# Email Authorship Identification
## (on Enron dataset)

Automated Learning and Data Analysis

Group P09

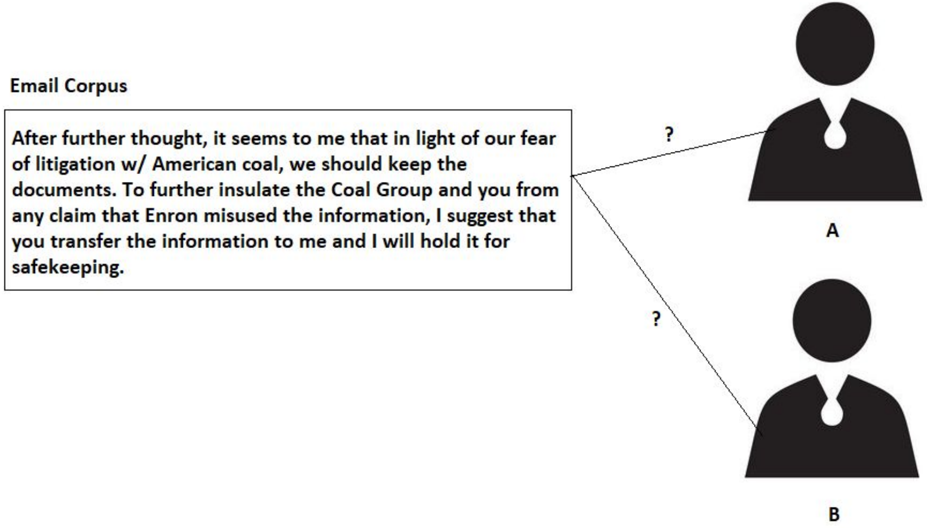
Mohammad Ali Khaksar – mkhaksa

Sumitosh Pal – spal3

Juan Jose Rivera – jrivera5

Preya Shabrina – pshabri

# Project Idea




- Our project is about identifying author of an email, using the stylometric characteristics of the email corpus.
- And we have also compared different classification algorithms.

## Related Works

- When was the first attempt taken to address the problem?

- The oldest research we found was in 2003

  **- Nizamani and Memon - cluster based classification technique**

  **- Zheng et al. applied inductive learning algorithms with message features including style markers**

  **- Chen et al. proposed a frequent pattern and machine learning based model**

- May be the first attempt was taken long before 2003

# Then why are we doing this again?

## Motivation



**Email Phishing scams cost American business 100 Billion dollars a year!!!**

## Motivation


Google and Facebook got duped out of **100 million dollars** through an email phishing scheme when a hacker impersonated a computer-parts vendor!!!

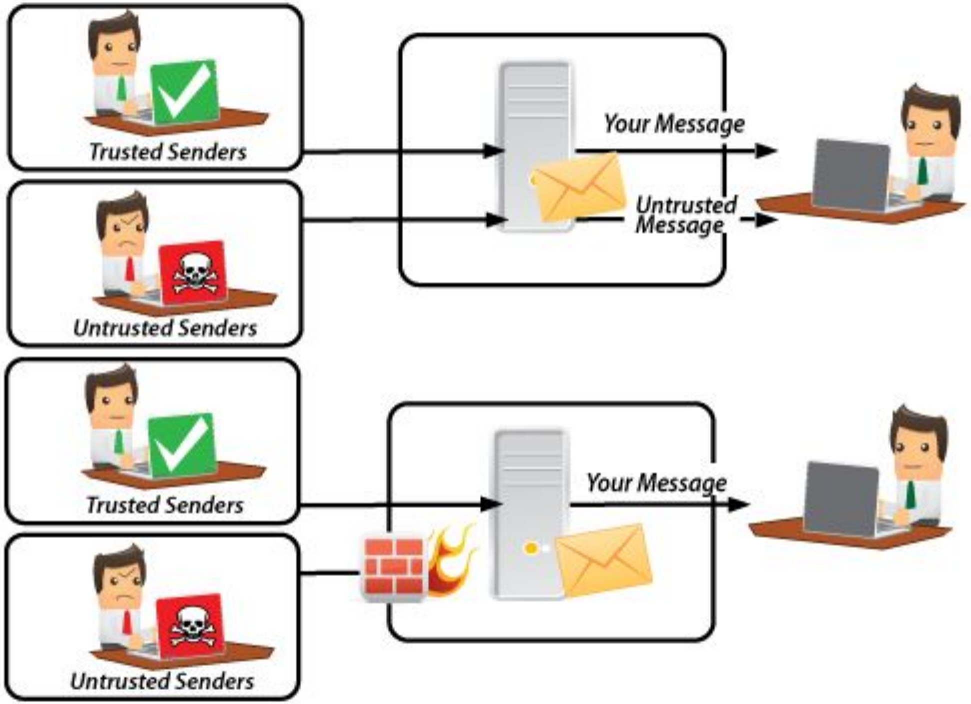
# Motivation

According to the FBI, criminals made off with at least **676 million** thanks to so-called business email compromise campaigns, which are attacks designed to trick company executives or accounting departments into sending money to fake vendors.

