Authorship Identification using short text in Spooky Author Dataset

Krishna Teja Chintanippu
MSBA : A53282442

Adarsh Mehra
MSBA : A53274261

Rucheng Jin
MSBA:A53288280

Abstract

Authorship identification is one of the most important topics in the field of Natural Language Processing (NLP). It helps us identify who most likely the author of an article, news or text is. Authorship identification can be applied to tasks such as identifying the original author of a widely reprinted anonymous text/quote, predict authorship of texts in dispute, spam identification and detecting plagiarism in articles/plagiarized paragraphs. In this project, the goal was to predict the possible author of a spooky story between three authors Edgar Allan Poe, HP Lovecraft and Mary Wollstonecraft Shelley. In simple words it's a 3-way classification problem. we tackled this problem at different levels, with different supervised Machine learning models and on a dataset made available on Kaggle. Among all models we tested, simple Multinomial Naïve Bayes classification on SVD word features of the data achieve the best result of .45 multi class log loss. For further improvement in accuracy of this type of NLP problem, we used simple feature engineering with word and character TF-IDF vectors(reducing dimensions through SVD), and stacking features by applying multinomial Naïve Bayes, TF-IDF, words vectors and using XGBoost modeling with these simple and stacked features gave us an improvement in accuracy of 0.33 multi class log loss error. Recent studies of authorship identification are primarily based on machine learning approaches such as deep neural networks, with utilization of various word vector representations.

Keywords

Logistic Regression, Naïve Bayes Classification ,Support Vector Machines, Hyper-Parameter Tuning-Grid search, Count-features/Bag of Words, Singular Value Decomposition/SVD, TF-IDF, Log-loss Error

1 Background

1.1 Introduction

Words gave us the power to convey our thoughts and understand others. We received messages and read text almost everyday. It's highly possible for us to tell who's the author of a written text if the author is who we are familiar with. However, when determine unknown author of text, it's hard to tell just by intuition. Meanwhile, authorship identification is important in our life. For example, when people received anonymous threatening letters, authorship identification could help us to predict who's the most possible author from a list of suspects.

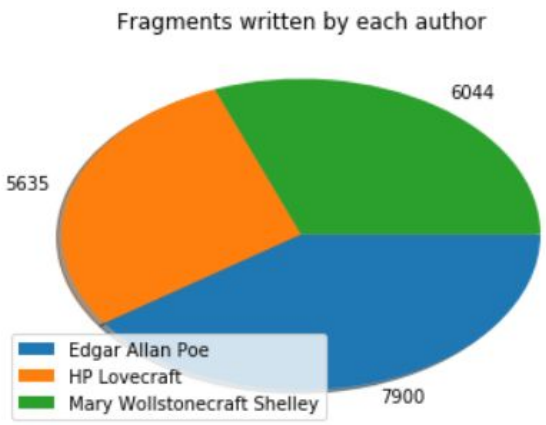
1.2 Exploratory Data Analysis

The dataset comes from a Kaggle competition "Spooky Author Identification", which began in October in consonance to Halloween theme. The dataset contains different fragments of horror fiction from three authors. Training data itself contains 20000 fragments. Each author has different writing habits. Thus, by training a classifier, we could predict who are the author based on fragments.

The data contains three columns:
1 id: the unique identifier for each text sentence;
2 text : each text fragment written by a author
3 author: the author of the fragment

	id	text	author
0	id26305	This process, however, afforded me no means of...	EAP
1	id17569	It never once occurred to me that the fumbling...	HPL
2	id11008	In his left hand was a gold snuff box, from wh...	EAP
3	id27763	How lovely is spring As we looked from Windsor...	MWS
4	id12958	Finding nothing else, not even gold, the Super...	HPL

The distribution of text written by each author:



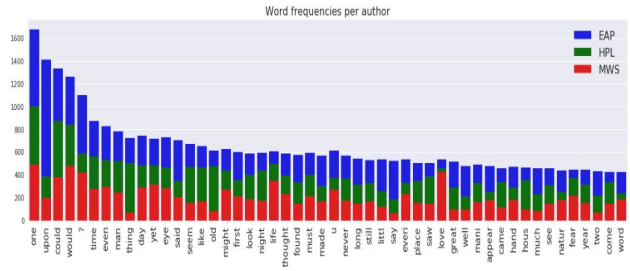
As we could see from the pie chart, although data is not evenly distributed, the difference is not that huge. For each author, there would be enough data to train a classifier.

