Introduction to Machine Learning
Spooky Author Identification

Abhijeet Ambadekar, Harsh Seth, Karthik Sharma, Satyarth Gohil, Yash, Tanisha

SCSE Department, VIT Chennai Campus, Chennai - 600127

Keywords: Test-Based Author Prediction, Multiclass classification, Feature Generation and Selection, Multinomial Naive Bayes, XGBoost

Abstract:

This report details the work done in an attempt to solve the Spooky Author Identification Challenge hosted on the online competitive platform Kaggle. The major problem posed by this challenge is that of feature engineering both meta feature as well as text-based ones. This was overcome by deriving features using techniques like TfIDf-MNB and SVD. The final result of this project leads to a kernel using Multinomial Naive Bayes for feature generation and the XGBoost model to train on these features and predict the results leading to a multiclass log-loss of 0.29.

Challenge Description:

This competition proposes a challenge to predict the author of excerpts from horror stories by Edgar Allan Poe, Mary Shelley, and HP Lovecraft. Given a training set of different lines of texts written by each author, the challenge is to predict given a line of text, the probability of each author writing the line.

The dataset given for this challenge has 19579 rows for training the model and 8392 rows to test its performance. The training data has columns consisting of an id for each line of text, the line of text and the author who wrote the line. The test set includes and id for a line of text, and the line of text.

The authors are given in the form of codes: EAP for Edgar Allan Poe, HPL for HP Lovecraft; MWS for Mary Wollstonecraft Shelley. The distribution of the dataset based on the authors was almost equal with a slight skewness towards EAP.

The submission requires the probability of each author writing the text for each line in the test set.

Initial Approaches:

One of the first few attempts was based on a basic statistical approach to solve the problem using NLP techniques. The central idea of this approach is to guess which author wrote a string of text based on the normalized unigram frequency, i.e. determining the count of how often each author uses every word in the training data and then divide by the number of total words the author wrote.

This was an intuitive approach for any text based machine learning problem and would serve as a starting point for any solutions designed in the future.

The tools used to implement this approach was the python NLTK library to tokenize the text into words and generate a probability distribution of the list of words for each author using the word frequency in each sentence. For a new given sentence, the joint probability distribution was computed for each author for all the words in the sentence and the one with maximum probability was chosen to be the author of the sentence.

This approach being naive, led to rather poor accuracy than expected and hence was abandoned, though it served as a good base model to build other approaches upon.

Another approach was attempted in a completely different direction than the previous one, by using some features derived from the text in the training of a Recurrent Neural Network.

Another approach was purely focussed on feature generation using TF-IDF Vectorization as a derivative of the first base approach and training some simple models on it.This was applied on top of a naive bayes classifier.

A for loop was applied upon the engraming techniques to improve the model.

Final Approach:

All the previous approaches led to the following conclusions: Feature generation and selection is a very important task to train the model on, and a major part of solving the problem is to preserve the context of words.

Thus the final approach resulted in a bulk of feature generation (both textual and meta features) and using the ensemble approach of XGBoost model to train on these features and produce the results.

The meta features selected for training were:

- Number of words in the text
- Number of unique words in the text
- Number of characters in the text
- Number of stopwords
- Number of punctuations
- Number of uppercase words
- Number of title case words
- Average length of the words

Training on these meta features resulted in a multiclass log-loss of 0.987 which was a good starting point to generate further text-based features.

The text-based features were generated using TF-IDF Vectorization resulting in a sparse matrix of features. This matrix was then compressed by using Singular Value Decomposition adding 20 features each for words and character tokens.

