Supervised and Unsupervised learning approaches for Authorship Identification

Tom Vaingart and Ayala Shaubi-Mann

Abstract

This document contains a detailed report referring to the work performed as part of final project in course "introduction to NLP", 22933, Open University, Israel.
Author identification is a research area that used for determining which author wrote a text based content while trying to make use of the data contained only inside the text itself.
In this report, we propose a supervised and unsupervised approach for selecting the most probable author, or cluster, while using a subset of the new Reuters corpus, consisting texts documents of 50 different authors and implementing technics learned as part of the course.

1 Introduction

"Any manually generated material will inevitably reflect some characteristics of the person who authored it, and these characteristics may be enough to determine whether two pieces of content were produced by the same person" (Narayanan et al.,2012).
In recent years, the attention to authorship analysis has increased in the framework of practical applications, for many purposes , such as verifying the authorship of emails, find anomalies in users messages, etc. (de Vel at el. 2001 ; Abbasi and Chen, 20015).
Authorship identification generally referring to a task of predicting the most likely author of a text given a predefined set of candidate authors and a number of text samples per author (Houvardas and Stamatatos, 2006).
One major subtask of the authorship identification problem is the extraction of the most appropriate features for representing the style of an author.
Holmes (1998), provided some concrete examples of manual author recognition in the past, where now days, we seek to be able to automatically recognize stylometry features to be feuded into machine learning models. This is a highly interdisciplinary as it takes advantage of machine learning, information retrieval, and natural language processing.

The problem becomes qualitatively more and more difficult as the number of authors increased and techniques from prior work fail. Accuracy results in related works are inversely proportional to the number of authors and the average length of a single text message.
When it comes to text-based news, and dealing with numerous number of authors, it might be more difficult to classify the texts into the different authors and the success of the task is highly dependent on the quality and subset of the chosen features.

For learning, there are two high-level approaches to use:

- Supervised learning: require author-class labels for classification. More specifically, in the training phase, the algorithm will take both author text and label as an input.
- Unsupervised learning: make classification with no prior knowledge of author classes. Usually, each text document will be represented as a point in a N-dimensional space, the algorithm will use only the text content as an input for the learning.

In our work, we will examine the use in both supervised and unsupervised learning with a focus on their performance, advantages and disadvantages.

2 Approach

We decided to investigate both supervised and unsupervised learning approaches, with a goal of successfully recognizing the key features separating the different authors.

2.1 Top challenges

- Numerus number of authors: the more authors you add, the harder it gets to classify. our dataset contains 50 different authors
- Low volume of messages per author: the less data you have, the more harder it gets to train a good model. In our dataset, we only have 50 messages per author.
- Using unsupervised method: unsupervised methods considered less accurate and more computational consuming in decoding than supervised ones. On the other hand, when the purpose is gather similar authors under the same cluster, rather than classify to correct author name, unsupervised learning is the right approach to take.

2.2 Work-Flow

Our workflow can be divide into four main phases:

1. Analyzing the data
2. Composing simple baseline models for supervised and unsupervised learning.
   Writing a good baseline will help us to get a meaningful reference point for our measurements. The results will also act as a lower bound score for our model performance.
3. Feature engineering: analyze numerous features, and their effect on author's classification and clustering.
4. Iteratively improving our models, while using the insights from previous phases.

2.3 Evaluation metrics

2.3.1 Supervised learning evaluation

We chose to use industry standard evaluation metrics.
Below metrics used for algorithm iterative learnings in addition to measuring features performance, compared to baseline model.

- *Accuracy*- a simple yet powerful measurement that quantifies the fraction of correct predictions by the model.

$$Accuracy = \frac{number\ of\ correct\ prediction}{total\ instances} \qquad (1)$$

  Accuracy is not providing the severity of the mistake because of its yes or no nature. Therefor we also used log-loss.
- *Log-loss*- quantifies the accuracy of a classifier by penalizing false classifications. Usually used where the prediction input is a probability value between 0 and 1, and it takes into account the uncertainty of the prediction based on how much it varies from the actual label.
  In order to calculate Log Loss, the classifier must assign a probability to each class rather than simply yielding the most likely class.

$$LogLoss = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} Y_{ij} log P_{ij} \qquad (2)$$

Where -
$N =$ number of instances
M = number of possible classes
$Y_{ij} =$ {1 if label j is thr right label for instance y}
$P_{ij} =$ model probability of assigning label j to instance i

2.3.2 Unsupervised learning evaluation

One of the advantages of our dataset, in contrast to other datasets used for unsupervised learning, is that we can use the labels for evaluation, which will, in turn, can give us a better and accurate results. for example, by using external methods, which requires class labels in order to measure the clustering quality.
We will use the next evaluation methods:

- *Purity*- an external evaluation criteria of cluster quality. This criterion represent the percent of data points that classified correctly. "correctly" implies that each cluster ci has identified a group of objects as the same class that the ground truth has indicated.
  Score of 1 will be obtained if we will have 1-to-1 mapping between C_i to t_i

$$Purity = \frac{1}{N}\sum_{i=1}^{K} \max_{j} |c_i \cap t_j| \qquad (3)$$

Where -
$N =$ number of instances
K = number of clusters
$C_i = cluster\ in\ C$
$t_i =$ the classification which has the *maximal count*

- *Normalized Mutual information*- considered a great method for determining the quality of the clustering. Mutual Information tells us the reduction in the entropy of class labels that we get if we know the cluster labels.
  This method allow comparing score between different clustering outputs having different number of clusters.

$$NMI(Y,C) = 2 * \frac{H(Y) - H(Y|C)}{H(Y) + H(C)} \qquad (4)$$

Where-
$Y =$ Class labels
C = Cluster labels
$H(.) = Entropy$
$H(Y|C) = Entropy\ of\ class\ labels\ within\ cluster$

2.4 The Data

Our Data set is composed out of 50 authors text documents, A subset of the Reuters RCV1 news

article dataset, obtained from the Center for Machine Learning and Intelligent Systems hosted by the University of California, Irvine.
All data came from the same text genre (Industrial news stories).
as there was an attempt to minimize the topic factor in distinguishing among the texts, we hope authorship differences will be a more significant factor in differentiating the texts.
Data set is divided into test and train sets, each has 2500 entries - 50 short text documents for each of the 50 authors.