The word cloud of text written by EAP:

The word cloud of text written by HPL:

The word cloud of text written by MWS:

From the word cloud of each author, we could have a basic understanding of what are the words authors preferred to use.



It's good to see that this plot suggests that our three writers prefer some words over others. Take for example word 'thing', which isn't used much by MWS compared to EAP and HPL. On the other hand, she clearly uses the word 'love' more than the other two writers.

1.3 Possible Approaches Analysis

TF-IDF

TF-IDF is the abbreviation of term frequency & document frequency. Term frequency, is the simplest count of times the term appears in a document.

$$tf(term, document) = \frac{number\ of\ times\ term\ appeared}{total\ terms\ in\ the\ document}$$

Inverse document frequency, on the other hand, is the logarithmically scaled inverse fraction of documents containing the term.

$$idf(term, document) = \frac{total\ number\ of\ ducoments\ in\ the\ corpus}{number\ of\ documents\ where\ term\ appears}$$

TF-IDF is equal to the product of tf and idf. Higher TF-IDF represents that the term appears more often in this document compared to others.

TF-IDF is efficient and easy to implement. However, using TF-IDF alone would fail to distinguish singular word and plural word. When implementing TF-IDF, it works better when combined with other approaches.

Naive Bayes

Naive Bayes is one on the supervised machine learning classifier technique. It is probably the simplest, fastest way to implement.It assumes all features are conditionally independent and all

samples are independent and identically distributed.

$p(\text{a} \mid \text{b}) = \frac{p(a) * p(b \mid a)}{p(b)}$

In the equation, p(a | b) is posterior probability, our hypothesis before seeing evidence; p(a) is prior probability; p(b | a) is the probability of seeing evidence given that hypothesis is true; p(b) is the probability of evidence happening under any possible circumstance.

Although the algorithm of Naive Bayes is simple and the assumption could never be true, Naive Bayes has other benefits. It does not need a lot training data and at the same time works well with scaled data.

Logistic Regression

Logistic Regression is the expanded version of linear regression. In linear regression, we predict the exact value. The equation is : $y = X_i \bullet \theta$. Logistic regression, on the other hand, converted exact value into a probability. y is equal to 0 when $X_i \bullet \theta > 0$. Otherwise, y is equal to 0.

Logistic regression allows us to use multiple explanatory variable to predict outcome.

Logistic regression requires a lot of data to train in order to have stable result. When number of features is huge, it is better to use logistic regression. If data is not large enough, SVM is a better predictor.

SVM

SVM is the abbreviation of support vector machine. It maximizes the margin between 2 closest separate classes.

SVM model could monitor non-linear decision boundaries, it helps prevent overfitting, especially in high-dimensional area.

Meanwhile, it took time for SVM to process when dealing with large datasets.Also, it is sensitive to irrelevant features.

2. Predictive Task Identification and Feature Engineering

2.1 Predictive Task Identification

This is a 3-way classification problem, the data set contains the text from works of fiction (Horror) written by the authors

1. Edgar Allan Poe (EAP)
2. HP Lovecraft and Mary (HPL)
3. Wollstonecraft Shelley (MWS)

The task was to accurately predict the author of the sentences of text in the Test Data set.