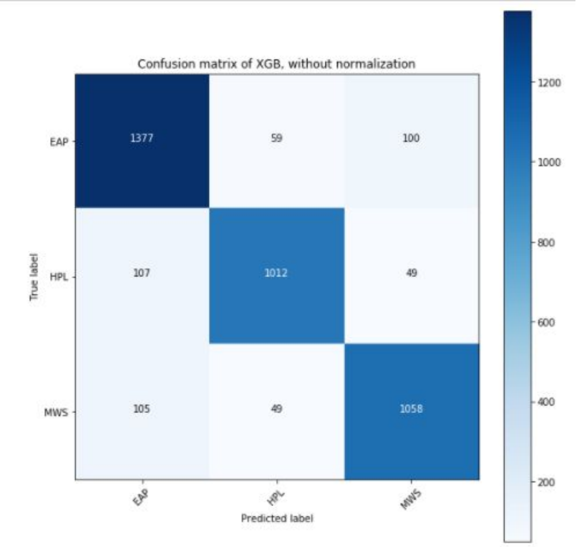
Another method to derive some parameters from the sparse matrix it to Build a model using just the sparse features and then use the predictions as one of the features along with other dense features.Thus, a One Vs Rest Classifier was trained using the Multinomial Naive Bayes classification model on count of words and character and the predicted probabilities of both cases were selected as features for training using the XGBoost model.

The utility of each generated feature was determined by using cross validation on the Naive Bayes models and interpreting the outcomes of the confusion matrix of the true values to the predicted values for each author.

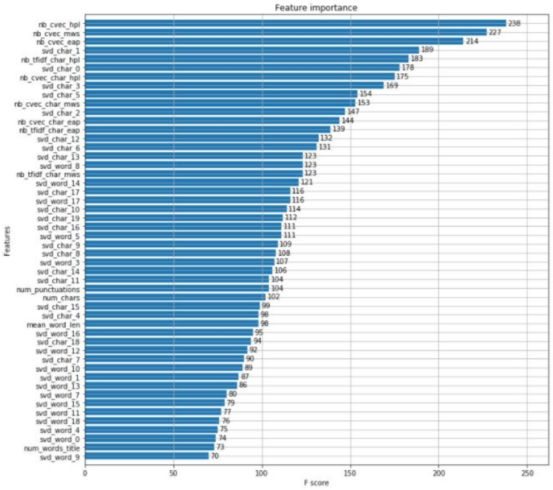
The XGBoost model was trained on all the generated features on the following hyper-parameter list:

- objective => 'multi:softprob',
- eta => 0.1,
- max_depth => 3,
- silent => 1,
- num_class => 3,
- eval_metric => "mlogloss",
- min_child_weight => 1,
- subsample => 0.8,
- colsample_bytree => 0.3,
- seed => None

The XGBoost model produced the probabilities for the authors for test set with a multiclass log-loss of 0.29 and a confusion matrix as follows:



One of the functionalities of the XGBoost model is to return the importance of the each feature in the form of F-Score leading to the following graph.


Feature importance
nb_cvec_hpl 238
nb_cvec_mws 227
nb_cvec_eap 214
svd_char_1 189
nb_tfidf_char_hpl 183
svd_char_0 178
nb_cvec_char_hpl 175
svd_char_3 169
svd_char_5 154
nb_cvec_char_mws 153
svd_char_2 147
nb_cvec_char_eap 144
nb_tfidf_char_eap 139
svd_char_12 132
svd_char_6 131
svd_char_13 123
svd_word_8 123
nb_tfidf_char_mws 123
svd_word_14 121
svd_char_17 116
svd_word_17 116
svd_char_10 114
svd_char_19 112
svd_char_16 111
svd_word_5 111
svd_char_9 109
svd_char_8 108
svd_word_3 107
svd_char_14 106
svd_char_11 104
num_punctuations 104
num_chars 102
svd_char_15 99
svd_char_4 98
mean_word_len 98
svd_word_16 95
svd_char_18 94
svd_word_12 92
svd_char_7 90
svd_word_10 89
svd_word_1 87
svd_word_13 86
svd_word_7 80
svd_word_15 79
svd_word_11 77
svd_word_18 76
svd_word_4 75
svd_word_0 74
num_words_title 73
svd_word_9 70
Features
F score