attention to the process of separating emails sent from safe and unsafe sources which involves authorship identification or at least grouping emails that are written or sent by the same author. To address this critical problem, we develop a framework that would identify authors of emails using Stylometric features extracted by `stylo` package. We use *Bag-of-Words* and *n-grams* feature sets to analyse the context in which a word is used. After obtaining these features, we train our models for authorship attribution using several machine learning models including BBN, KNN and SVM. We carry out a comparative analysis on different models to find out the best one. In our study, we also include Delta Metrics that have been proven to be robust, especially for authorship attribution.

1 Introduction & Background

Email is the most common medium of communication today. Everyday millions of emails are exchanged for both official and unofficial purposes. With the growing usage of email, email abuses are also increasing. Email senders can easily hide or misrepresent their identities to exploit the naivety of receivers. Thus author identification from the contents of email becomes important in order to separate the email from trusted and untrusted sources.

Considering the prevalent problem mentioned above as our motivation, we have decided to apply machine learning techniques on the Enron dataset to identify authors of emails in this project. To develop our authorship identification framework, we would consider some common algorithms(BBN, Naive Bayes, KNN, SVM) as well as a less commonly used algorithm (Delta Metrics) and perform a comparative study to choose the best one.

Machine learning plays a key role to analyze the email authorship identification problem and also has a long history that starts with Stylometry. Stylometry is a statistical method that identifies the stylistic features in writing. These features can be used to train a machine learning model that would identify or at least separate the author of one email from another. Previous works were mostly focused on applying state-of-the-art machine learning algorithms to achieve the goal.

Nizamani and Memon employed a cluster based classification technique to identify email authors where they extended Stylometry feature set with some interesting features like the last punctuation mark used in an email, the tendency of an author to use capitalization at the start of an email, or the punctuation after a greeting, farewell, etc. They also employed info gain based feature selection method. Their proposed model outperformed SVM based classification technique and attained an accuracy of 81% for 50 authors on Enron dataset, [11].

Zheng et al. applied inductive learning algorithms with message features including style markers, structural features and content-specific features to identify cyber criminals from the messages they post on the internet, [13].

Chen et al. proposed a frequent pattern and machine learning based model to find authorship similarity using Enron email dataset. They detected 150 stylistic cues for the problem, [5].

Mikros and Perifanos explored the adequacy of using features from different linguistic levels to identify authorship of emails based on regularized logistic regression and one-class SVM, [10].

Bracardo et al. employed supervised learning technique combined with n-gram analysis for authorship verification in short texts. They used Enron email dataset and Stylometry as their main features. With a supervised learning technique they tried to overcome the shortcomings of Stylometry in scenarios when messages are short and author population is large. They obtained an error rate of 14.35% for message blocks of 500 characters, [4].

Abdallah et al. mined anonymous email content with several machine learning algorithms to identify unique writing style for each suspect. The concentration of their study was to find a small but effective set of features to accomplish the goal, [3].

Abbasi and Chen analyzed Arabic and English Web forum messages posted by known extremist groups with a special multilingual model. The model was used to identify the language's unique characteristics to identify a terrorist's writing style for intelligence commmunity, [2]. They also developed an authorship visualization tool called Writeprints (Ahmed Abbasi, Hsinchun Chen. (2006)) that outperformed SVM in identifying authors of email messages. They used principal component analysis(PCA) based technique that employs a dynamic feature-based sliding window algorithm to perform the task, [1].

Juola and Baayen experimented on a pre-existing set of Dutch corpus of university writing and measured linguistic distances applying a cross-entropy technique that is sensitive to the distributions of language features and also to their relative inter-sequencing which helps to make classification judgments with great sensitivity, significance, confidence, and accuracy, [8].

Koppel et al. introduced a method of author attribution in the wild where the message length is short and the population of known candidates is large. They use similarity-based methods with multiple randomized feature sets to achieve high accuracy, [9].

In this paper, we would extract features using `stylo` package which is a new package for stylometric feature extraction, available in both R and Python. We would utilize 'Bag-of-Words' and 'n-grams' feature set to take into consideration the context in which the words in a email corpus are used. Primarily, we would apply BBN, KNN, Delta Metrics and SVM to train different models. Then we would perform a comparative study on the run-time and outcome of using different feature sets and machine learning models to solve the authorship identification problem.

2 Proposed method

The purpose of this project is to compare different classification techniques that could be used for *Email Authorship Identification*. As the format of the dataset that we want to explore is unstructured, there are important preprocessing steps that need to be done before the comparison of the classification techniques.

Figure 1 shows a high level workflow that explains the sequence of steps that we are going to perform in the context of the current project.

2.1 Data Cleaning

This is the initial step of our data preprocessing step. We take the files that contain the raw data and with the help of APIs provided by python, create our dataset. There are several types of emails that we have initially, including conversations, inbox, sent items, etc. There are several types such as standalone emails where the data contains the metadata and the email content of only one sender and recipient. Other types include emails with a series of conversations between two or more parties.

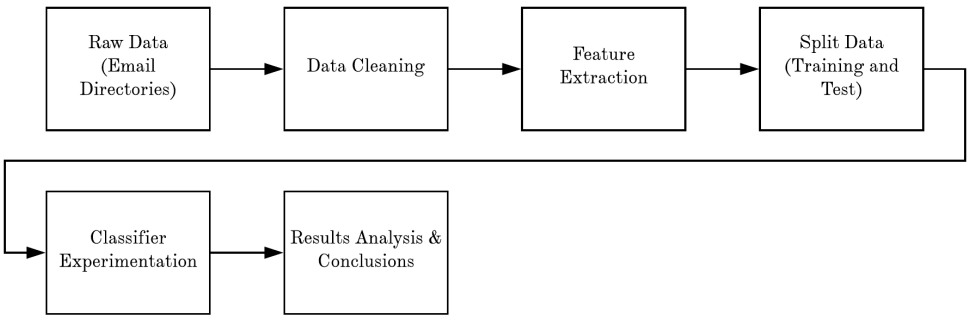


Figure 1: Proposed workflow for email authorship identification

The objective of this cleaning process is to get only the author and the authors message from any type of email that is initially present in the raw data. Using the script we parse the email surrounded by metadata to help assign a label to these email bodies. This is done with the help of `email.parser` packages to extract the label *'from'* from metadata and the *'body'* in standalone emails and our own script for emails with conversations. From this we can identify the author of that text and assign a class to it. This gives us an unstructured clean data which contains the labels author required for the next step of the project which is feature extraction.