# Motivation

All these incidents happened in 2018!!!

# Motivation




- Email authorship identification is important to categorize emails from trusted and untrusted sources
- It's also important for identity validation
- It's hard but selecting an approach is even harder

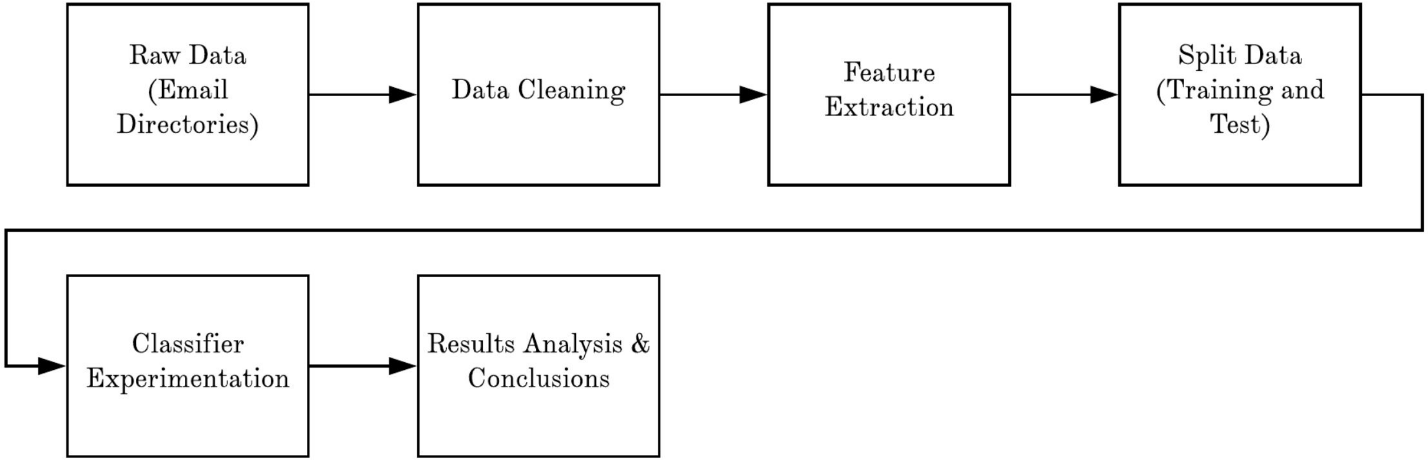
# What's new?

We tried to answer the following research questions:

- Will Delta metrics Algorithms together with feature extraction mechanisms such as Bag-of-Words and n-grams outperform traditional machine learning approaches?

- Will NSC Algorithm together with feature extraction mechanisms such as Bag-of-Words and n-grams outperform traditional machine learning approaches?

# Proposed Method



# Dataset Description

**Raw MIME format Email files**

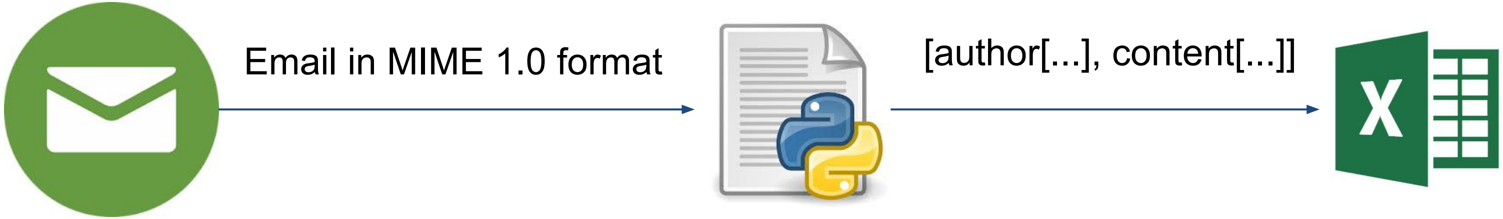
- Initially, the dataset comprised of 500,000 files from 150 authors which included files from sent items, sent mail, all documents, contact, inbox, deleted items, etc
- These files contained emails sent by the author, sent to the author, automated script mails and even spam.
- Each file is encoded in MIME version 1.0 which had metadata such as message ID, email id of sender, receiver, date sent, content-type,cc, bcc, etc.