We will use the data in two different granularities:

- ***original documents/messeges***
  The format of the messages in the dataset is exactly as in the original data, each message contains dozens of sentences.
- ***Sentence granularity***
  Applied sentence transformation, where each text message in the train and test sets divided into independent sentences.
  The motivation is real world problems related to social media, where texts are usually one sentence. In Addition, we aim to look at features related to sentence structure as a separator between authors.

2.5 Baseline

High-level linguistic characteristics used as features in our baseline model, for supervised and unsupervised models. Used features are words in sentence, characters in sentence, number of noun chunks, number of punctuation used, verb chunks, sentiment polarity (negative, positive) and average word length

2.6 Methods

2.6.1 Text pre-processing and features

We performed a few optional preprocessing on the original text, so we can later use the preprocessed text as an input for different algorithms and produce multiple features.
Traditional machine learning methods are implemented utilizing the below preprocessing and configurations.

*Part of speech: P*reprocessed version of the corpus, where Each word in the text replaced with (POS_word) pair.
The tagging of POS performed using *python NLTK* default tagger which is a Greedy Averaged perceptron tagger, pre-trained on Wall Street Journal corpus (97.1% tagging accuracy).
We strove for the features related to POS tagged text to reflect aspects of writing style that remain unchanged for some given author, regardless of the topic at hand.

Named Entities: A named entity is a "real-world object" that's assigned a name – for example, a person, a country, a product or a book title.
We added additional preprocessed version of original text in which each Named entity is replaced by its Name-tag.
Tagging performed using *python spacy* default entities tagger, which can recognize few limited predefined entities like PERSON, ORGANISATION, DATE, etc.

Character N-grams: Configuration of N "Character window". We experiment N value of 1-3.

Word N-grams: Configuration of N-gram. We experiment N value of 1-3.

On top of preprocessed text versions we used below features in multiple variations.

Bag of words: commonly used in methods of document classification where the occurrence of each word is used as a feature for training a classifier.
We used *python Kares* implementation of Bag of words (*CountVectorizer*).

TF-IDF: Numerical statistic that intended to reflect how important a word is to a document in a collection or corpus.
TF-IDF is one of the most popular term-weighting schemes today; 83% of text-based recommender systems in digital libraries use tf–idf (Beel at el., 2016).
The TF-IDF value increases proportionally to the number of times a word appears in the document and offset by the number of documents in the corpus, which contain the word.
In our experiments, terms are n-grams of characters, words, part of-speech tags or any combination of them.
TF-IDF Combines the weights of TF and IDF by multiplying them where TF gives more weight to a frequent term in an essay and IDF downscales the weight if the term occurs in many essays.

$$w_{i,e} = \left(1 + log\left(tf_{i,e}\right)\right) * \left(log\left(\frac{N}{n_i}\right)\right) \quad (5)$$

Where -
$tf_{i,e}$ = *raw count of* term i in document e
n_i = number of documents where the term i apears

$N = total\ number\ of\ documents\ in\ the\ corpus$

We used *python Kares* implementation of TF-IDF (*TfidfVectorizer*).

Basic stylometry features (Meta features): Word-count, Character count, Punctuation count, Noun phrases count, Fraction of verbs, Fraction of noun, Entities count and Polarity of text (between -1 to 1. Represent to level of positivity or negativity of text).

2.6.2 Machine learning methods

On top of the preprocessed text and obtained features, as mentioned in previous section, we tested multiple machine learning methods, utilizing the above features.
Next, we will briefly describe the algorithms and the parameter values we selected.

2.6.2.. Supervised methods

Naive Bayes - Simple "probabilistic classifiers" based on applying Bayes' theorem. It is a model that assign class labels to problem instances, represented as vectors of feature values, where the class labels drawn from some finite set.
Given a problem instance to be classified, represented by a vector $(x_1..x_n)$ representing some n features (independent variables), it assigns to this instance probabilities $P(ck \mid x_1..x_n)$ (Caruana and Niculescu-Mizil, 2006).
We used *python sklearn* implementation of Naive Bayes (*MultinomialNB*).

Multinomial Logistic regression - Linear predictor function that constructs a score from a set of weights, linearly combined with the explanatory variables (features) of a given observation using a dot product.
For the implementation of the algorithm, we used *python sklearn* implementation of Logistic regression with regularization parameter of 1.0.

Supported Vector Machine (SVM) - An SVM classifier finds a hyperplane that separates examples into two classes with maximal margin, where Multi-classes are handled by multi one-versus-rest classifiers (Cortes and Vapnik, 1995). SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. We used a non-linear kernel and a penalty parameter, C, of value 1.0, using *python sklearn* implementation of non-linear kernel SVM (*SVC*).