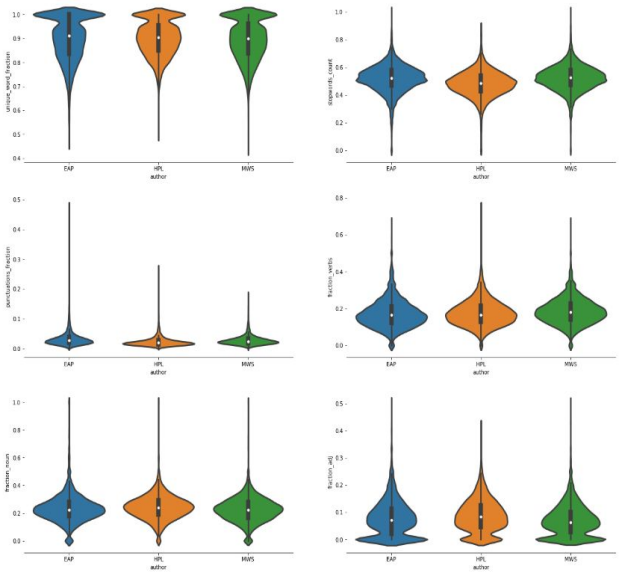
2.2 Feature Engineering

Because we are trying to segregate authors of the same type of literature, just working on the sentences and words used won't help as the genre is same. So we decided to explore some embedded features of the different sentences in the books. This will help analyse the style of writing for the different authors. Based on this thought we created some of the derived features from the text given.

- word_count - The count of words in given text
- unique_word_frac – Fraction of unique words in a given text
- stop_words - Fraction of stopwords present in a given text - Number of stopwords/Total words
- punctuations - Fraction of punctuation present in a given text - Number of punctuations/Total words
- fract_Nouns - Fraction of Nouns
- fract_adj - Fraction of Adjectives present in a text

Apart from this we thought, performing a sentiment analysis on the text seemed like a reasonable option as it helped us understand the

kind of words different authors use which again corresponds to our agenda of differentiate on the writing styles of the authors. For that we used the NLTK sentiment module,
"nltk.sentiment.vader module"[1]



This helped us get a compound score on the sentiment of each line in the book from different authors. Visualizing the above derived features to check if there is actually some help using these.
Using, the above we can observe that stop words are more frequently used by HPL .
For punctuations ,we can clearly see that any fraction above 0.3 is definitely the author EAP, which can be used for flat predictions.
In terms of Unique words, HPL uses a larger fraction of Unique words. Same goes for Nouns, verbs and adjectives.
Although there is not much difference in visualizations it will definitely help us classify some texts.

3. Model Selection

3.1 Features

We have considered two major types of features for this modeling,

1) Which were extracted from the text like word count, stop words count, punctuation count, etc.as described in feature engineering section
2) which are directly based on text/words like frequency, TF-IDF, SVD, word2vec etc.

3.2 Model Exploration

Here, we elaborate on the supervised machine learning methods we have used in our analysis. In total 4 different models with different combinations of features were implemented: Simple logistic regression, Naïve Bayes Classification, Support Vector Machine, and XGBoost (tree boosting).

(1) Logistic Regression on TF-IDF and word count vectors: The model built on logistic regression was built as the basic model. Since it was a simple, we used it as a first classifier to try. It takes weighted input features and uses sigmoid function , which maps a real number to a probability between 0 and 1. Positive class is predicted if output is greater than 0.5 , negative otherwise. Since this is a multiclass classification problem, pairwise classification is implemented. This method performs decently but studying related works we found out that Logistic Regression and SVM usually perform well on long documents, but here since we are classifying short ones it is not very ideal.

(2)Naïve Bayes Classification: we used a simple probabilistic classifier, in this specific case the classifier, for a given sentence/ excerpts, the model should return, out of all possible classes author i.e. Author (EAP, HPL, MWS), the estimated author with maximum posterior probability P(author/sentence). We have tried using both TF-IDF word vectors and just simple Word counts. However we implemented hyperparameter (alpha/Laplace smoothing) tuning here, the reasons for this are discussed in the results section.