2.2 Feature Extraction

Given an unstructured clean data and a label that represents the author of that text, we continue preprocessing of this clean data to extract features that allow the classification algorithm to learn and classify the observations properly.

In order to do this we would perform multiple steps using the `stylo` package in R.

1. Tokenize each observation, removing stop words[1], given that these words do not add much value to our analysis.
2. Once we have our tokenized observations, we would try two different methods to extract features from each observation: 1) Bag-of-Words, and 2) n-grams[2], both are different representations of the observation in a feature vector. We would have a list of each word or 'n-gram' that we find on each observation. We would then apply both these approaches, given that while 'Bag-of-Words' adds simplicity to our analysis, 'n-grams' approach provides us with context in which a particular word was used.
3. Now that we have extracted the above-mentioned features from each observation, we would build a relative frequency table, in which each row will represent an observation, each column the feature that we want to analyze and each cell the number of occurrences for that feature in a particular observation. We would use relative frequency to avoid any problem due to magnitude of frequency in a set of observations.
4. After extracting several number of features from our dataset, we would work with a representative subset of features. The n most frequent features across all the observations. This n value would be determined by performing exploratory data analysis on the total number of available features in the dataset and total occurrences of this features in the whole dataset. It is important to highlight that this value could be different based on the feature extraction method that is used. We anticipatie that for the 'n-grams' approach, the value will be greater due to the fact that in this approach, the feature space is sparser than the 'Bag-of-Words' approach.

[1]For this analysis stop words are considered as punctuation and pronouns.

[2]If we have the following observation 'this is an example'. The 2-gram (bigram) representation is : 'this is','is an','an example'

At the end of this step we would have a frequency table representing our observations with two possible set of features : 'Bag-of-Words' or 'n-grams'.

2.3 Split Data

The dataset object of this project contains a large number of observations that in some cases could be a factor that slows us down during our experimentation phase. For this reason and also to improve the run time of the classifiers that we want to validate, we would work with a representative smaller sample of the whole data set to train and test these classifiers. The sampling methodology that we are going to use would be a stratified sampling in order to avoid any possible class imbalance problem due to sampling issues.

In other words we would treat our data set as a real population and sample this population to validate and test our classifiers

The classifiers with the higher classification accuracy would be trained and tested using training and testing sets that involves the entire dataset.

2.4 Classifier Experimentation

In this step, we would explore different classification algorithms, some of these classifiers are implemented as part of the `stylo` package in R, and others would be implemented as part of this project. Each algorithm would be trained and tested using the two feature extraction methods mentioned above and using a subset of the whole dataset as training and test set. The accuracy of each of these algorithms would be calculated in both cases.

In order to achieve higher and better estimate along with the high accuracy, we would need to perform hyperparameter tuning on the algorithms to enable higher accuracy. This hyperparameter tuning would use a training, validation and test sets composed from the actual dataset.

List of algorithms that we would explore are presented below. All of these algorithms use feature vector as a representation of an observation.

1. Nearest Shrunken Centroid(NSC) supported by `stylo`.
2. K-NN, supported by `stylo`, with distance metrics such a: Euclidean, Manhattan, Classic Delta, Argamon's Linear Delta, Eder's Delta, Eder's Simple, Canberra and Cosine.
3. Naive Bayes, supported by `stylo`.
4. Support Vector Machine, supported by `stylo`.

Delta family of metrics have been claimed and proven to be very robust when it come to authorship attribution, [7]. In short, Delta metrics are variants of existing metrics, working on normalized feature vector. The normalization of each frequency for a particular feature is relative to the mean and standard deviation of that feature in the dataset, [6].
In addition, NSC has been shown to be quite effective as classifying vectors in high dimensional spaces,[12].

2.5 Results Analysis & Conclusions

After the development of the processes, we would be done with the following : clean raw data; extract features from unstructured data; split observations in different sets and; train, tune and test classifiers. We would then analyse the results and support our analysis and conclusions using classifier evaluation metrics such as Accuracy, Recall, Precision and AUC/ROC.

2.6 Rationale

Due to the complex state and the nature of our input dataset, we have decided to formulate the steps described above and implement the data extraction process using the specified feature vectors. Also, the data cleaning and filtering process generate a high magnitude of the dataset and retain the same number of classes from the original dataset i.e 150 employees. Hence, we have decided to work with a representative sample of the dataset for a subset of classes preserving the original distribution by

applying stratified sampling.
Moreover, the new generated dataset with the associated features exhibit high dimensionality and could pose as a potential bottleneck in our performance and experiment. Therefore, we would extract features out of this population subset, and use very reliable, well known and simple approaches to represent each observation: Bag-of-words and n-grams.
We have taken the following algorithms for classification tasks: Delta in text analysis[7]; and NSC in a different domain, identifying cancer using gene representation(high-dimensional representation)[12]. The intuition for selecting the algorithms came after analyzing the literature survey. According to the past results from several research papers and similar work, these algorithms and strategies demonstrated high accuracy for classification in similar tasks and domains. Finally, we would experiment using traditional Machine Learning methods in order to have a base line to compare with, and test our hypotheses.

