Modeling versus Clustering for Text Classification

Michelle Li

June 24, 2019

Abstract

Classification is a supervised learning problem that can be approached using any model capable of classification. Clustering is an unsupervised learning problem where the resulting clusters of a clustering algorithm do not attempt to classify the input data. There is no straightforward method for comparing modeling and clustering algorithms on the task of classification. In this paper we propose a method for estimating the accuracy of a clustering algorithm on a text classification problem. We find that the best performing model, multinomial Naïve Bayes, is ~40% more accurate than the best performing clustering algorithm, spectral clustering.

1. Introduction

Text classification requires extraction of features from text into a numerical form. Several feature extraction methods are explored: bag of words, term frequency-inverse document frequency, positive pointwise mutual information, and word2vec. Various modeling algorithms are used to classify the texts by author and clustering algorithms are used to find clusters within the data. Accuracy is used to evaluate the modeling algorithms. The figure of merit proposed by (Levine & Domany, 2000) and the adjusted Rand index are used to evaluate cluster stability. In this paper we propose an accuracy estimation method for clustering algorithms based on resampling. We use these estimated accuracy scores to compare how modeling and clustering perform on this text classification problem.

2. Data Set and Preparation

The data were obtained by downloading novels as plain-text files from the "Books about Love stories (sorted by popularity)"[1] list from Project Gutenberg. Novels with fewer than 500,000 characters were rejected from the data set and replaced with new novels until ten novels by ten different authors were acquired. Text cleaning was performed to remove idiosyncrasies such as chapter headings. Some novels had quotes introducing each chapter, and they were not removed. It may be beneficial to remove these quotes because they are not reflective of the authors' word choices and phrasing. However, the time and effort required may outweigh the potential benefit, since the multinomial Naïve Bayes model achieved ~90% accuracy even with the quotes.

After cleaning the text, each novel was tokenized with spaCy's default tokenizer and split into 250-word chunks. 250 words was chosen because according to Cabot[2], books tend to have 250-300 words per page. The first 480 chunks from each novel were taken to create a data set of

[1] https://www.gutenberg.org/ebooks/subject/2487

[2] https://www.megcabot.com/about-meg-cabot/frequently-asked-questions-getting-published/

4,800 chunks. 25% of these chunks (1,200 chunks) were selected randomly and set aside as the holdout set. The remaining 75% (3,600 chunks) were used for training and validation.

3. Feature Extraction

Bag of words, term frequency-inverse document frequency (tf-idf), positive pointwise mutual information (PPMI), and word2vec were used to extract features from the data. These feature sets were created using lower-cased token lemmas while excluding stop words, proper nouns, and punctuation. Proper nouns were excluded to make the classification task more difficult because each novel has a set of character names which appear repeatedly throughout the novel. Scikit-learn's CountVectorizer and TfidfVectorizer classes were used to create the bag of words and tf-idf feature sets. PPMI word vectors were calculated following the method in the "Word Vectors from PMI Matrix"[3] Kaggle notebook by Altay, and Laplace smoothing[4] was applied. The Gensim library was used to create the word2vec word vectors. The PPMI and word2vec word vectors were averaged to create feature sets for the 250-word chunks.

Hyperparameters for creating the bag of words and tf-idf feature sets were tuned by performing a randomized search with Scikit-learn's RandomizedSearchCV class and selecting the parameters which resulted in the highest cross-validation score (cv=5) for a multinomial Naïve Bayes model. Multinomial Naives Bayes was used because it is fast and often used as a baseline for text classification. Hyperparameters for creating the word2vec feature set were tuned similarly, except a Gaussian Naïve Bayes model was used instead because the resulting feature values can be negative, which are not valid inputs for multinomial Naïve Bayes. The single hyperparameter for creating the PPMI feature set was tuned using multinomial Naïve Bayes through trial and error.