Gradient boosting machine - Produces a prediction model in the form of an ensemble of weak prediction models in an iterative manner.
The method is to find the best approximation to instance label by starting with a model, consisting of a constant function F0, and incrementally expands it in a greedy fashion.
In each step, pseudo-residuals, from previous iteration, are used to train a decision tree model, using Information Gain[1] to select splitting feature, by using pseudo-residuals as y, and update the model in accordance (Friedman, 2001).
We used *python XGBoost* library, to implement a GBM classification with log-loss loss function.

Convolutional Neural network (CNN) - The primary role of the neural models is to represent the variable-length text as a fixed-length vector. These models generally consist of a projection layer that maps words, sub word units or n-grams to vector representations, and then combine them with the different architectures of neural networks.
Neural networks models take as input the embedding's of words in the text sequence, and summarize its meaning with a fixed length vectorial representation (Liu at el., 2016).
In CNN, each layer learn an abstract combination of the previous layer, using a convolution function. In this case, we will have word embedding in the input layer, multiple sequential convolution filters in the hidden layers, which eventually encode the classification class (see Appendix A-Machine learning methods).
We used *python Kares Sequential* library, to implement a Sequence classification with 1D convolution, which applies 64 convolution filters of size 5 each.

2.6.2.. Un-supervised methods

K-means - A well-known method of vector quantization, popular for cluster analysis.
K-means clustering aims to partition N observations into k clusters in which each observation belongs to the cluster with the nearest mean
We used *python sklearn* implementation with up to 50 clusters and Euclidian distance function.

[1] information gain IG(Fi) = H(B)−H(B|Fi), where H denotes entropy, B is the random variable corresponding to the blog number, and Fi is the random variable corresponding to feature i

Principal component analysis (PCA) - PCA is a method for performing a transformation of the data that maximizes the variance in the new axes. It projects high dimensional data into a low dimensional sub-space.
In general, handling high dimensional data using clustering techniques obviously a difficult task in terms of higher number of variables involved.
In order to improve the efficiency, the noisy and outlier data may be removed (Napoleon and Pavalakodi, 2011).
We used *python sklearn TruncatedSVD* library with different number of components.

3 Experiments

3.1 data analysis

In this section, we analyzed the data in the granularity of sentences where we had 38,000 sentences in training corpus (see Appendix B- Data analysis for author distribution).
Data set distribution among the different authors is such that the most common author is Lynne O'Donnell with about 1K sentences.
As such, a constant classifier predict the most common author 100% of the time will achieve accuracy of ~2.5%.

To get a sense on the differences between the authors, we looked at basic stylometry features as common words used, number of words in sentence, entities types the author mentioned, punctuations used, POS tags relations and text sentiment polarity (see Appendix B- Data analysis)

3.2 Supervised learning

To construct described supervised learning model, we used data in the granularity of sentences.

3.2.1 Baseline

To have a "smarter" baseline than majority label classifier we tested multiple structured classification methods on top of the list of meta features only. This method, using logistic-regression model, raised accuracy level of 5.5%, and log-loss of 3.8.
As the low accuracy is expected, we used this as the minimal performance levels we can reach.

3.2.2 Bag of words

A more sophisticated model than the one achieved in the baseline use "Bag of words" features.
Greatly outperforms the baseline method, even with the simplest implementation using Naive-Bayes classifier, without N-gram, on the original sentences, which achieved log-loss value of 1.65 and accuracy of 0.69.

model	Log-loss	accu-racy
Bag of words, NB, text to sentences, 1-gram	1.64	0.69
Bag of words, NB, text to sentences, 3-gram	2.84	0.73
Bag of words, NB, lemmatized sentences, 3-gram	1.913	0.75
TF-IDF, NB, text to sentences, 3-gram	2.15	0.72
TF-IDF, Logistic-regression, text to sentences, 3-gram	1.94	0.73
TF-IDF, Logistic-regression, POS-tagged sentences, 3-gram	2.24	0.72
TF-IDF, SVM, text to sentence, 3-gram	1.533	0.57
CNN, text to sentences	1.54	0.65
CNN, POS-tagged sentences	1.56	0.65
GBM- stacked features	0.761	0.795

Table 1: supervised models performance

As seen in the confusion matrix, presented in Figure 1, there are authors where very high percent of their sentences correctly classified vs others with relatively low percent of correct classification.
Manual error inspection of common confusions shows that authors having high error rate (some with 0 correct classifications) are with high variance of different topics coverage.
For example, author John Mastrini, write about cars, politics, and economics in different documents therefore using multiple different names and terms in different sentences and documents.
In fact, the names used in it's sentences (as different country names) has high impact on sentence classification to the writer constantly relates to this names.

Adding higher word N-grams configuration, together with punctuation and stop-words removal, we farther improving accuracy to 0.75.

3.2.3 TF-IDF

Using TF-IDF features, with various ml models (describe in section 2.6.2), achieved max accuracy of 0.73, when configuring N-gram value of 3, using Logistic regression model on original text as sentences.
Minimal log-loss value of 1.53 achieved using SVM basic model, with 3-gram, but with a much lower accuracy level of 0.57.