3 Plan & Experiment

The goal of our experiment is to validate if Delta Algorithms and Nearest Shrunken Centroid(NSC) Algorithm outperform traditional Machine Learning Algorithms in the Authorship Identification domain using different types of feature vector to represent observations(e-mails).

3.1 Dataset

The dataset that is available to us initially is a collection of 150 user mailboxes containing 500,000 email raw files which are in MIME format. Each of the user mailboxes contain a collection of sub directories mostly organized as all inboxes, sent_items, inbox, presentations, deleted_items, savedmail, vanderbilt, conversations,etc which are the raw files containing email text. The average user mailbox contains around 3500 email raw files.
Each text file contains email meta data such as Message ID, Content-Type, Origin,cc, bcc, mime-version and finally the content. Some files are also a series of conversations between the sender and recipient in the same email thread which also have several metadata of their own.

3.2 Hypothesis

Based on the current work and existing research on text Authorship Identification[7],and classification algorithms in high-dimensional spaces[12] we have hypothesized the following:

- Delta metrics Algorithms together with feature extraction mechanisms such as Bag-of-Words and n-grams will outperform traditional machine learning approaches.
- NSC Algorithm together with feature extraction mechanisms such as Bag-of-Words and n-grams will outperform traditional machine learning approaches.

3.3 Experimental Design

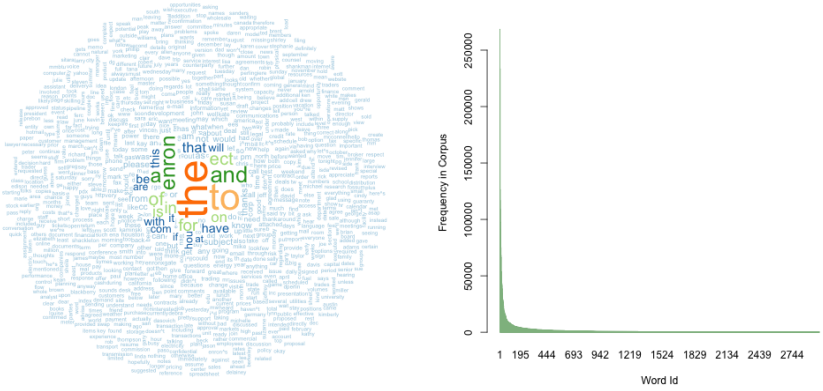
3.3.1 Data Extraction

The initial step of the experiment is to start collecting the raw files and start parsing them. The raw files are parsed using the email.parser package that is provided by python to segregate the metadata in the text message. After parsing of raw data we apply basic string filter operations to get the actual message written by the author which is the clean and unstructured data. After getting the clean but unstructured data, features would be derived from the content to add meaning to the dataset and use for further classification using machine learning classifiers. The output of this process is a csv file containing ≈ 96000 observations and the the following structure: "Class(Author's Name)", "E-mail Body".

3.3.2 Data Analysis

To continue with our analysis, we need to transform the data into a format that allows us to manipulate and extract useful metrics out of it. We are using `stylo` package in R to perform transformation and cleaning task on the dataset. We have tokenized every observation to analyze the data, after this we applied different methods to explore the data and identify relevant characteristic of this data.

Figure 2, shows word cloud and the corresponding word frequency for each word in the corpus. We can observe that there are words that have high frequency(≈ 250000) on the corpus, most of these words are called stopwords[3] and this words are very likely to not add relevant information on each observation.



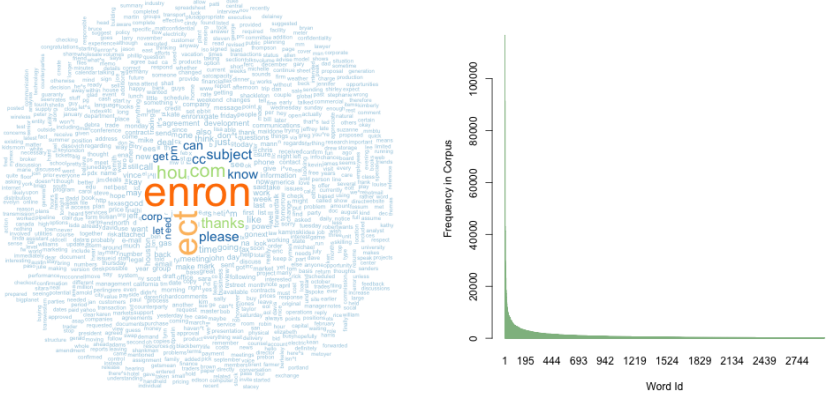
(a) Word-cloud before removing stop words

(b) Word freq. before removing stop words

Figure 2: Word-cloud after stop words removal

3.3.3 Data Cleaning

In order to remove these stopwords from our corpus, we have used `stylo` and `stopwords` packages in R. Figure 3, shows the new distribution of words in the corpus, and new set of corresponding frequencies. We noticed that the set of words: "the", "to", "and", are not part of the corpus anymore and are considered as stopwords.