# Data Extraction

- Only those emails that were composed by the author or the emails in which the metadata ‘from’ matched the authors name were selected and from sent, sent-items, sent mail folder category.

- Each of the file was parsed using email parser package in python to get the sender information as well as the mail body of the email.

- The set of author and their corresponding content was stored as a list which was finally stored in a CSV file with around 90000 rows for ‘author’ and ‘content’.



# Data Transformation

- Transform data to be loaded in to Package ‘stylo’.

Transform data to be loaded into package

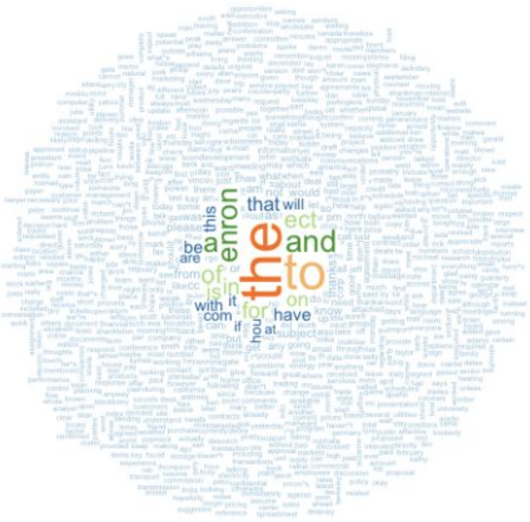
Load Corpus

Package ‘stylo’

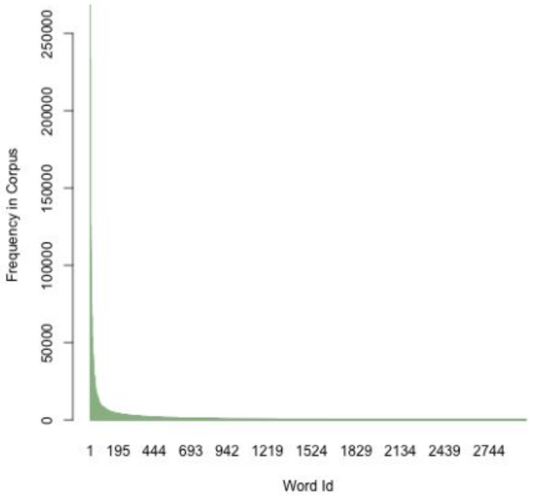
CSV

R script

# Word Frequency- Before Cleaning




(a) Word-cloud before removing stop words




(b) Word freq. before removing stop words

# Data Cleaning - Snowball Stopwords

- Snowball set of stopwords.

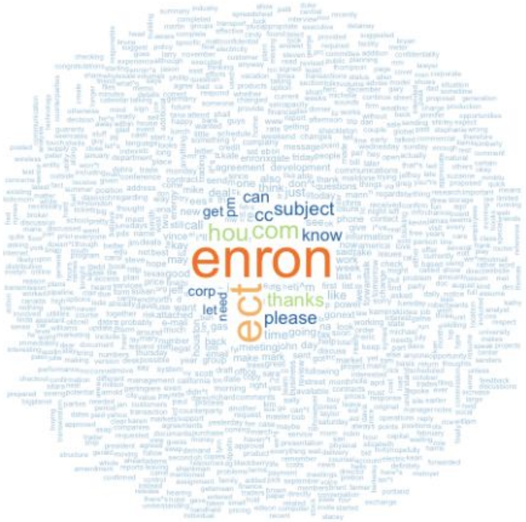
```
> length(stop_words_snowball)
[1] 175
> stop_words_snowball
  [1] "i"          "me"         "my"         "myself"     "we"         "our"        "ours"       "ourselves"
  [9] "you"        "your"       "yours"      "yourself"   "yourselves" "he"         "him"        "his"
 [17] "himself"    "she"        "her"        "hers"       "herself"    "it"         "its"        "itself"
 [25] "they"       "them"       "their"      "theirs"     "themselves" "what"       "which"      "who"
 [33] "whom"       "this"       "that"       "these"      "those"      "am"         "is"         "are"
```