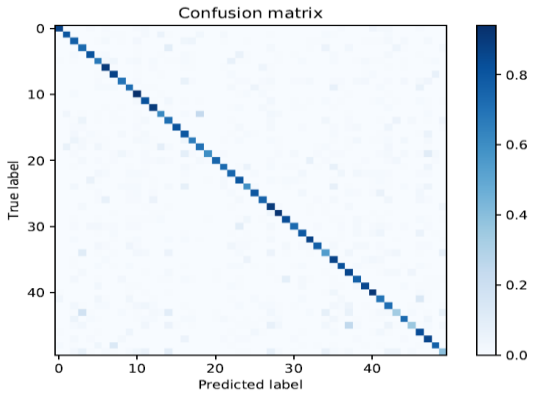


Figure 1: confusion matrix for bag-of-words with stop words and punctuation removal, using N=3 word grams

3.2.4 CNN

We used CNN on top of author's sentences using *Adam* optimizer[2].
As we used configuration of 64 convolution filters of size 5 each, no need in preprocessing of N-grams.
Although received accuracies are not the maximal among the different experiments, CNN's models achieved low log-loss values (~1.5) and as such, has high importance in the ensemble GBM model.

3.2.5 Best model & conclusions

To combine all strong features into a single model that will leverage each one of them,
We operated in accordance to the schema describe in Figure 2

1. tested multiple combinations of preprocessed text, word n-gram settings and character n-gram settings
2. Use as an input to text vectorization
3. Use as input to supervised classification model (e.g. Multi-nominal NB)
4. "Stack" probability features to GBM model

In this way, on top of the above metadata and preprocessed text, we used classification algorithms describe in section 2.6.2 to conduct probability features.

In Table 1, we can see the accuracy and log-loss improvements over the different experiments.
The most meaningful features, according to their F-score (number of times a feature is used to split the data across all GBM trees), are features obtained by CNN model, on top of POS tagged sentences and Entity tagged sentences (see Appendix C- Supervised learning for top 50 features F-scores).

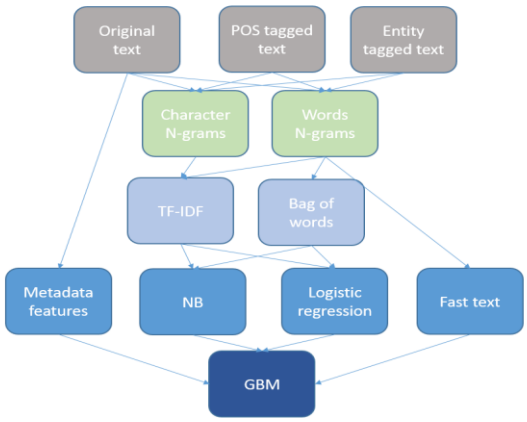


Figure 2: Stacked GBM model schema

Using this technique, we achieve an accuracy of 0.795, with log-loss of 0.76.

3.3 Un-supervised learning

We chose to implement clustering method for grouping data objects into disjointed groups; in our case, these objects represent text messages.

3.3.1 Distance function

We used 'Euclidean distance', which represents the length of a line segment connecting two points and is the most commonly used distance function, especially if all coordinates are normalized.

3.3.2 Baseline

A dump model, which always predict the same cluster for all documents (no majority, all authors have equal number of documents) would get the following purity and NMI scores –

$$Purity = \frac{1}{2500}\sum_{i=1}^{1}\max(50) = \frac{50}{2500} = 0.02 \quad (5)$$

$$MC(C,Y) = 0 \ (independant\ variables) \quad (6)$$

To have a "smarter" baseline, we used a list of meta features mentioned in section 2.6.1.
the set of features that participated in our baseline were punctuation count, text polarity, entities count, nouns and adjective counts (all normalized) which provided purity score of 0.137, almost 7 times better than the dumb model, and Normalized mutual score of 0.227.

[2] stochastic gradient descent optimization algorithm that is used to update network weights iterative based on training data

3.3.3 Sentence transformation experiment

As mentioned before, we wanted to test clustering performance on sentences (where original, medium messages length, converted into a sentence-based dataset).
Performing multiple tests, trying to find the best sentence clustering, using the specified set of features, best reached performance was when using different combination of words and character N-grams, achieving purity score of 0.11 and Normalized mutual score of 0.139.
When comparing the score to clustering using original documents, sentences transformation provides underperformed results.
We suggest two explanations for this outcome.

First, single sentence contains less data comparing to multiple sentences together, as part of a message, which makes person attribution harder.
Looking at cluster results for messages and corresponding sentences, we can see that cluster distribution is with much higher variance for sentences use case.
it can suggest that referring to an entire document and using the order of sentences within holds additional information relevant for clustering which the representation of a single sentence is missing.
An example from *Aaron Pressman's* 10 first messages (documents), which corresponds to ~100 single sentences presented in Table 2 and Table 3, (see Appendix D- Unsupervised learning)

Second, when clustering according to sentences; we are removing the inner dependencies of sentence, inside the same message. One example could be that one author is using a unique term in each and every document, but not necessary this term will show up in every sentence of the message.
Aaron's "120600newsML" message, for example, show that for top 10 sentences in the above message (see Table 4, Appendix D- Unsupervised learning) cluster groups of first 10 sentences in *120600newsML* message, *Aaron Pressman* majority, 7/10 sentences, grouped into only two different clusters.
It is probably right to assume that if we analyze the clusters, for the whole message, these 7 sentences would tip the scales regarding the whole message cluster.
Intuition for this scenario can give us a chance of at least 50% to be right for the whole message, which is better than 40% that we had in the sentence scenario.
to conclude, the full message scenario can help us neglect the minority sentences, which is probably a good thing for most of the models, with a big bonus which is generating a more generative and accurate modules as a result.

3.3.4 Best method & conclusions

Refer to clustering generated using messages granularity.
After many experiments with different sets of features, Figure 3 presents the model that gave us the best score.