(a) Word-cloud after removing stop words

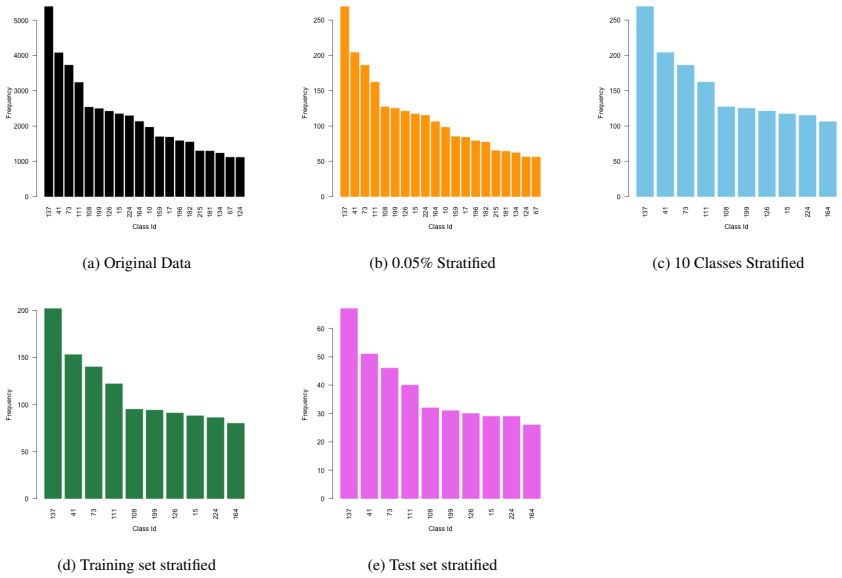
(b) Word freq. after removing stop words

Figure 3: Word-cloud before stop words removal

3.3.4 Data Sampling

So far we have cleaned each observation by removing stopwords, but we still have a large corpus in terms of number of observation and classes. Given the magnitude of data it is difficult to analyze and process it , hence we have decided to extract a representative subset of this corpus. In order to achieve this we have used `caTools` R package to get Stratified samples out of the corpus. We have

[3]Stopword, are the most common words in a language



(a) Original Data (b) 0.05% Stratified (c) 10 Classes Stratified

(d) Training set stratified (e) Test set stratified

Figure 4: Data Sampling Scheme

decided to work with 10 Classes over 0.05% of the whole population.
Figure 4, shows the sampling schema that we have used from Original data set to training and test set to perform the experimentation.

3.3.5 Feature Extraction

We have used two different feature extraction methods for this corpus: Bag-of-words and n-grams. We have done experimentation with different flavors of each one.Bag-of-Words:w/ stopwords, w/o stopwords(Snowball), w/o stopwords(Smart); n-grams: n=3000 and n=6000. We used `stylo` and `stopwords` R packages to perform this. For each feature extraction method we have generated a different feature vector of frequencies, which is the representation of each observation. This process requires the use of highly intensive computing solutions, therefore we used AWS instances and performed experimentation with 2 different types of hardware which are specified in the Appendix E.

3.3.6 Classifier Experimentation

We have used the five different types of feature vector as input to train and test each algorithm that we have in scope for this project. We also performed hyperparameter tuning to get the set of parameters that achieve the best accuracy in each algorithm, this is shown in Appendix D. The algorithms that we have used for experimentation are wrapped or written in the `stylo` package in R.
During the experimentation we have observed some unusual behavior with `stylo` Delta Algorithms (function `perform.delta`). After analyzing the library, we also identified a bug, that we have fixed as part of this project. This is shown on Appendix B.

4 Results & Discussion

Table 1, shows the max accuracy achieved for each Algorithm using different types of feature vector, the sample used is Stratified Sample of 10 Classes over the 0.05% of the whole population.
The Algorithm that present the highest accuracy together with BoW feature vector representation is NSC. This confirms that the algorithm that has been used in cancer prediction domain with vector gene representation, has a good performance using a similar representation as feature vector, that could be understood as a high dimensional vector.
On the other hand we have observed that the second most accurate algorithm is the one that used Canberra as distance on a normalized vector. As described in the table this is the same as K-NN with Canberra distance but on a normalized feature vector. The algorithms that implement the delta

metrics have not been accurate in comparison with these two: Delta.Canberra and NSC. We also observed from our results that BoW generally has a better accuracy in comparison with n-gram. We know that n-gram preserve the context in which each word has been used. We could infer that giving the level of sparsity of the vector n-gram to these algorithms are not achieving accurate results.
In theory Delta algorithms should perform better that the results of our experimentation. We believe that improving the feature selection techniques of the experiment using bestsubset or stepwise; and using different techniques to extract features out of our corpus like NLP-based features or different stylometric metrics such as: average sentence length,average word length and ratio of unique words; could improve the current results of our experiment.

4.1 Classification Results Table

This table shows the average result of each algorithm after 10 runs. Details of each run could be found in Appendix C

	Accuracy									
	BoW						**n-gram**			
Algorithm	**BoW**	**SE**	**Snowball**	**SE**	**Smart**	**SE**	**3000**	**SE**	**6000**	**SE**
Delta-based Algorithms										
Delta.Delta	0.1774	0.0116	0.1052	0.0138	0.2219	0.0075	0.3137	0.0077	0.2860	0.0077
Delta.Eder	0.1740	0.0148	0.1787	0.0168	0.2394	0.0070	0.3395	0.0080	0.3086	0.0080
Delta.Argamon	0.2168	0.0159	0.2207	0.0069	0.2614	0.0057	0.3048	0.0074	0.2839	0.0074
Delta.Simple	0.3136	0.0144	0.2887	0.0068	0.3572	0.0095	0.4191	0.0056	0.4022	0.0056
Delta.Cosine	0.5646	0.0138	0.6903	0.0049	0.7152	0.0054	0.5223	0.0072	0.5497	0.0072
Delta.Wurzburg	0.5919	0.0067	0.5898	0.0070	0.5913	0.0081	0.4793	0.0058	0.4978	0.0058
Delta.Entropy	0.2858	0.0111	0.2782	0.0076	0.3486	0.0062	0.4118	0.0059	0.3884	0.0059
Delta.Euclidean*	0.4759	0.0119	0.4850	0.0089	0.5045	0.0038	0.4546	0.0083	0.4478	0.0083
Delta.Manhattan*	0.4003	0.0391	0.2467	0.0095	0.2979	0.0068	0.3941	0.0076	0.3702	0.0076
Delta.Canberra*	**0.6612**	0.0073	**0.7213**	0.0065	**0.7449**	0.0076	**0.5355**	0.0077	**0.5570**	0.0077
Non Delta-based Algorithms										
K-NN	0.4748	0.0106	0.4903	0.0098	0.5034	0.0056	0.4849	0.0101	0.4696	0.0077
SVM	0.5850	0.0112	0.5745	0.0097	0.5323	0.0205	0.4102	0.0086	0.3970	0.01012
NSC	**0.7407**	0.0068	**0.7239**	0.0090	**0.7404**	0.0078	**0.5062**	0.0077	**0.5333**	0.0086
Naive-Bayes	0.1052	0.0061	0.0924	0.0049	0.0903	0.0044	0.0836	0.0005	0.0823	0.0077