(3) Support Vector Machine: We have tried using pairwise Support vector machines classification. A model with representation of each example as a point in space, mapped such that separate classes are divided by a clear gap, i.e. a maximal Margin Hyperplane that is as wide as possible. This hyperplane is based on minimizing the generalization error while minimizing the training error, i.e. regularization. We used this because in previous studies of authorship identification, SVM seem to have been successfully implemented.

(4)Naïve Bayes Classification with optimization/Hyperparameter tuning of Alpha (Laplace/Lid stone correction) parameter: We have performed a simple grid search over different settings of alpha, and cross validated this to evaluate the performance in each setting on the training data set. We applied this both on TF-IDF features and simple word counts.

3.3 Selecting the best modelling approach

Now we have couple of choices after using the above basic models.

1) We can choose to get the top 1000 features from the TfIdf or Countvectorizer, and merge it with the features we identified in the feature engineering

2) Using the predictions from the basic models we did and then use the predictions as one of the features along with other features we found in the feature engineering before.

After testing Naive Bayes Classifier using TF-IDF Vectors and Count Vectors we noticed that Naive Bayes performs quite good independently itself in case of using TF-IDF as the predictor with hyper parameter tuning(Log Loss of 0.45)

Also, we got similar results when we used Character TfIdf vectoriser with SVD, wherein we created a character level TfIdf vectorizer, compacted it using SVD into 150 metrics and then used Naive Bayes on this. We got a log loss of 0.45 for this model as well.

So, we decided to go ahead with the second approach of using the prediction metrics from the Naive Bayes models for both words and characters + the top 150 SVD features for the character and word vectorizer as an input to the Gradient Boosted Decision Trees formally known as XGBoost.

4.References and Current Research

4.1 Existing Literature and Previous Research[2]:

Most research on author identification considers large texts. Not many research is done on author identification for short texts, while short texts are commonly used since the rise of digital media. The anonymous nature of internet applications offers possibilities to use the internet for illegitimate purposes. In these cases, it can be very useful to be able to predict who the author of a message is. Van der Knaap and Grootjen [28] showed that authors of short texts can be identified using single words (word unigrams) with Formal Concept Analysis.

In author identification research different aspects can influence the performance of the author classification task. These aspects are the language of the messages used, the length of these messages, the number of authors and messages, the types of features and the classification method. Before executing experiments the researchers make choices on these aspects.

The previous research discussed all have one thing in common: they all use less than 10

authors. Zheng et al. [30] executed an experiment that revealed that performance increases when the number of authors decreases.

We also studied different projects on Kaggle based on Text Mining and Classification which helped us understand how, when and which models to apply to have an effective text based multi class classifier.

4.2 Dataset Used

The dataset we used was part of the kaggle competition.https://www.kaggle.com/c/spooky-author-identification

We have used the train and test data provided and performed our analysis on the train set with a 80:20 split for training and validation.

4.3 Best Practices

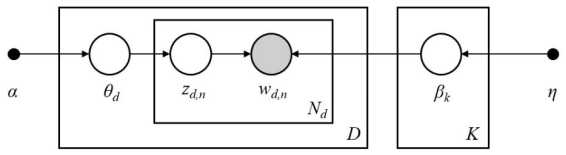
Best practises for Natural Language Processing in terms of short text analysis includes algorithms that employ Unsupervised learning.

Topic Modelling

Latent Dirichlet Allocation

Corpus - Document - Word : Topic Generation

In LDA, the modelling process revolves around three things: the text corpus, its collection of documents, D and the words W in the documents. Therefore the algorithm attempts to uncover K topics from this corpus via the following way (illustrated by the diagram)



Model each topic,κ via a Dirichlet prior distribution given by βk. Model each document d by another Dirichlet distribution parameterized by α. Subsequently for document d, we generate a topic via a multinomial distribution which we then backtrack and use to generate the correspondings words related to that topic via another multinomial distribution

The LDA algorithm first models documents via a mixture model of topics. From these topics, words are then assigned weights based on the probability distribution of these topics. It is this probabilistic assignment over words that allow a user of LDA to say how likely a particular word falls into a topic. Subsequently from the collection of words assigned to a particular topic, are we thus able to gain an insight as to what that topic may actually represent from a lexical point of view.