# Data Cleaning - Smart Stopwords

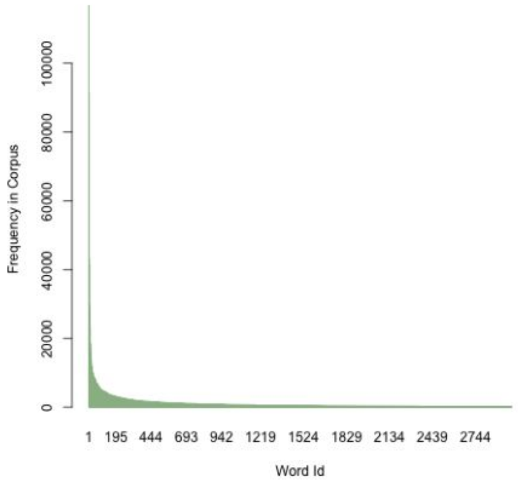
- Smart set of stopwords.

```
> length(stop_words_smart)
[1] 571
> stop_words_smart
  [1] "a"            "a's"          "able"         "about"        "above"        "according"
  [7] "accordingly"  "across"       "actually"     "after"        "afterwards"   "again"
 [13] "against"      "ain't"        "all"          "allow"        "allows"       "almost"
 [19] "alone"        "along"        "already"      "also"         "although"     "always"
 [25] "am"           "among"        "amongst"      "an"           "and"          "another"
 [31] "any"          "anybody"      "anyhow"       "anyone"       "anything"     "anyway"
```

# Word Frequency - After Cleaning



(a) Word-cloud after removing stop words

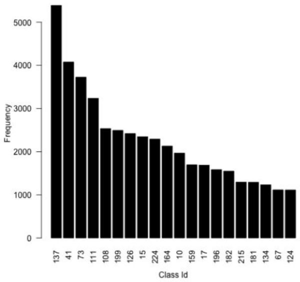



(b) Word freq. after removing stop words

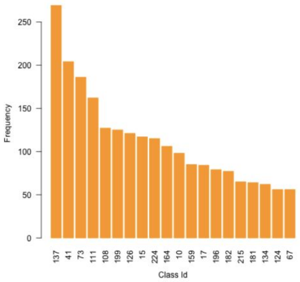
# Data Sampling

- Large Dataset
- Stratified Sampling to keep distribution from this Authors.
- From 5% of Population and 10 Author Sample (≈1500 emails)
- Same Sampling method to get training and test set (0.75/0.25)

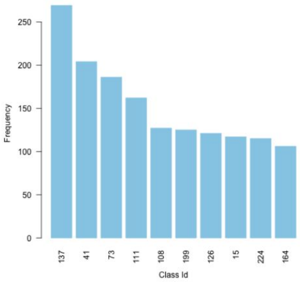
# Data Sampling and Data Splitting



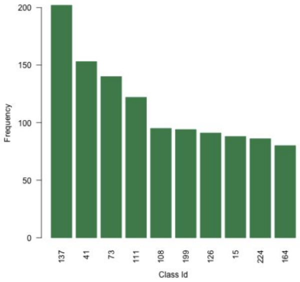

(a) Original Data



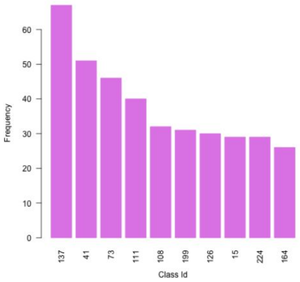

(b) 0.05% Stratified




(c) 10 Classes Stratified




(d) Training set stratified




(e) Test set stratified

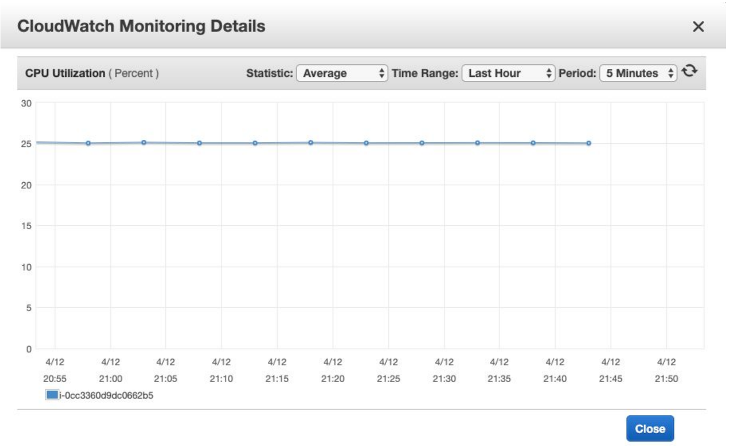
# Feature Extraction

Frequency Vector:

- BoW
- BoW after Snowball filter
- BoW after Smart filter
- n-gram 3000
- n-gram 6000

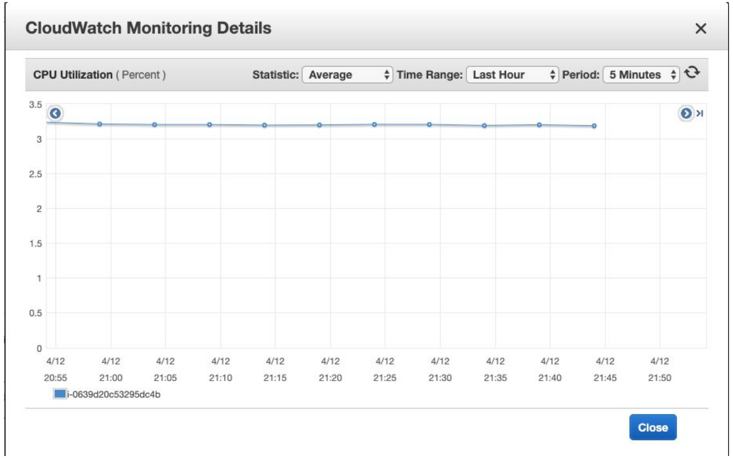
Computation-Intensive Process!!!

# Feature Extraction




AWS EC2 c4.8xlarge | 36 vCPU, 60 GiB

AWS EC2 p2.xlarge | 1 GPU, 4 vCPUs, 61 GiB



# Bag-of-Words (BoW)

- Representation of an observation in a tokenized way, where each word represents a feature(column).

```
> text_sample_BOW<-"This is an example of bag of words"
> print(text_sample_BOW)
[1] "This is an example of bag of words"
> tokenized_BOW<-txt.to.words.ext(text_sample_BOW, language = "English.all",
+                  preserve.case = FALSE)
> BOW<-txt.to.features(tokenized.text = tokenized_BOW,ngram.size = 1,features = 'w')
> print(BOW)
[1] "this"    "is"      "an"      "example" "of"      "bag"     "of"      "words"
```

# n-gram

- Representation of an observation in a tokenized way, where each sequence of ”n” words or characters represents a feature(column).

```
> text_sample_ngram<-"This is an example of n-gram"
> print(text_sample_ngram)
[1] "This is an example of n-gram"
> tokenized_ngram<-txt.to.words.ext(text_sample_ngram, language = "English.all",
+                                   preserve.case = FALSE)
> ngram<-txt.to.features(tokenized.text = tokenized_ngram,ngram.size = 3,features = 'w')
> print(ngram)
[1] "this is an"        "is an example"     "an example of"     "example of n-gram"
```

# Feature Vector

- Representation of an observation as a frequency vector where each column represent a feature an its value the respective frequency of that feature.

```
> fec_vec_training_set_bow[1:3,1:10]
             enron ect com hou   thanks subject cc   please     know      can
108_1038.txt     0   8   0   4  0.000000       2  2 0.000000 0.000000 0.000000
108_1086.txt     0   0   0   0 50.000000       0  0 0.000000 0.000000 0.000000
108_112.txt      0   0   0   0  2.272727       0  0 2.272727 2.272727 2.272727
```

# Delta Algorithm and Metrics

- Variation of k-nn algorithms with different distance.metrics.
- Algorithms that work on normalized vectors (rows).
- Also generate new distance metrics that normalize values over each feature(columns).
- Different types: Delta, Argamon, Eder, Eder Simple.

$$\Delta_{(AB)} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{A_i - \mu_i}{\sigma_i} - \frac{B_i - \mu_i}{\sigma_i}\right|$$

J. Burrows. 'Delta': a Measure of Stylistic Difference and a Guide to Likely Authorship. Literary and Linguistic Computing , 17(3):267–287, 9 2002.