Table 1: Classifier Comparison, average of 10 runs. (*) k-nn Algorithm on a normalized vector

5 Conclusions

In this project we have faced multiple challenges, that made us realize the complexity in working with unstructured data. We have observed that transformation and processing of this type of data is a high intensive process as we required the use of external environments that allows us to make progress in time that fits the timeline of this project.We have used AWS components as specified in Appendix E.
In addition, given that we addressed this project as a multiclass classification problem over a large population (number of Classes and number of observations), we used stratified sampling to get a representative sample of the population. This helped us to analyse smartly and focus on a population subset that holds the same properties and shape as the actual population.
We have also observed that in the case of feature vectors using BoW the process of removing stopsword improves the accuracy in the algorithms that present high accuracy. As areas of improvement, we have identified that some of the algorithms do not perform well with high-dimesional vectors. In our opinion, this problem can be solved by improving feature selection and feature engineering steps which could improve the existing results, and validate our hypothesis related to Delta.Algorithms and Delta distance metrics.

References

[1] A. Abbasi and H. Chen. Visualizing Authorship for Identification. pages 60–71. Springer, Berlin, Heidelberg, 2006.

[2] A. Abbasi, H. C. I. I. Systems, and u. 2005. Applying authorship analysis to extremist-group web forum messages. *ieeexplore.ieee.org*.

[3] E. Abdallah, A. Abdallah, M. B. . o. S. . . . , and u. 2013. Simplified features for email authorship identification. *researchgate.net*.

[4] M. L. Brocardo, I. Traore, S. Saad, and I. Woungang. Authorship verification for short messages using stylometry. In *2013 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pages 1–6. IEEE, 5 2013.

[5] X. Chen, P. Hao, R. Chandramouli, and K. P. Subbalakshmi. Authorship Similarity Detection from Email Messages. pages 375–386. Springer, Berlin, Heidelberg, 2011.

[6] S. Evert, T. Proisl, F. Jannidis, S. Pielström, C. Schöch, and T. Vitt. Towards a better understanding of Burrows's Delta in literary authorship attribution. Technical report, 2015.

[7] D. L. Hoover. Testing Burrows's Delta. *Literary and Linguistic Computing*, 19(4):453–475, 11 2004.

[8] P. Juola, R. B. L. Computing, Linguistic, and u. 2005. A controlled-corpus experiment in authorship identification by cross-entropy. *academic.oup.com*.

[9] M. Koppel, J. Schler, and S. Argamon. Authorship attribution in the wild. *Language Resources and Evaluation*, 45(1):83–94, 3 2011.

[10] G. Mikros, K. P. N. f. P. a. CLEF, and u. 2011. Authorship identification in large email collections: experiments using features that belong to different linguistic levels. *researchgate.net*.

[11] S. Nizamani and N. Memon. CEAI: CCM-based email authorship identification model. *Egyptian Informatics Journal*, 14(3):239–249, 11 2013.

[12] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. Technical report, 2002.

[13] R. Zheng, Y. Qin, Z. Huang, and H. Chen. Authorship Analysis in Cybercrime Investigation. pages 59–73. Springer, Berlin, Heidelberg, 2003.

A Github Repository

https://github.ncsu.edu/jrivera5/ALDA_Project

R Packages: `stylo`, `pmar`, `stopwords`, `wordcloud`, `caTools`, `caret`, `plyr`, `uniqtag`, `devtools`, `rlist`.

B Bug Fixed

Package: `stylo`, Function: `perform.delta()`, File: `perform.delta.R`

Original

```
for(h in 1:length(selected.dist[,1])) {
        ranked.c = order(selected.dist[h,])[1:no.of.candidates]
        current.sample = classes.training.set[ranked.c[1]]
        classification.results = c(classification.results, current.sample)
        #
        current.ranking = classes.training.set[ranked.c]
        current.scores = selected.dist[h,ranked.c]
        classification.scores = rbind(classification.scores, current.scores)
        classification.rankings = rbind(classification.rankings, current.ranking)
}
```

Fixed

```
for(h in 1:length(selected.dist[,1])) {
  ranked.c = order(selected.dist[h,])[1:no.of.candidates]

  #current.sample = classes.training.set[ranked.c[1]]
  #classification.results = c(classification.results, current.sample)
  #
  current.ranking = classes.training.set[ranked.c]

  #Begin P09 JuanJoseRivera Modification
  freq_table<-table(current.ranking)
  current.sample<-names(freq_table)[which (freq_table==max(freq_table))][1]
  classification.results = c(classification.results, current.sample)
  #End P09 JuanJoseRivera Modification

  current.scores = selected.dist[h,ranked.c]
  classification.scores = rbind(classification.scores, current.scores)
  classification.rankings = rbind(classification.rankings, current.ranking)
}
```

C Experimentation Results

Here we have run each Classifier 10 times with different observation representation (BoW, n-grams) and with different population samples(first 10 columns represent a different population sample).