1. preprocessing phase for cleaning and lemmatizing the text
2. word n-gram settings and character n-gram settings together with TF-IDF
3. multiple dimensionality reduction phases

Using the described flow achieved average purity score of 0.535 and NMI score of 0.7, which is more than 3.5 and 3 times better than the baseline purity and NMI scores and more than 26 times better than the dumb model.

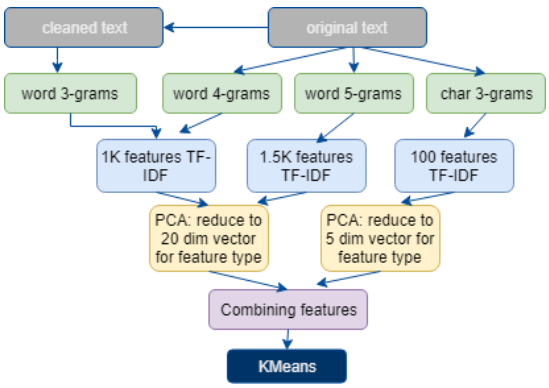


Figure 3: Unsupervised best method schema

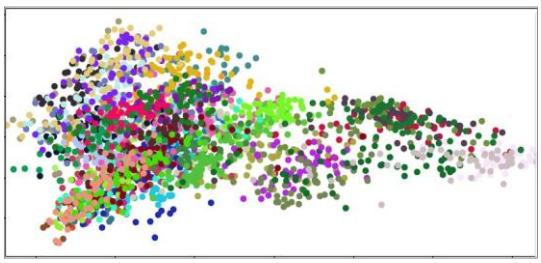


Figure 4: 2D (using PCA) visualization of 50 clustering results

Presented in Figure 4 clustering results after reducing output into 2-dimensional space.

To review the groups of similar authors, under different granularity, we produced a Dandogram[3] using agglomerative clustering [4] algorithm, which presents the split of authors to different cluster numbers between 1 to 50 (Figure 5).

Although lower cluster numbers achieved lower scores, when it comes to selected evaluation metrics, 2D visualization of the clusters showed good separation between created groups (see Appendix D- Unsupervised learning).

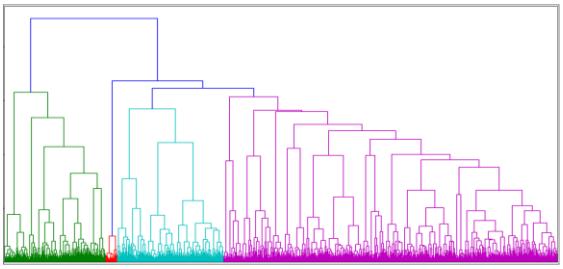


Figure 5: Hierarchical Agglomerative clustering Dendrogram

4 Conclusion and further work

During the work on the project, we have investigated different aspects of authorship classification problem. This is by using multiple NLP libraries and conducting numerus number of experiments for ideas raised in the course to confer with a real business case problem.
We used an extended set of features such as structural characteristics and linguistic patterns.

Our experiment showed that although features constructed from POS tagged sentences are with the highest information gain for supervised classification models, they have lower strength, compared to sentences dependencies in entire document, when it comes to clustering, where we do not have a training phase. We also learnt that the relations between sentences inside a document has a major effect, when we tried split text to independent sentences, a transformation which got less accurate results compared to original format. This result could be changed, when dealing with different corpus nature, where sentences usually linked together and not spontaneously written. In addition, as author classification can be a very hard task, supervised learning models can be improved using stacked featured model, as we saw in our best model, which improved baseline accuracy by 10% and baseline log-loss in more than 2 points. This is while most reviewed articles inspect the use of a single method for classification.

Inspection of supervised model errors shows that some authors are harder to separate, probably due to the fact that they have relative similar style to others.
this point was also emphasized by our clustering model, in which, we saw that some authors (regardless of the chosen features) indeed splitted into multiple different clusters, dominated by other author, where others, more consolidated in their style and covered topics, gathered in a single cluster.

Our work holds few important limitations.
First, while we have validated our models with validation set, we have not tested it in a cross-settings and context, e.g., labeled text originated in news posts, whereas anonymous text can be an e-mail or social media post.
Second, although we tested our ability to classify relatively high amount of authors, our method might not meet the requirements of anonymous posts applications where the number of authors might be very large (thousands of authors).
Third, computational efforts of using both supervised and unsupervised methods for text classification and clustering are very high, sometimes not adjusted to average PC. Upon corpus expansion, execution times need to be calculated and hardware adjustments are required in accordance.

We point out three main avenues for future research.
First, testing achieved models and methods on different domains texts, and fully understand the successes and failures of our models on this domains. Understanding and modeling how writing style is affected by context has the potential to bring major gains to accuracy
Second, testing deep syntactic features addition to our models, as syntactic tries, which we did not use in our project as to hardware limitations.
Third, test and possibly extend our model to work with other aspects of group identity[5] (except for author), such as sex, education level, etc.

[3] a tree diagram frequently used to illustrate the arrangement of the clusters produced by hierarchical clustering
[4] a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy

[5] Group identity refers to a person's sense of belonging to a particular group. At its core, the concept describes social influence within a group.

References

[1] Abbasi, A., Chen, H.: *Applying Authorship Analysis to Extremist-Group Web Forum Messages*. IEEE Intelligent Systems, 20:5 (2005) 67-75.

[2] Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2016*). paper recommender systems: a literature survey*. International Journal on Digital Libraries, 17(4), 305-338.

