A Binary Classification of Beatles Song Authorship

Miles Bennett, Casey Haaland, and Atsu Kobashi

I. Introduction

MACHINE learning algorithms have a long history with the problem of text classification which has been used extensively for a variety of applications including spam classification as well as the identification of the disputed authorship of several Federalist Papers [1][2]. In our project, we attempt to use the tools of machine learning to identify the primary author of a given Beatles' song using the lyrics as features. Since a large majority of Beatles songs were written by either John Lennon or Paul McCartney, we restricted the scope of our project to a simple binary classification problem. Though text classification is a well known and widely used application of machine learning, the aim of this project is to analyze the performance of various classification algorithms and quantify their suitability for this problem.

II. Methodology

A. Data Collection and Preprocessing

After converting a PDF of Beatles lyrics to plain text format we first separated the set of lyrics into separate text files for each song [3]. In the first iteration of testing, we also removed words we believed were content free, in an attempt to improve the accuracy of our classifier. Since some basic words were likely to appear in a vast majority of the songs, we felt it was likely that these words would not be informative features in determining authorship so we would be justified in removing them. As such, we defined the following words which would *not* be included in the feature space:

1) "the"
2) "a"
3) "and"
4) "or"
5) "but"
6) "if"

In addition to removing these so called "stop words", we also lemmatized the song lyrics using a library provided by the *Natural Language Toolkit* (NLTK), an open source module for python [4]. The lemmatization process converts a noun, verb, or adjective to the canonical form of the word. For example, the plural noun "dogs" would be converted to the singular noun "dog" and the word "went" would be converted to "go." By lemmatizing the aggregate list of all words that appear in the corpus of Beatles songs, the size of the vocabulary was reduced from 1435 to 1142. This is a similar preprocessing technique that was used in Problem Set #2.

Lastly, because there are only 93 songs written by either Lennon or McCartney, our data set was relatively small. For this reason, in the algorithms to follow, all accuracies are reported under leave one out cross validation. Specifically, we evaluated our experimental *accuracy* as:

$$\hat{\tau}_{LOOCV} = 1 - \frac{1}{n}\sum_{i=1}^{n} 1\left\{h_\theta\left(\boldsymbol{x}^{(i)}\right) \neq y^{(i)}\right\}$$

Since 43 of the songs were by McCartney and 50 by Lennon, there was very little class imbalance and this seemed to be an appropriate metric for the classifier.

B. Algorithms

There are a variety of well known algorithms in the field of machine learning for the purpose of binary text classification. One possibility that our group hypothesized was that songs written by Lennon and songs written by McCartney would lie close together in a cluster. For this reason we wanted to explore the use of non-parametric clustering algorithms such as k-means or k-Nearest Neighbors as well as investigate the effect of using different distance metrics (e.g. ℓ_1 and ℓ_2) on our classification accuracy.

Another well known algorithm considered to be both simple and effective for text classification problems is Naive Bayes. The large volume of literature concerning the use of Naive Bayes for textual analysis seems to suggest both the Bernoulli and Multinomial event model would be well suited for our classification problem.

The last two algorithms we seek to investigate in this project are Support Vector Machines and regularized logistic regression. SVMs have many desirable properties such as large margin, the ability to be kernelized, and a relatively low tendency to overfit, all of which would allow it to perform well on this problem. Logistic regression is generally accepted as an algorithm that performs well on binary classification. With the addition of a regularization term we can obviate the problem of overfitting that logistic regression tends to suffer from.

III. Results

A. Initial Classification Approach

Before attempting the algorithms listed above, we first attempted to see if there was a simple feature space in which we could visualize and perhaps find structure in the data. One feature that was often cited as being useful for the purpose of text classification was the number of unique words per document. In accordance with this, we created a scatter plot of the number of unique words *vs.* song length.

As evidenced by Figure 1, this two dimensional feature space was very densely populated with little discernible pattern that would enable us to distinguish Lennon from McCartney. Therefore, we found that lack of linear separability or distinct clustering meant this feature space was not rich enough to accurately perform the classification. This observation was

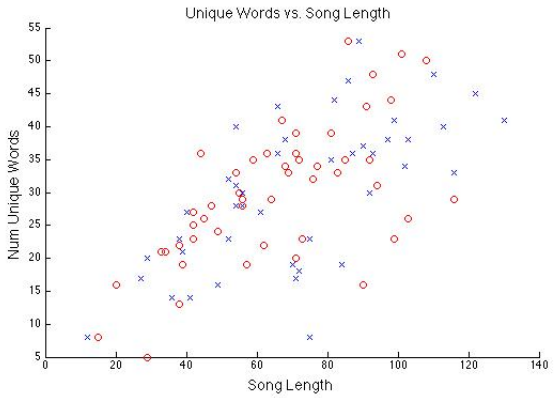

Fig. 1. The circles denote songs composed by John Lennon while the x's denote ones written by Paul McCartney