C.1 BoW

```
> bow
                  1         2         3         4         5         6         7         8         9        10       mean         sd          se
Delta.Delta  0.1627297 0.1942257 0.2204724 0.08923885 0.1942257 0.1837270 0.17060367 0.21522310 0.17585302 0.1679790 0.1774278 0.03654070 0.011555182
Delta.Eder   0.2020997 0.1994750 0.2283464 0.08923885 0.2020997 0.1916010 0.15223097 0.09711286 0.19947507 0.1784777 0.1740157 0.04682177 0.014806345
Delta.Argamon 0.2152231 0.2362204 0.2992126 0.09448819 0.2257218 0.2152231 0.22309711 0.23622047 0.21784777 0.2047244 0.2167979 0.05030735 0.015908582
Delta.Simple 0.3622047 0.3720341 0.3858267 0.30446194 0.2965879 0.2965879 0.24671916 0.29658793 0.26771654 0.3070866 0.3136483 0.04551534 0.014393214
Delta.Cosine 0.5328084 0.6325459 0.6325459 0.55905512 0.5616798 0.5301837 0.56692913 0.58530184 0.55118110 0.4934383 0.5645669 0.04364731 0.013802492
Delta.Wurzburg 0.5669291 0.6062992 0.6167979 0.58005249 0.5905512 0.5695538 0.62992126 0.58267717 0.60104987 0.5748031 0.5918635 0.02107924 0.006665840
Delta.Entropy 0.3202100 0.3070866 0.3385827 0.30183727 0.2729659 0.2992126 0.23359580 0.29396325 0.24671916 0.2440945 0.2858268 0.03509282 0.011097325
Delta.Euclidean 0.4671916 0.5275906 0.5039370 0.50131234 0.4855643 0.4173228 0.45931759 0.51968504 0.44094488 0.4356955 0.4758530 0.03761114 0.011893687
Delta.Manhattan 0.5616798 0.4829396 0.4960629 0.54068241 0.2467192 0.4304462 0.38582677 0.25721785 0.22834646 0.3727034 0.4002625 0.12349446 0.039052378
Delta.Canberra 0.6614173 0.6771653 0.6876640 0.64041995 0.6561680 0.6482940 0.69291339 0.67979003 0.64829396 0.6194226 0.6611549 0.02319533 0.007335007
K-NN         0.4750656 0.5196850 0.5013123 0.48556430 0.4829396 0.4251969 0.46194226 0.51968504 0.43832021 0.4383202 0.4748031 0.03366795 0.010646740
SVM          0.5931759 0.6167979 0.5538057 0.55643045 0.6115486 0.5196850 0.59580052 0.64041995 0.59317585 0.5695538 0.5850394 0.03548326 0.011220792
NSC          0.7480315 0.7611548 0.7742782 0.71128609 0.7401575 0.7270341 0.72178478 0.75853018 0.75065617 0.7139108 0.7406824 0.02145891 0.006785904
Naive-Bayes  0.1338583 0.0866141 0.0997375 0.12335958 0.1128609 0.1102362 0.07874016 0.08661417 0.09186352 0.1286089 0.1052493 0.01938431 0.006129856
```

C.2 BoW Snowball

```
> bow_snowball
                  1         2         3         4         5         6         7         8         9        10       mean         sd          se
Delta.Delta  0.2047244 0.10236220 0.10498688 0.2204724 0.1811024 0.2309711 0.17847769 0.1942257 0.1732283 0.19685039 0.17874016 0.04348919 0.013752491
Delta.Eder   0.2099738 0.10236220 0.10498688 0.2257218 0.1049869 0.2362205 0.19160105 0.2099738 0.2047244 0.19685039 0.17874016 0.05306556 0.016780804
Delta.Argamon 0.2204724 0.19947507 0.22309711 0.2388451 0.2020997 0.2572178 0.19685039 0.2257218 0.1968504 0.24671916 0.22073491 0.02187052 0.006916064
Delta.Simple 0.2703412 0.25984252 0.27034121 0.3018373 0.2598425 0.3070866 0.31233596 0.2965879 0.2939633 0.31496063 0.28871391 0.02157275 0.006821903
Delta.Cosine 0.6929134 0.71653543 0.66404199 0.7060367 0.6692913 0.6902887 0.69553806 0.6850394 0.6955381 0.68766404 0.69028871 0.01550310 0.004902511
Delta.Wurzburg 0.5564304 0.57480315 0.58792651 0.5905512 0.5774278 0.5616798 0.60367454 0.6194226 0.6089239 0.61679790 0.58976378 0.02223841 0.007032401
Delta.Entropy 0.2729659 0.24934383 0.26509186 0.2913386 0.2309711 0.2939633 0.30183727 0.2992126 0.2755906 0.30183727 0.27821522 0.02415079 0.007637149
Delta.Euclidean 0.4908136 0.42519685 0.46981627 0.4724409 0.4619423 0.5118110 0.50131234 0.5013123 0.4960630 0.51968504 0.48503937 0.02810020 0.008886064
Delta.Manhattan 0.2335958 0.22834646 0.24409449 0.2782152 0.1758530 0.2703412 0.25721785 0.2545932 0.2493438 0.27559055 0.24671916 0.03000250 0.009487623
Delta.Canberra 0.7165354 0.75065617 0.70341207 0.7427822 0.6850394 0.7165354 0.70866142 0.7270341 0.7454068 0.71653543 0.72125984 0.02054783 0.006497793
K-NN         0.4724409 0.42519685 0.49868766 0.4750656 0.4645669 0.5223097 0.51968504 0.5091864 0.4960630 0.51968504 0.49028871 0.03102595 0.009811266
SVM          0.5538058 0.55118110 0.60367454 0.5590551 0.5800525 0.5616798 0.55118110 0.5433071 0.6062992 0.63517060 0.57454068 0.03066988 0.009698667
NSC          0.6692913 0.74015748 0.70866142 0.7244094 0.7191601 0.7349081 0.74540682 0.6902887 0.7664042 0.74015748 0.72388451 0.02845211 0.008997347
Naive-Bayes  0.1154856 0.09973753 0.08923885 0.1049869 0.0839895 0.1102362 0.09448819 0.0656168 0.0839895 0.07611549 0.09238845 0.01564073 0.004946035
```