From a standard LDA model, there are really a few key parameters that we have to keep in mind and consider programmatically tuning before we invoke the model:

n_components: The number of topics that you specify to the model

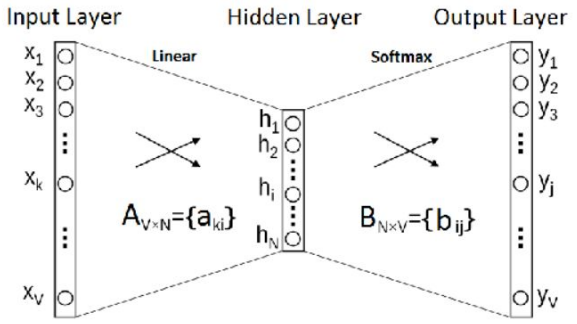
α parameter: This is the dirichlet parameter that can be linked to the document topic prior

β parameter: This is the dirichlet parameter linked to the topic word prior

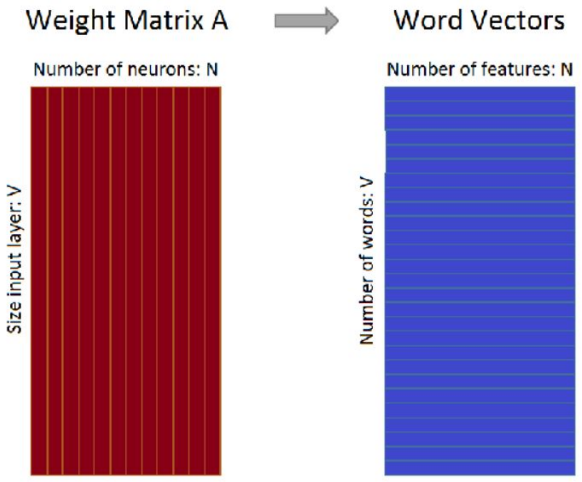
Deep Learning

Word2Vec

Word2vec is a deep learning method for creating vector representations of words. There are basically two word2vec approaches: continuous skip-gram modeling (SG) or continuous bag of words (CBOW). Both create word vector representations, but with different approaches. The basic idea for skip-gram modeling, is given a word wi , predict the context wi−2 , wi−1 , wi+1 , wi+2 of this word. Whereas for the continuous bag of words approach, this process is reversed; given the context wi−2 , wi−1 , wi+1 , wi+2 , predict word wi .



Where V is vocabulary size and N is hidden layer size. This is the network in its simplest form, since here there's only one input vector (basically you'd have given word wi predict word wi+1). So the matrix we're eventually interested in, is matrix A with where we have a linear activation function. The following image shows how the where we get the vector representations for the individual words come from.