[3] Caruana, R., & Niculescu-Mizil, A. (2006, June). *An empirical comparison of supervised learning algorithms*. In Proceedings of the 23rd international conference on Machine learning (pp. 161-168). ACM.

[4] Corinna Cortes and Vladimir Vapnik. 1995. *Supportvector networks*. Machine learning, 20(3):273–297.

[5] de Vel, O., Anderson, A., Corney, M., Mohay, G.: *Mining E-mail Content for Author Identification Forensics*. SIGMOD Record, 30:4 (2001) 55-64.

[6] D.Napoleon, S.Pavalakodi. (2011). "*A New Method for Dimensionality Reduction using KMeans Clustering Algorithm for High Dimensional*" http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.206.3001&rep=rep1&type=pdf.

[7] Friedman, J. H. (2001). *Greedy function approximation: a gradient boosting machine*. Annals of statistics, 1189-1232

[8] Gebre, B. G., Zampieri, M., Wittenburg, P., & Heskes, T. (2013). *Improving native language identification with tf-idf weighting*. In the 8th NAACL Workshop on Innovative Use of NLP for Building Educational Applications (BEA8) (pp. 216-223)

[9] Houvardas, J., & Stamatatos, E. (2006, September). *N-gram feature selection for authorship identification*. In International Conference on Artificial Intelligence: Methodology, Systems, and Applications (pp. 77-86). Springer, Berlin, Heidelberg

[10] Holmes, D.: *The Evolution of Stylometry in Humanities Scholarship*. Literary and Linguistic Computing, 13:3 (1998) 111-117.

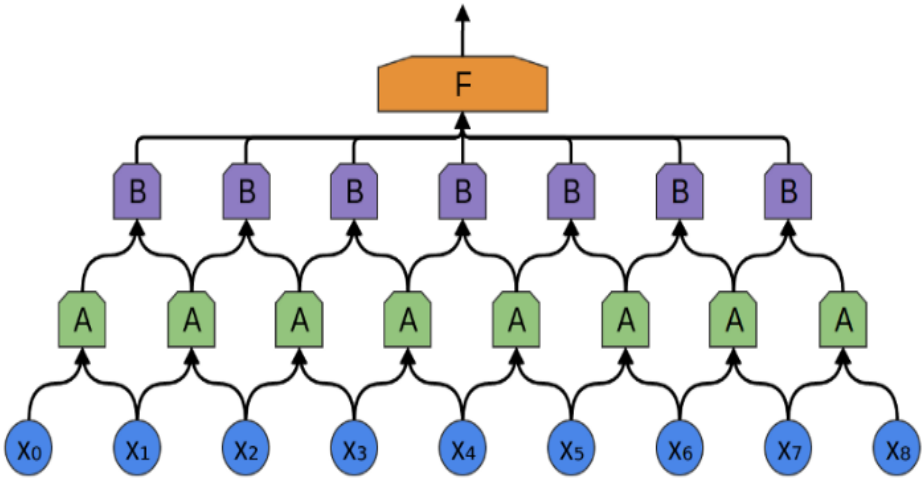
[11] Liu, P., Qiu, X., & Huang, X. (2016). *Recurrent neural network for text classification with multi-task learning.* arXiv preprint arXiv:1605.05101

[12] M. Koppel, S. Argamon and A. Shimoni (2002). *Automatically categorizing written texts by author gender, Literary and Linguistic Computing, vol.* 17(4), pp. 401–412.

[13] Narayanan, Arvind, et al. *On the feasibility of internet-scale author identification*. Security and Privacy (SP), 2012 IEEE Symposium on. IEEE, 2012.

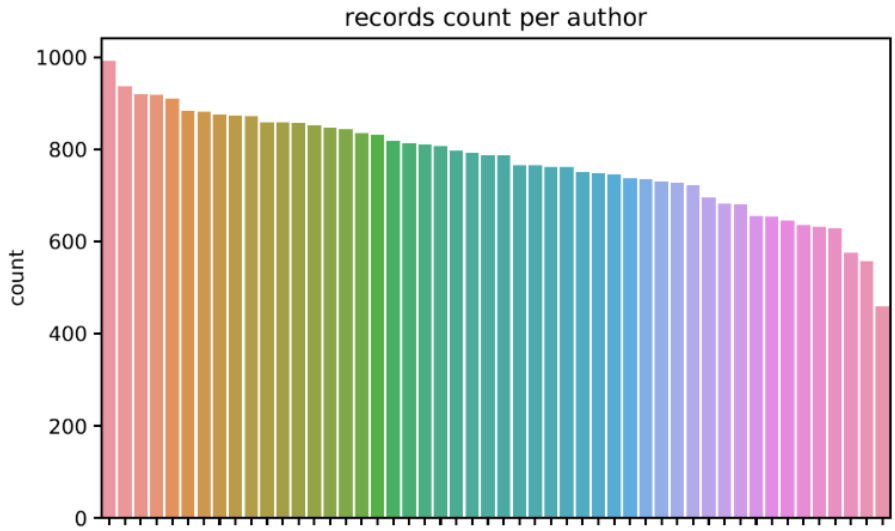
[14] Wang, L. Z..*News authorship identification with deep learning* (2017).

5 Appendix A- Machine learning methods



CNN. Each “A” sees two inputs. Each “B” sees two “As” which have a common input so each “B” has a receptive field of 3. Each B is exposed to exactly 3 inputs.