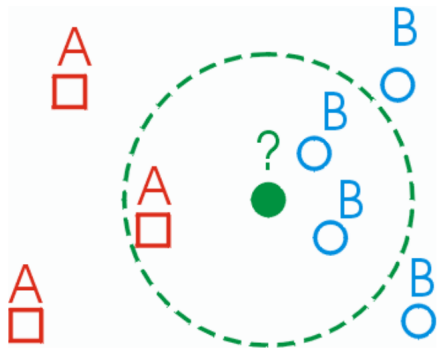

Fig. 2. Graphical depiction of the k-nearest neighbors algorithm. The dashed circle corresponds to the chosen distance metric

verified by our initial attempts using *k-NN* which yielded an accuracy of 45.16%.

In hopes of better capturing more information that could be used to discriminate between the two artists, we chose to create a design matrix $X \in \mathbb{R}^{93\times 1435}$ and a classification vector $y \in \mathbb{R}^{93}$

$$X = \begin{bmatrix} - & x^{(1)} & - \\ & \vdots & \\ - & x^{(93)} & - \end{bmatrix} \qquad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(93)} \end{bmatrix}$$

where $x_j^{(i)}$ gives the number of times the j^{th} word in our vocabulary V appears in the i^{th} Beatles' song in our training set where $y^{(i)} = 1$ if McCartney was the artist of song i and $y^{(i)} = 0$ if Lennon was the artist.[1]

B. k-means & k-Nearest Neighbors

The *k*-means algorithm is an unsupervised learning algorithm that clusters data based on a chosen distance metric. This algorithm is ideal in the case that the convex hulls of the points from each class are mutually disjoint.[2] The k-Nearest Neighbors (*k-NN*) algorithm is another non-parametric algorithm in which the class label of a new example is predicted by a majority vote of the *k* nearest neighbors (where nearest was measured by either the ℓ_1 or ℓ_2 norm). We chose to break ties by using the class label of the closest example to the test point. It is also worth noting that *k-NN* exploits local clustering of data and as a result tends to perform well on data in which classes lie in a number of dense clusters [5].

[1]For the case of the SVM, the class designations are swtiched so that $y^{(i)} = -1$ identifies a song composed by Lennon rather than 0

[2]The convex hull of a set of points $S = \{x_1, \cdots, x_n\}$ is simply the set of all convex combinations of points in the set (i.e. conv$(S) = \{\lambda^T x |$ $0 \prec \lambda,\ \lambda^T \mathbf{1} = 1\}$

Initially, we supposed that the data for the Beatles' songs written by each author might lie in two distinct clusters of our feature space. This situation might occur if, for example, McCartney used one subset of the vocabulary much more frequently than Lennon did. Another hypothesis for the structure of our data that we thought was reasonable was that the classifications might lie close together in a number of smaller clusters in the feature space. Since both of these seemed to be plausible assumptions about the data, we used the *k*-means and *k-NN* MATLAB implementations, iterating over the number of neighbors and utilizing the 1-norm and 2-norm, the results of which are shown in the table.

Algorithm	k	**Accuracy**	Metric
k-means	2	53.76%	ℓ_2
k-NN	1	54.84%	ℓ_2
k-NN	1	53.76%	ℓ_1
k-NN	2	37.63%	ℓ_2
k-NN	2	25.81%	ℓ_1
k-NN	5	49.46%	ℓ_2
k-NN	5	50.54%	ℓ_1

As evidenced by the poor performance of *k*-means, it was clear the data did *not* lie in two distinct clusters of our chosen feature space. This result might be explained by Lennon and McCartney having similar writing styles and therefore the different classes were highly interspersed amongst each other. The negative results of both algorithms could also be attributed to the curse of dimensionality, as the feature space is over 1400 dimensional while the number of test examples was just 93.