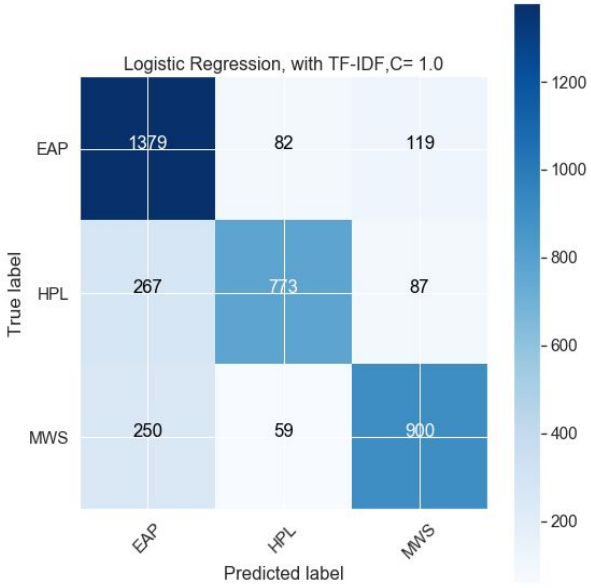


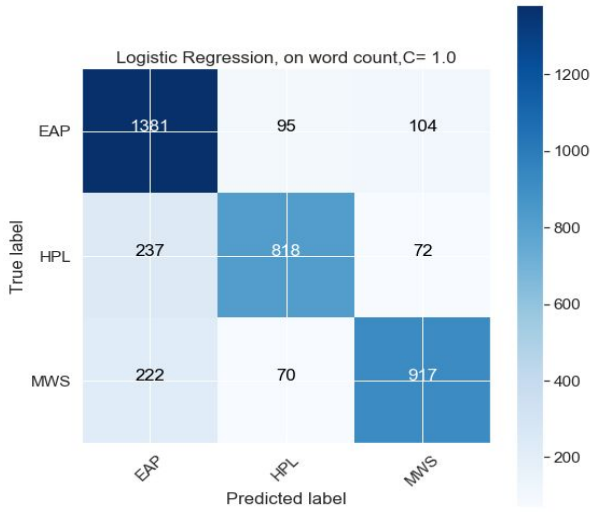
Thus, after training, if for example you'd like to have the vector representation of the jth word in the vocabulary, then you should get the jth row of matrix A (you're basically multiplying the jth unit vector with matrix A and you're left with the vector representation).

5. Results

5.1 Logistic Regression on TF-IDF and word count vectors:

The model with TF-IDF features performs with a multiclass log loss accuracy of 0.635. We have also implemented word count-features on this model, to realize that the multiclass log loss accuracy improves significantly to 0.532. The reason for this unexpected result could be that the models are not well "tuned", i.e. we are not comparing 2 tuned Logistic regression models(even though the Logistic regression is relatively robust to regularization of hyperparameters, the advantage of one model over other may be solely due to tuning of parameters). We wanted to test the fine tuning of logistic regression on penalty(L2 norm) and C , but as Naïve Bayes classifier was already performing well on this data set, we limited the Hyperparameter optimization to Naïve Bayes. Following are the confusion matrix for both the models



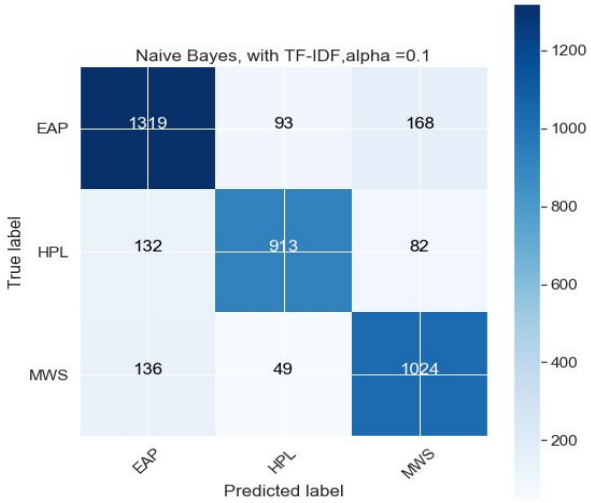


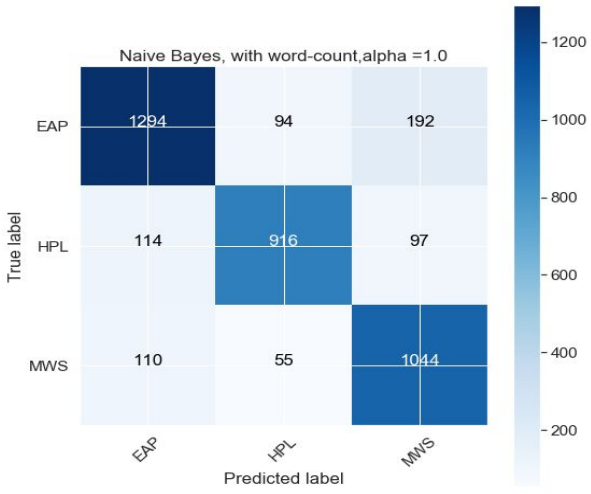
5.2 Support Vector Classification :