6 Appendix B- Data analysis



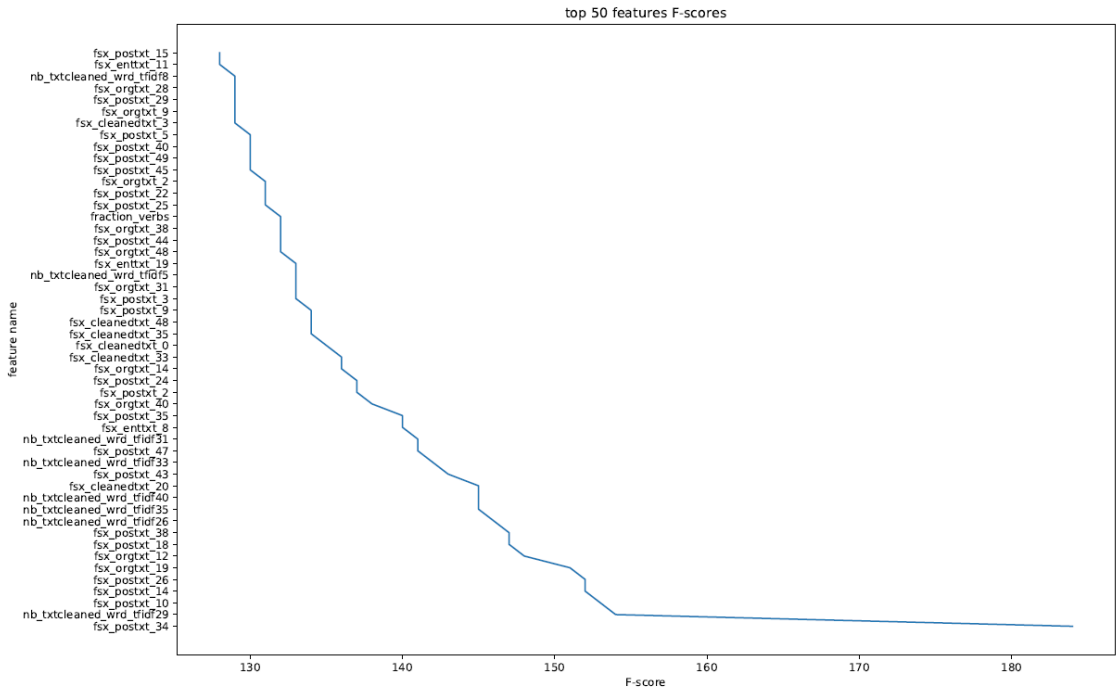
Instances per author in train set

Data piece	average	Conclusion
Sentence words count	between 0 to 50 words	some authors are characterized with long right a-symmetric distribution of sentence length
Punctuation use	0-4	up to encountered sentence with 20 signs
Text polarity	mostly neu-tral sentences	Balance in the use of negative and positive sen-tences such that sentiment is not a factor in the clas-sification.

Use of noun phrases	5-10 NN's	few exceptions, as Joe Ortiz, which has on average, less than 5 NN's.
Top used words		there are significant differences between the authors, which provides indication to their area of coverage.
Top used entities		One author my talk about Location entities the more where others on persons.

stylometry features inspection

7 Appendix C- Supervised learning



Features F-score for GBM final model, supervised learning

8 Appendix D- Unsupervised learning

Cluster number	Number of messages in cluster
33	8
6	2

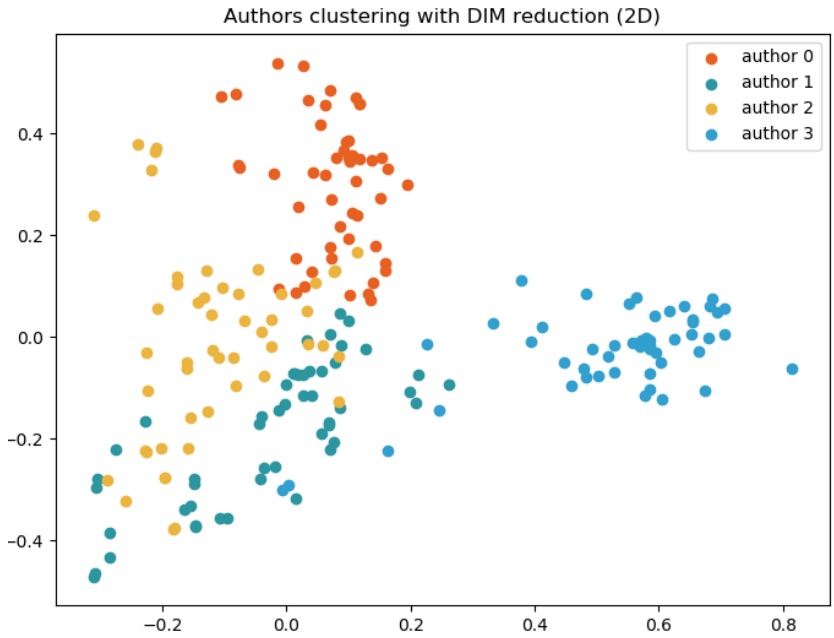
Table 4: first 10 documents cluster distrbution, Aaron Pressman