Each feature set was reduced to 421 dimensions, which was chosen according to the formula for estimating LSA dimension by (Fernandes, Artífice, & Fonseca, 2011). LSA was used instead of principal component analysis (PCA) because PCA mean-centers the data before performing singular value decomposition (SVD), so counts of 0 become informative when they are not. LSA performs SVD decomposition before normalizing the data, so counts of 0 remain uninformative. For the PPMI feature set, SVD decomposition was performed on the word vectors, then the word vectors were averaged for each 250-word chunk, and finally the averaged vectors were normalized. Gensim's word2vec implementation has a hyperparameter for controlling word vector size, so SVD decomposition was not applied to the resulting word vectors, but the word vectors were normalized after averaging them for each 250-word chunk. Since the dimensionality of the word2vec word vectors were not reduced with LSA, 421 may not create the best possible feature set. This was deemed acceptable because the estimated dimension of 421 is not guaranteed to be the optimal LSA dimension for the other feature sets either.

---

[3] https://www.kaggle.com/gabrielaltay/word-vectors-from-pmi-matrix#Word-Word-Count-Matrix
[4] https://en.wikipedia.org/wiki/Additive_smoothing

4. Modeling

Scikit-learn's implementations for logistic regression, Bernoulli Naïve Bayes, multinomial Naïve Bayes, Gaussian Naïve Bayes, support vector machine, random forest, and multi-layer perceptron were trained on each feature set, except for the Bernoulli Naïve Bayes and multinomial Naïve Bayes models. These two models do not accept the negative values resulting from LSA, so they were trained on the original bag of words and tf-idf feature sets. Hyperparameters were tuned by performing a randomized search. The models were compared using the mean accuracy from cross-validation (cv=5) on the training set. Accuracy was used so that the models' scores could be compared to the clustering algorithms' accuracy scores later. Logistic regression, Bernoulli Naïve Bayes, multinomial Naïve Bayes, support vector machine, and multi-layer perceptron had the highest scores, so they were tested on the holdout set. The accuracies for each model are listed in Table 1.

Model	Mean CV Accuracy	Holdout Accuracy	Feature Set
Multinomial Naïve Bayes	0.894 (0.875, 0.913)	0.908	bow *
Bernoulli Naïve Bayes	0.894 (0.866, 0.922)	0.902	tf-idf **
Multi-layer Perceptron	0.843 (0.819, 0.867)	0.872	tf-idf
Logistic Regression	0.872 (0.841, 0.903)	0.869	tf-idf
Support Vector Machine	0.860 (0.814, 0.906)	0.851	tf-idf
Gaussian Naïve Bayes	0.728 (0.680, 0.776)	--	tf-idf
Random Forest	0.609 (0.591, 0.627)	--	tf-idf

Table 1. Highest mean accuracy scores from cross-validation (cv=5) on the training set and on the holdout set. Gaussian Naïve Bayes and Random Forest were not tested on the holdout set.
* The original bag of words feature set with binary=False and no dimensionality reduction.
** The original tf-idf feature set with no dimensionality reduction.

5. Clustering

5.1 Adjusted Rand Index - Choosing the Best Clustering Algorithm

Scikit-learn's implementations for mean shift, spectral clustering, K-means, affinity propagation, and DBSCAN were used to find clusters in each feature set. The clustering algorithms were first compared using the adjusted Rand index (ARI) because it is faster to calculate than the figure of merit. ARI was used because the clustering algorithms under consideration make different assumptions about the structure of the data. It is bounded between [-1, 1] and the best score is 1. Spectral clustering and K-means were applied to each feature set with n_clusters=10 because there are 10 classes in the data. The other algorithms do not take the number of clusters as a parameter, so they were tuned through trial-and-error to achieve the highest ARI possible. Spectral clustering and K-means had the highest ARI scores, so they were selected for further examination. The highest ARI scores for each algorithm are listed in Table 2.

Spectral clustering and K-means were applied to 5 resamples of 90% of the data for n_clusters from 10 to 39. Spectral clustering with affinity=linear had the highest scores and n_clusters=20 was chosen for further examination because it had the lowest standard deviation.

Clustering Algorithm	ARI	Feature Set
Mean Shift	0.023	word2vec
Spectral Clustering	0.164	tf-idf
K-means	0.184	tf-idf
Affinity Propagation	0.020	tf-idf
DBSCAN	0.004	tf-idf

Table 2. Highest ARI scores for each clustering algorithm on the entire training set. A score of 0 indicates randomly assigned cluster labels, and the best score is 1.