C. Naive Bayes

Naive Bayes is a simple generative learning algorithm that is easily applied to binary classification problems. The key assumption of the algorithm is the conditional independence of each of the words given the document's classification (i.e. $p(x_i, x_j|y) = p(x_i|y)p(x_j|y)$. The algorithm then seeks to learn a model by computing

$$\max_\theta p(\boldsymbol{x}, y) = \max_\theta p(y) \prod_{i=1}^{n} p(x_i|y)$$

where θ is the vector of parameters $p(y = 1)$, $p(x_i|y = 1)$, and $p(x_i|y = 0)$ which are to be estimated. In our initial attempts

to apply Naive Bayes we implemented both the Bernoulli and Multinomial event models with Laplace smoothing. The results of which are summarized below.

Event Model	Accuracy
Bernoulli	62.37%
Multinomial	53.76%

As was the case with the previous learning algorithms, Naive Bayes only slightly outperformed random guessing. The reasons for the failure of this particular learning algorithm is most likely due to the small training set as well as the conditional independence assumption being violated.

D. SVMs & Regularized Logistic Regression

The last two algorithms that we attempted were Support Vector Machines and regularized logistic regression, both of which are supported by the LIBLINEAR package [6]. In order to fully investigate the range of classifiers available, we used a variety of objectives for the SVM algorithm such as ℓ_2 loss with ℓ_2 penalty:

$$J_1 = 1/2\|\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{\xi}\|_2^2$$

as well as ℓ_2 loss with ℓ_1 penalty:

$$J_2 = 1/2\|\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{\xi}\|_1$$

among other options supported by the software package. For the logistic regression learning algorithm we also experimented with values of the regularization term for the objective

$$J_{LR}(\theta) = \sum_{i=1}^{m} \log\left(1 + \exp\left(-y^{(i)}\theta^T\tilde{\boldsymbol{x}}^{(i)}\right)\right) + \lambda\|\theta\|_2^2$$

with $\theta, \tilde{\boldsymbol{x}}^{(i)} \in \mathbb{R}^{94}$ where $\tilde{\boldsymbol{x}}^{(i)} = \left[\left(x^{(i)}\right)^T 1\right]^T$. Assuming we have a cost function J for the *unregularized* algorithm, our objective for the *regularized* learning algorithms take the form

$$J + \lambda \cdot f_p\left(\|\boldsymbol{x}\|_p\right)$$

where $\|\boldsymbol{x}\|_p$ is defined for $p \geq 1$ as

$$\|\boldsymbol{x}\|_p \triangleq \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$

and $f_p : \mathbb{R}_+ \to \mathbb{R}_+$ is the mapping

$$f(z) \triangleq \begin{cases} z & \text{if } p = 1 \\ z^2 & \text{if } p = 2 \end{cases}$$

The value $\lambda \in \mathbb{R}$ is the regularization parameter which helps stabilize the objective J [7]. To choose the value of the regularization parameter, we iterated over a logarithmic spacing of values from 0.0001 to 5000 and computed the resulting accuracies for each value of λ. The accuracies reported in table below are for

$$\hat{\lambda} \approx \arg\max_{\lambda} \hat{\tau}_{LOOCV}$$

Specifically, we selected the value of λ that produced the maximum accuracy on our set of examples under Leave One Out Cross Validation.

Algorithm	Loss	Penalty	Accuracy	$\hat{\lambda}$
Logistic	log-loss	ℓ_2	55.69%	0.0015
SVM	ℓ_2	ℓ_2	58.06%	6.9 $\times 10^{-4}$
SVM	ℓ_1	ℓ_2	61.29%	0.002
SVM	ℓ_2	ℓ_1	63.44%	0.02

IV. An Alternative Feature Space

In light of the poor performance of many of the algorithms we used to perform the classification, we sought alternative feature mappings for the lyrical data. In the course of our research, we discovered *term frequency - inverse document frequency* (*tf-idf*) is a standard feature mapping that performs well in text classification. To perform the feature mapping we define the following quantities [7]

$$n_w(\boldsymbol{x}) = \text{number of times word } w \text{ appears in doc } \boldsymbol{x}$$

$$f(w|\boldsymbol{x}) = \frac{n_w(\boldsymbol{x})}{\sum_w' n_{w'}(\boldsymbol{x})} = \text{ term frequency}$$

$$\phi_w(\boldsymbol{x}) = f(w|\boldsymbol{x}) \log\left[\frac{\text{\# of docs + 1}}{\text{\# of docs with word } w + 1}\right]$$

The "term frequency" portion of the mapping gives a weighting corresponding to how often word w appears in document $\boldsymbol{x}$ while the "inverse document frequency" portion serves to deemphasize words which appear in a large fraction of the documents. The addition of one in both the numerator and denominator of the *idf* term acts as a form of smoothing and prevents division by zero in the case that a word does not appear [8].