We found out for this dataset SVM performs weakly compared to Naïve Bayes classification. We have applied Singular Value decomposition (SVD) of TF-IDF features and took top 150 components as, SVM took a long time to run.We could only achieve a multiclass log loss of 0.733 on the basic SVM without hyperparameter tuning. The reason for Naïve Bayes to perform better than SVM could be that , they are both sensitive to parameter optimization (i.e. different parameter selection can significantly change their output) . That is , this is only true for the selected parameters. However, for another parameter selection, we might find SVM is performing better. Also, it could be that the assumption of independence of Naïve Bayes Classification is satisfied well by the Features derived from this dataset and the degree of class overlapping is small (the linear decision boundary).Also the Naive Bayes classifier is faster so we chose it over SVM.

5.3 Naïve Bayes Classification :

Just as in the case of logistic regression the word counts features (0.46 multiclass log-loss accuracy) are still better than the TF-IDF features (multiclass log-loss accuracy of 0.58).The reason for this unexpected result again could be that the models are not well "tuned", i.e. we are not comparing 2 tuned models. So we performed a grid search on both these models to find out that Naive Bayes on TF-IDF features indeed is performing better than the word count features after Hyperparameter tuning. The Multiclass log loss on Naive Bayes classifier with 0.1 alpha on TF-IDF features was 0.44 on the validation set.The same on the word count features was capped at 0.46 for an alpha 1.0 (optimal) .Following are the representations of Confusion matrices in both cases

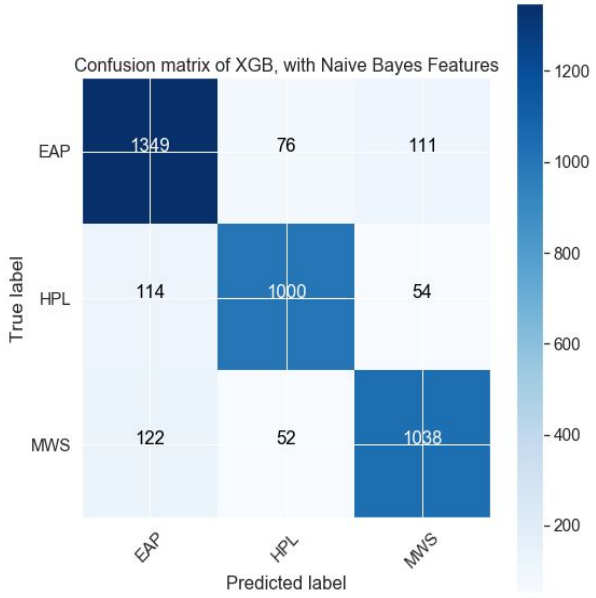




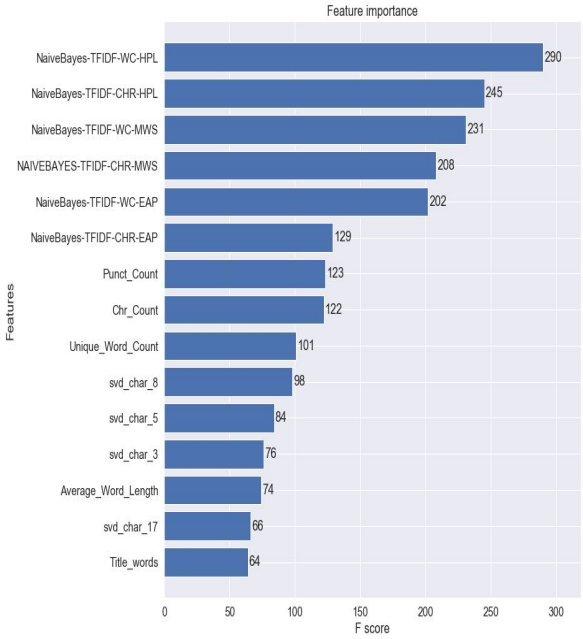
We have used the predictions from this model as features for the final Boost model we have used. These features have actually resulted among the top features in terms of feature importance in the model(based on F1 score)