5.2 Figure of Merit – Structural Stability

Spectral clustering with affinity=linear and n_clusters=20 was applied to 152 resamples of 66% of the training data to calculate the figure of merit described by (Levine & Domany, 2000). 66% was recommended by (Levine & Domany, 2000), and 152 was chosen so that each observation would appear about 100 times. The figure of merit calculates the proportion of data that are still clustered together in the resample and is bounded between [0, 1] with 1 as the perfect score. The mean figure of merit could be calculated for the holdout set in the same way. However, this does not provide insight into whether the clusters found in the training set are similar to those found in the holdout set. To capture this insight, the final evaluation of the clustering algorithm on the holdout set was performed by applying the clustering algorithm 152 times to the holdout set plus a random 66% of the training set and the figure of merits were calculated separately for the training set and holdout set observations (Table 3).

Data Set	Mean Figure of Merit
Entire Training Set	0.438 (0.360, 0.516)
Holdout + 66% Training (Training Observations)	0.415 (0.369, 0.461)
Holdout + 66% Training (Holdout Observations) *	0.458 (0.400, 0.516)

Table 3. Mean figure of merits for spectral clustering with affinity=linear and n_clusters=20 on 152 resamples. 1 is the best score.
* This dataset was resampled only 100 times because every holdout observation appears in each resampling.

5.3 Estimating Accuracy and Prediction Stability

The clustering algorithm does not tell us which author each cluster represents, but we know the ground truth, so we can assume each cluster represents the author for which most of its cluster members represent. This translation from the clustering algorithm's output to author labels was performed for each of the 152 resamples of the training data. Each observation appeared about 100 times and the most frequently predicted author for each observation was considered the clustering algorithm's final prediction for that observation. Final accuracy can be calculated using these final predictions (Table 4). We can also calculate the "prediction stability" of the final predictions by examining how often each data point was predicted to be the final prediction

across all the resamples and find the median prediction stabilities for authors correctly and incorrectly predicted (Figure 1).

Estimating accuracy on the holdout set could be done by applying the clustering algorithm to the holdout set, but this approach does not provide insight into whether authors predicted for the training observations are similarly predicted for the holdout observations. To capture this insight, the final evaluation of the clustering algorithm on the holdout set was performed by applying the clustering algorithm 152 times to the holdout set plus a random 66% of the training set (Figure 1). 78% of the final predictions for the training observations in this combined set were the same as for the training set (Figure 2).

Data Set	Final Accuracy
Entire Training Set	0.539
Holdout + 66% Training (Training Observations)	0.521
Holdout + 66% Training (Holdout Observations)	0.502

Table 4. Final accuracies for spectral clustering with affinity=linear and n_clusters=20 on 152 resamples.