Using this new feature mapping we reran a selection of our previous algorithms to see if the new feature mapping $\phi_w(\boldsymbol{x})$ garnered any improvement in classification accuracy and perhaps be worth pursuing. The following table summarizes the results of implementing the *tf-idf* frequency feature mapping.

Algorithm	Loss	Penalty	Accuracy	$\hat{\lambda}$
Logistic	log-loss	ℓ_2	60.22%	2.5
SVM	ℓ_1	ℓ_2	58.06%	10
k-means	-	-	44.09 %	-

We speculate that the reasons for the poor performance of *tf-idf* for this particular application are similar to the reasons why many of our aforementioned algorithms failed. More specifically, we were using very few training examples relative to the size of the feature space we were attempting to learn. Additionally, since each training example consists of the lyrics to a song, rather than a much larger body of work, say an essay or a book, the examples will not be as informative as would be the case for classification of larger documents.

V. Reducing the Feature Space Dimensionality

Lastly, with most of the algorithms unable to achieve any significant accuracy, we explored the possibility of reducing the dimensionality of the data. Liang et. al. remarks, "low-content 'stop words' have been shown to be very useful statistical cues of sentiment and psychology" which led us

to believe that perhaps we were not justified in removing the stop words from our vocabulary [9].

Futhermore, Fung discusses in his paper "*The Disputed Federalist Papers: SVM Feature Selection via Concave Minimization*" the use of a smaller set of 70 "function words" rather than a complete vocabulary for improving text classification [2]. In fact, Fung demonstrates the ability to classify 12 disputed Federalist Papers using only a three dimensional feature space consisting of the words 'upon' , 'would', and 'to'.

In order to choose the subset of say, 10 words, that provide the best accuracy to use as our feature space, we would need to try $\binom{1435}{10} \geq \left(\frac{1435}{10}\right)^{10} \approx 4 \times 10^{21}$ different combinations of words, a clearly intractable problem already, let alone for a vocabulary of size 20. In light of this, we decided that implementing a simple forward search would be the best way to generate of low dimensional representation of our data.

Starting with an empty vocabulary, we built up a lexicon by iterating over the inclusion of every additional word that was not already in the set and subsequently included the word that achieved the highest value of $\hat{\tau}_{LOOCV}$. The number of words we ultimately included in our vocabulary was selected as the number of words after which the addition of several new features no longer improved training accuracy. Figure 3 shows how the training accuracy increase as a function of the vocabulary size under forward search. For the SVM algorithms, the first item in the legend denotes the regularization norm and the second specifies the loss function for the objective.

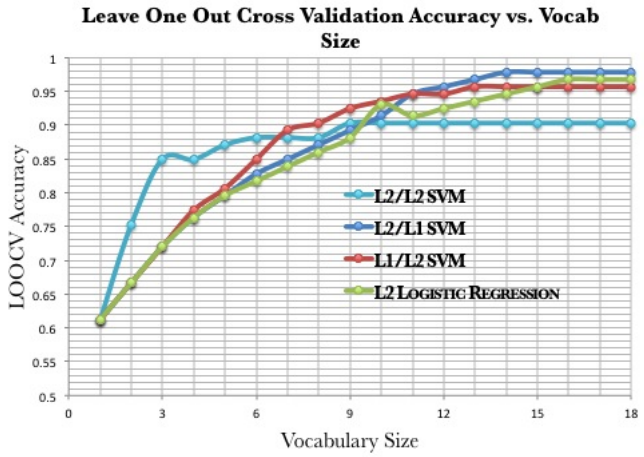

Fig. 3. Training accuracy as a function of vocabulary size for various optimization objectives. All algorithms were trained using regularization parameter C = 1

It can clearly be seen from the graph above that using this greedy approach to choose a subset of our feature space had a profound effect on classification accuracy. As can be seen from the plot, the accuracy grows quickly as new words are added to the vocabulary (at least 5% with each added word across all algorithms) and then almost all of the algorithms' training accuracies begin to plateau after 13 words, with the ℓ_2 regularized ℓ_1 penalty SVM performing the best, attaining a classification accuracy of 97.85%

Before moving on, there are some interesting patterns in Figure 3 that are worth drawing attention to. First, the ℓ_2 regularized ℓ_2 loss SVM learning algorithm (shown in light blue) learned far better with a small feature space than any of the others, initially gaining nearly 10% accuracy for each word added to the vocabulary (after the first word). However, it ended up performing the worst by the completion of the forward search, with the learning curve tapering off to $\approx 90\%$ after only 9 words. On the other hand, the ℓ_2 regularized ℓ_1 loss SVM, which ended up performing the best, experienced one of the slower learning rates, coming in with consistently lesser or equal accuracies to the other three learning objectives.