5.4 XGBOOST using Naive Bayes Predictions and Top 150 SVD features based on Words and Character level vectors:

Along with other word based features like word counts, word lengths , punctuations,etc we have used the top 150 dense components of SVD decomposition of TF-IDF vectors of ngrams(we have used a mix of 1,2,3 word tokens). Also , we have used the Naive Bayes predictions based on these SVD components of word TF-IDF vectors and character TF-IDF vectors as features for the Final XGBoost model.This improved the multi log loss classification error of the base XGBoost model (based only on base features)from 0.98 to .33 on the validation set. Also, this not only is our best score but also is a very decent score based on the top 10 percentile submissions. Following is the representation of confusion matrix of our final model



Also, based on the current features following is the feature importance chart.



It is no surprise the Naive Bayes based features are of the highest predictive power.

To further improve this accuracy (If we were not time constrained) we wanted to try adding in more features based on Punctuations and other basic model outputs , also we could have tried Hyperparameter tuning for XGBoost just as we did in the case of Naive Bayes TF-IDF model, Also, we could try ensembling different models. Sentiment analysis based features could also add in value to this model.

6 Conclusions

In this analysis of short text data using advanced machine learning techniques to address a 3-way multiclass classification problem , we've incorporated many methods for text mining, ranging from tf-idf to n-gram tokenization. However, throughout the analysis , we've been able to rely on a small list of common tools for exploration and visualization. The learning from

this assignment for us is: the right feature scaling can be extremely helpful for classification. The right features built from well performing base models like Naïve Bayes can improvise the predictive power of the informative words and appropriately downscale the common words. Also, Hyperparameter tuning is the crucial element that needs to be performed before comparing the base models. TF-IDF features from Naïve Bayes classifier have performed better than the Simple word count based features on Naïve Bayes classification only after regularization of Laplace correction. Also, SVD to reduce the dimensionality of the sparse TF-IDF feature matrix has been a crucial element before using these features. The uniqueness in our approach was to explore basic models like SVM, Logistic regression and Naïve Bayes with and without regularization on both simple word-based features and TF-IDF text-based features, and then feature engineering the outputs of these basic models (only the high/better performing ones) to build features to train the XGBoost model. This worked for us as , the multi log loss classification error of the base XGBoost model (based only on base features)from 0.98 to .33 on the validation set.

Citations and References

[1]Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

[2]http://www.socsci.ru.nl/idak/teaching/batheses/MarciaFissette_scriptie.pdf

[3] L. van der Knaap and F. A. Grootjen. Author identification in chatlogs using formal concept analysis. In M. M. Dastani and E. de Jong, editors, Proceedings of the 19th Belgian-Dutch Conference on Artificial Intelligence (BNAIC 2007), pages 181–188, Utrecht, the Netherlands, November 2007.

[4] Rong Zheng, Jiexun Li, Hsinchun Chen, and Zan Huang. A framework for authorship identification of online messages: Writing-style features and classification techniques. J. Am. Soc. Inf. Sci. Technol., 57(3):378–393, 2006**.**

[5]https://web.stanford.edu/class/cs224n/reports/2760185.pdf

[6]https://www.kaggle.com/sudalairajkumar/simple-feature-engg-notebook-spooky-author/notebook

[7]https://www.kaggle.com/abhishek/approaching-almost-any-nlp-problem-on-kaggle

[8]Jianing Fang(2013): Why Logistic Regression Analyses Are More Reliable Than Multiple Regression Analyses. Journal of Business and Economics, ISSN 2155-7950, USA July 2013, Volume 4, No. 7, pp. 620-633