C.3 BoW Smart

```
> bow_smart
                  1         2         3         4         5         6         7         8         9        10       mean         sd          se
Delta.Delta  0.23884514 0.1968504 0.20209974 0.24934383 0.18635171 0.2362205 0.19422572 0.23884514 0.23359580 0.24146982 0.22178478 0.02380764 0.007528636
Delta.Eder   0.27559055 0.2152231 0.22572178 0.25459318 0.24934383 0.2467192 0.19685039 0.24671916 0.23622047 0.24671916 0.23937008 0.02209164 0.006985989
Delta.Argamon 0.26509186 0.2309711 0.26771654 0.30183727 0.25721785 0.2598425 0.25459318 0.27034121 0.24934383 0.25459318 0.26115486 0.01811045 0.005727026
Delta.Simple 0.35170604 0.3333333 0.37007874 0.33858268 0.35958005 0.3595801 0.31233596 0.42257218 0.34383202 0.38057743 0.35721785 0.03006494 0.009507369
Delta.Cosine 0.74015748 0.7375328 0.69816273 0.70866142 0.71391076 0.7191601 0.70341207 0.70078740 0.69553806 0.73490814 0.71522310 0.01697603 0.005368294
Delta.Wurzburg 0.60104987 0.5931759 0.57217848 0.62467192 0.62204724 0.5590551 0.58792651 0.55905512 0.57217848 0.62204724 0.59133858 0.02565832 0.008113874
Delta.Entropy 0.35170604 0.3412073 0.36482940 0.32283465 0.35433071 0.3543307 0.31233596 0.37532808 0.34120735 0.36745407 0.34855643 0.01971127 0.006233250
Delta.Euclidean 0.51181102 0.5039370 0.50131234 0.51443570 0.48031496 0.5065617 0.49343832 0.51706037 0.49606299 0.51968504 0.50446194 0.01217325 0.003849519
Delta.Manhattan 0.30971129 0.2782152 0.30446194 0.29921260 0.31233596 0.3070866 0.26771654 0.30183727 0.26509186 0.33333333 0.29790026 0.02140355 0.006768397
Delta.Canberra 0.75065617 0.7690289 0.71128609 0.77952756 0.73490814 0.7217848 0.75853018 0.71391076 0.74278215 0.76640420 0.74488189 0.02401730 0.007594938
K-NN         0.52755906 0.5118110 0.48293963 0.51443570 0.48031496 0.4855643 0.49081365 0.52493438 0.50131234 0.51443570 0.50341207 0.01761987 0.005571893
SVM          0.53280840 0.5958005 0.53805774 0.51706037 0.57742782 0.5695538 0.36220472 0.52493438 0.54330709 0.56167979 0.53228346 0.06471589 0.020464962
NSC          0.72178478 0.7427822 0.71653543 0.76902887 0.74803150 0.7165354 0.73490814 0.71391076 0.75328084 0.78740157 0.74041995 0.02466664 0.007800277
Naive-Bayes  0.08136483 0.1286089 0.08136483 0.08661417 0.08136483 0.0839895 0.08923885 0.09186352 0.08661417 0.09186352 0.09028871 0.01406371 0.004447337
```

C.4 ngram 3000

```
> ngram_3000
                    1         2         3         4         5         6         7         8         9        10        mean          sd          se
Delta.Delta  0.30645161 0.33870968 0.28225806 0.33870968 0.31451613 0.28763441 0.29032258 0.29569892 0.33870968 0.34408602 0.31370968 0.024475655 0.0077398817
Delta.Eder   0.35752688 0.37634409 0.31989247 0.36290323 0.32795699 0.30913978 0.30376344 0.32526882 0.35483871 0.35752688 0.33951613 0.025187023 0.0079648360
Delta.Argamon 0.28225806 0.33602151 0.30645161 0.33870968 0.29569892 0.26881720 0.29838710 0.29032258 0.33064516 0.30107527 0.30483871 0.023441792 0.0074129455
Delta.Simple 0.42473118 0.44086022 0.41397849 0.44354839 0.42204301 0.39784946 0.39784946 0.40053763 0.43817204 0.41129032 0.41908602 0.017670742 0.0055879791
Delta.Cosine 0.52688172 0.55107527 0.54032258 0.51344086 0.51881720 0.47849462 0.51075269 0.50806452 0.55645161 0.51881720 0.52231183 0.022846827 0.0072248010
Delta.Wurzburg 0.48118280 0.50000000 0.45430108 0.49731183 0.48655914 0.47311828 0.45698925 0.46774194 0.50806452 0.46774194 0.47930108 0.018453126 0.0058353907
Delta.Entropy 0.41397849 0.44086022 0.41129032 0.43817204 0.40860215 0.38172043 0.39247312 0.40053763 0.42473118 0.40591398 0.41182796 0.018744526 0.0059275396
Delta.Euclidean 0.46236559 0.50000000 0.46236559 0.44892473 0.44086022 0.41129032 0.43279570 0.44623656 0.49193548 0.44892473 0.45456989 0.026352314 0.0083333333
Delta.Manhattan 0.39516129 0.42741935 0.38978495 0.42741935 0.38978495 0.35215054 0.37096774 0.38440860 0.41666667 0.38709677 0.39408602 0.024050420 0.0076054105
Delta.Canberra 0.54569892 0.56451613 0.55376344 0.52419355 0.54032258 0.50537634 0.50537634 0.52419355 0.57526882 0.51612903 0.53548387 0.024335835 0.0076956669
K-NN         0.48655914 0.54838710 0.50806452 0.47849462 0.44623656 0.45698925 0.45698925 0.45967742 0.51344086 0.49462366 0.48494624 0.031988118 0.0101155310
SVM          0.38440860 0.43010753 0.39516129 0.42741935 0.40053763 0.40053763 0.39784946 0.41129032 0.47311828 0.38172043 0.41021505 0.027273136 0.0086245229
NSC          0.50268817 0.55376344 0.48387097 0.52956989 0.49731183 0.48924731 0.48387097 0.48924731 0.53494624 0.49731183 0.50618280 0.024442828 0.0077295010
Naive-Bayes  0.08333333 0.08333333 0.08064516 0.08602151 0.08333333 0.08333333 0.08333333 0.08333333 0.08602151 0.08333333 0.08360215 0.001525931 0.0004825416
```