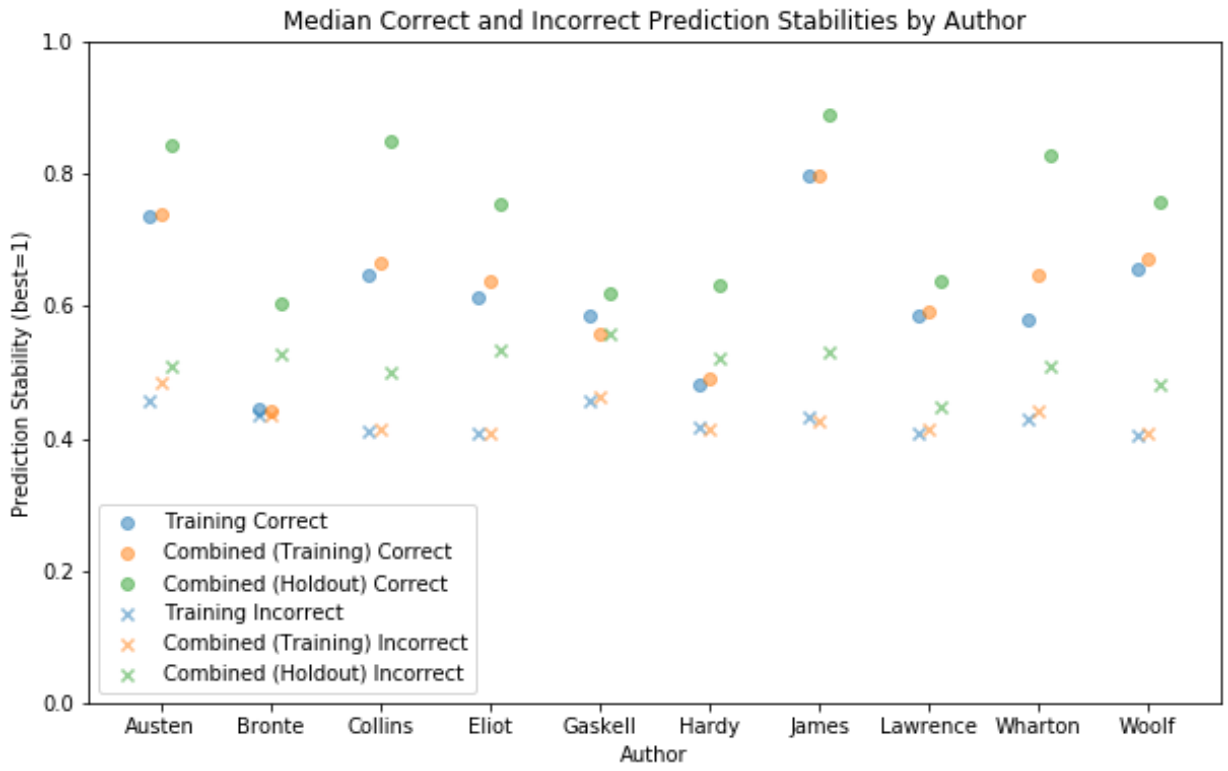

Figure 1. Median correct and incorrect prediction stabilities for each author. Authors with a larger difference between the correct and incorrect prediction stability are more consistently identified by the clustering algorithm.

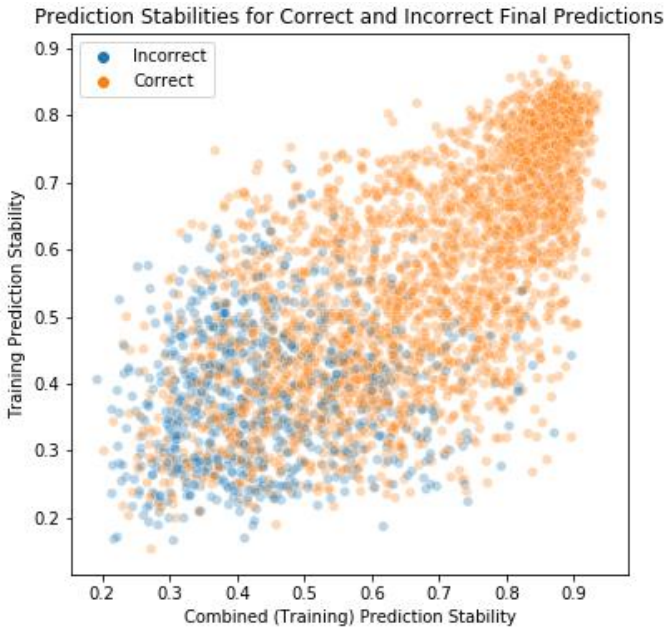

Figure 2. Prediction stabilities for correct and incorrect final predictions in the training set and the combined set (training observations). 78% of the final predictions for the training observations in the combined set were the same as for the training set.

6. Conclusions

In this paper we propose a method for estimating the accuracy of a clustering algorithm on a text classification problem based on resampling. The structural stability of the clusters as quantified by the figure of merit is consistent across the training set and the combined sets created by combining the holdout set with random 66% samples of the training set. The difference in median prediction stabilities for correct and incorrect predictions varies between authors. Authors with a larger difference are more consistently identified by the clustering algorithm. The clusters from the training set do not change dramatically when the holdout set is introduced because 78% of the final predictions remain the same.

The models consistently outperform the clustering algorithm on the training set and the holdout set. We do not see any advantages to using clustering over modeling for this text classification problem. Training the models is faster than clustering many resamples, and the models have higher accuracy. However, it is possible that the clustering algorithm found structure in the data that does not represent authors. If the problem were not to classify but to find structure, the clustering algorithm would be preferred.

References

Fernandes, J., Artífice, A., & Fonseca, M. J. (2011). Automatic Estimation of the LSA Dimension. *International Conference on Knowledge Discovery and Information Retrieval (KDIR'11)*, (pp. 309-313).

Levine, E., & Domany, E. (2000). Resampling Method For Unsupervised Estimation Of Cluster Validity. *Neural Computation*.