# Bug Fixed

## Original

```r
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

```r
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)
}
```

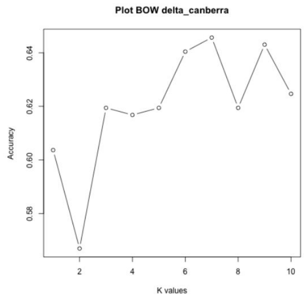
Package: stylo, Function: perform.delta(), File: perform.delta.R

# Nearest Shrunken Centroids (NSC)

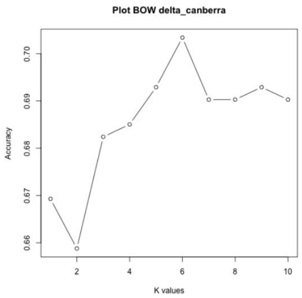
- Variation of Nearest Centroid Classifier.
- Used in cancer diagnosis with gene expressions (high-dimensional representation)
- High accuracy in that domain.
- Feature Vector could be considered a high-dimensional space.

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

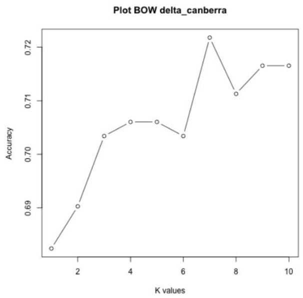
# Classification Comparison (Accuracy)



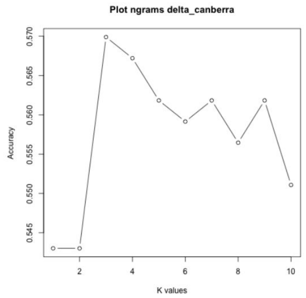

(a) w/ stop words



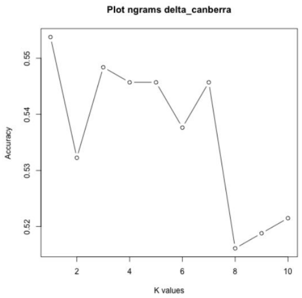

(b) w/o Snowball




(c) w/o Smart




(d) n-gram 3000




(e) n-gram 6000

# Classification Results (Accuracy)

| Algorithm | Accuracy | | | | |
|---|---|---|---|---|---|
| | BoW | | | n-gram | |
| | BoW | Snowball | Smart | 3000 | 6000 |
| Delta-based Algorithms | | | | | |
| Delta.Delta | 0.1942257 | 0.2204724 | 0.2729659 | 0.3494624 | 0.311828 |
| Delta.Eder | 0.2020997 | 0.2152231 | 0.2808399 | 0.3709677 | 0.3225806 |
| Delta.Argamon | 0.2493438 | 0.2388451 | 0.3175853 | 0.3602151 | 0.3225806 |
| Delta.Simple | 0.3569554 | 0.3175853 | 0.3700787 | 0.4301075 | 0.4112903 |
| Delta.Cosine | 0.5879265 | 0.7007874 | **0.7270341** | 0.5510753 | 0.5430108 |
| Delta.Wurzburg | 0.5931759 | 0.5748031 | 0.5800525 | 0.5376344 | 0.4892473 |
| Delta.Entropy | 0.335958 | 0.2965879 | 0.3595801 | 0.422043 | 0.3978495 |
| Delta.Euclidean* | 0.4934383 | 0.488189 | 0.5301837 | 0.4677419 | 0.4301075 |
| Delta.Manhattan* | 0.5380577 | 0.2834646 | 0.328084 | 0.4005376 | 0.3844086 |
| Delta.Canberra* | **0.6456693** | **0.7034121** | 0.7217848 | **0.5698925** | **0.5537634** |
| Non Delta-based Algorithms | | | | | |
| K-NN | 0.4908136 | 0.5013123 | 0.5459318 | 0.4892473 | 0.4301075 |
| SVM | 0.5984252 | 0.5616798 | 0.5564304 | 0.4489247 | 0.3575269 |
| NSC | **0.7454068** | **0.7506562** | **0.7427822** | **0.5349462** | **0.483871** |
| Naive-Bayes | 0.1181102 | 0.1128609 | 0.1049869 | 0.07795699 | 0.08333333 |

# Conclusions

- **NSC** with BoW performed best.
- **Delta.Canberra** with BoW is the next best.
- For BoW the removal of stopswords, improves the accuracy of the classifiers that shows high accuracy(Delta.Cosine, Delta.Canberra, NSC).
- According to theory, Delta Algorithms should perform better than the results of our experiments.
- We think that the improvement of our feature selection techniques and feature engineering could improve the current experiment results for Delta and NSC classifiers.

- Do not underestimate data preprocessing and transformation!!!

# Future Improvement Plans

- Feature Selection
- Feature Engineering
- NLP-based Features

# Questions????