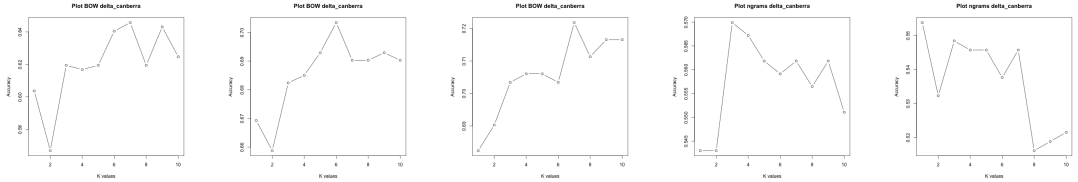
C.5 ngram 6000

```
> ngram_6000
                    1         2         3         4         5         6         7         8         9        10        mean          sd          se
Delta.Delta  0.27956989 0.32795699 0.25000000 0.26612903 0.31451613 0.29301075 0.29301075 0.29032258 0.26612903 0.27956989 0.28602151 0.023235373 0.0073476703
Delta.Eder   0.30645161 0.34677419 0.26344086 0.27956989 0.32526882 0.31182796 0.31989247 0.31989247 0.29301075 0.31989247 0.30860215 0.024170306 0.0076433219
Delta.Argamon 0.27688172 0.32526882 0.24193548 0.25806452 0.31182796 0.27956989 0.28763441 0.28763441 0.26075269 0.30913978 0.28387097 0.026222502 0.0082922834
Delta.Simple 0.38978495 0.41935484 0.36290323 0.37096774 0.42204301 0.42473118 0.40591398 0.41129032 0.38978495 0.42473118 0.40215054 0.022675740 0.0071706986
Delta.Cosine 0.55107527 0.56182796 0.52956989 0.52956989 0.55645161 0.56720430 0.55376344 0.52150538 0.55376344 0.57258065 0.54973118 0.017201034 0.0054394445
Delta.Wurzburg 0.50268817 0.51612903 0.47580645 0.47580645 0.49193548 0.51344086 0.51612903 0.47849462 0.50000000 0.50806452 0.49784946 0.016415223 0.0051909492
Delta.Entropy 0.37365591 0.41935484 0.34408602 0.37365591 0.40053763 0.39516129 0.38978495 0.40322581 0.36827957 0.41666667 0.38844086 0.023511903 0.0074351165
Delta.Euclidean 0.44354839 0.47580645 0.44086022 0.43279570 0.43279570 0.45161290 0.43279570 0.43279570 0.44086022 0.49462366 0.44784946 0.021098298 0.0066718676
Delta.Manhattan 0.36290323 0.39784946 0.33333333 0.35215054 0.39247312 0.37634409 0.36559140 0.37365591 0.35215054 0.39516129 0.37016129 0.021130621 0.0066820890
Delta.Canberra 0.54838710 0.56720430 0.52956989 0.54032258 0.56182796 0.56720430 0.58333333 0.54032258 0.56451613 0.56720430 0.55698925 0.016561312 0.0052371467
K-NN         0.47043011 0.48387097 0.45698925 0.45161290 0.48387097 0.46774194 0.44892473 0.47043011 0.44892473 0.51344086 0.46962366 0.020198095 0.0063871983
SVM          0.38709677 0.42204301 0.34408602 0.38709677 0.43010753 0.40591398 0.45967742 0.40053763 0.33870968 0.39516129 0.39704301 0.036728515 0.0116145763
NSC          0.54838710 0.54569892 0.52419355 0.51881720 0.51075269 0.52419355 0.54301075 0.53225806 0.52150538 0.56451613 0.53333333 0.016580693 0.0052432756
Naive-Bayes  0.08333333 0.08333333 0.08333333 0.08333333 0.07526882 0.08333333 0.08333333 0.08333333 0.08333333 0.08064516 0.08225806 0.002597021 0.0008212501
```

D Hyperparameter Tuning

All the non-nn classifiers we have used had hyperparameter tuning built in. In case of nn, On each sample run, we have run each Algorithm n times (n= number of classes, in this case = 10) to tune the hyperpameter K. Here we have shown plots for one of the k-nn Algorithms in a single run. As mentioned in Appendix C, we have run this experimentation 10 times so each nn algorithm for each observation representation has been executed 100 times. As we have 5 different observation representation(3 BoW and 2 n-grams) we have executed each nn algorithm 500 times.

D.1 Delta Classifier w/ distance Canberra



(a) w/ stop words (b) w/o Snowball (c) w/o Smart (d) n-gram 3000 (e) n-gram 6000

E Hardware Computation Metrics

AWS EC2 p2.xlarge | 1 GPU, 4 vCPUs, 61 GiB	AWS EC2 c4.8xlarge | 36 vCPU, 60 GiB