Cluster number	Number of messages in cluster
20	4
28	3
15, 13, 48	1 each

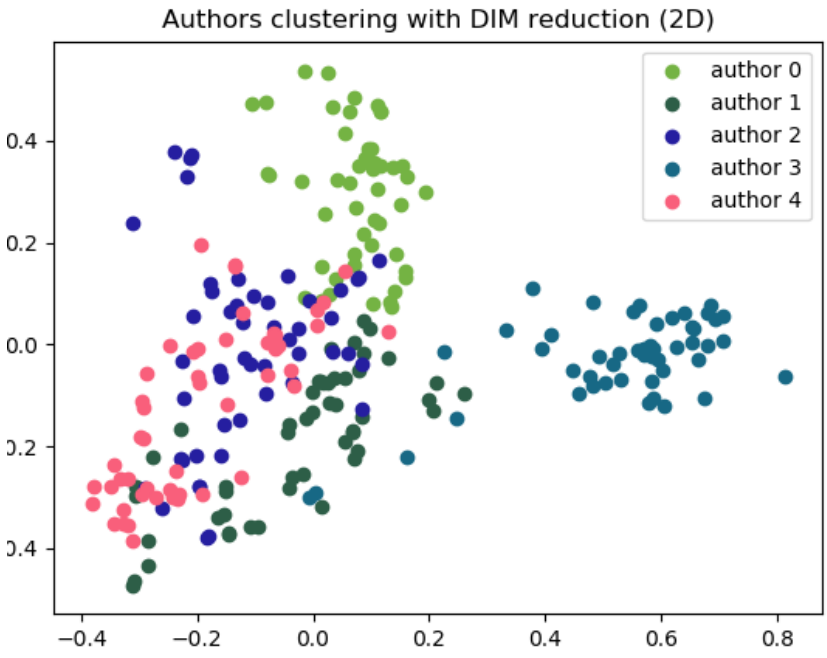
Table 3: cluster groups of first 10 sentences in 120600newsML message, Aaron Pressman

Cluster number	Number of sentences in cluster
15	22
28	14
1	12
20	10
13	9
0	7
36	4
38, 45	3 each
31, 47, 40, 14, 2	2 each

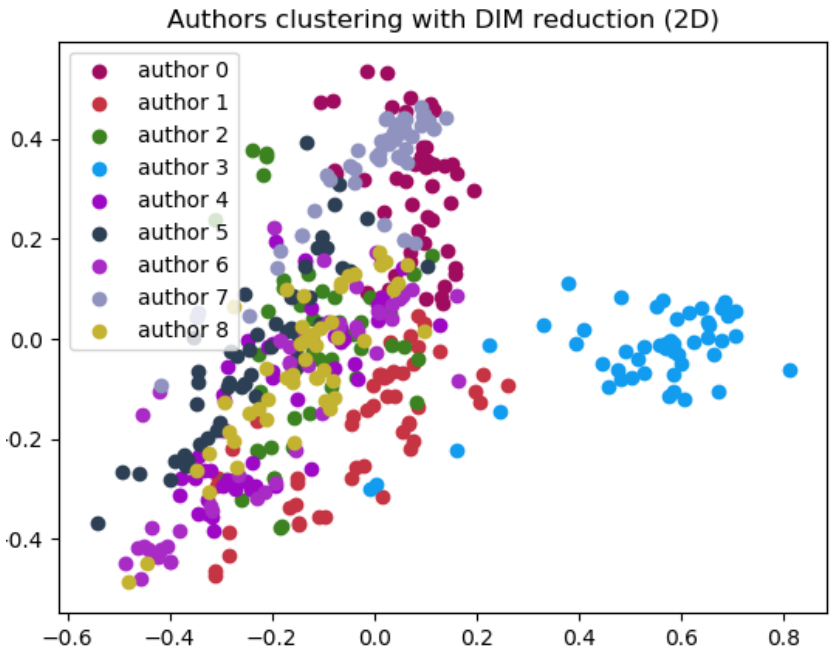
Table 2: first 10 messages, sentence distribution to clusters, Aaron Pressman



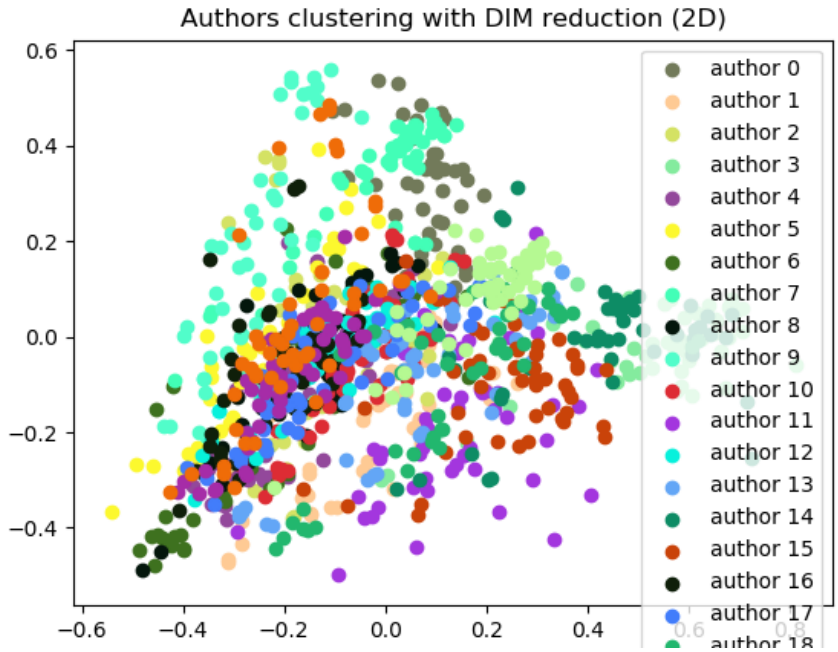
2D (using PCA) visualization of 4 clustering results



2D (using PCA) visualization of 5 clustering results



2D (using PCA) visualization of 9 clustering results



2D (using PCA) visualization of 22 clustering results