In addition to examining the Accuracy vs. Number of Features curve, we also looked at the vocabularies generated. The words obtained from the forward search using the ℓ_2 regularized, ℓ_1 penalty SVM is displayed below

1) it's	7) bag	13) answer
2) would	8) everybody's	14) remember
3) can	9) fill	15) sky
4) alone	10) float	16) about
5) stop	11) mr	17) above
6) do	12) after	18) accidents

Though using forward search with the other three objectives *did* result in different vocabularies for each algorithm, the first several words generally tended to be the same across all the algorithms indicating that there are certain words which are particularly important in distinguishing the two composers.

Seeing that the forward search provided such dramatic improvements in classification accuracy for the SVM and logistic regression based algorithms, we also implemented a forward search using Naive Bayes.[3] As we had come to expect from our research as well as our success using SVMs, the Naive Bayes approach saw a remarkable increase in classification accuracy when used in conjunction with feature selection. While the Naive Bayes algorithm using the original 1435 dimensional feature space only achieved $\hat{\tau}_{LOOCV} = 53.76\%$, using feature selection to build up a 12 word vocabulary we were able to increase this number to nearly 94%.

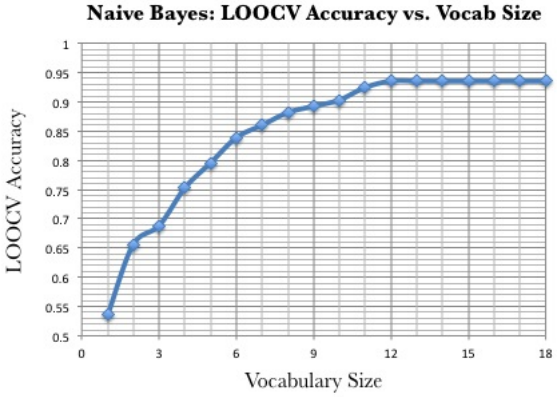

Fig. 4. The learning curve for Naive Bayes plateaus after a 12 dimensional feature set is achieved, attaining an accuracy of 94% at saturation

[3]Multinomial Event model

Having markedly improved our accuracy for SVM, logistic regression, and Naive Bayes we revisited the *tf-idf* feature mapping to see if it similarly improved with the use of feature selection. Interestingly, retraining on all the algorithms (not including Naive Bayes) achieved the accuracy profile shown below.

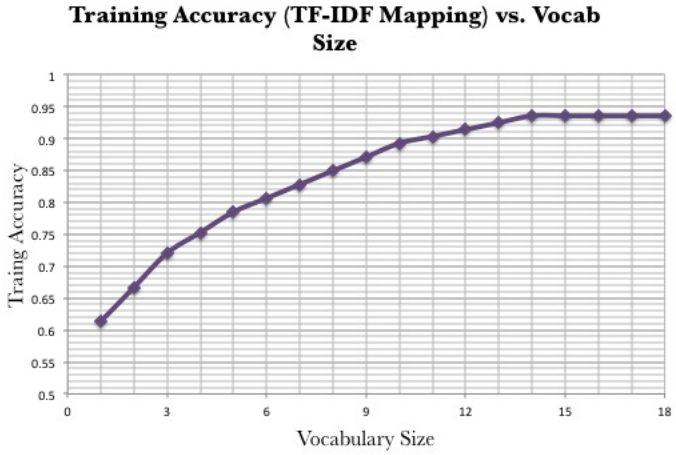

Fig. 5. Shows how the tf-idf feature mapping affects training accuracy. Interestingly, the curve was identical for the three SVM algorithms and regularized logistic regression

Overall the *tf-idf* feature mapping exhibited a similar increase in accuracy from the forward search but overall was not noticeably better than our initial feature mapping consisting of raw word counts. The general shape of the learning curve shows the same general trend as the SVM and logistic regression accuracies, flattening out after a little over a dozen words.

VI. Conclusions

The results of our project indicate that it is indeed possible to distinguish between songs composed by Paul McCartney and songs by John Lennon based only on song's lyrics. From our results we have concluded that Support Vector Machines yield the best performance for this binary classification problem. More important than the particular algorithm however, is choice of features. Through the course of this project we discovered that many of these algorithms will only perform well provided you have chosen a 'good' feature space in which to represent the data.

Additionally, much of the time spent preprocessing the data (i.e. lemmatization and removing 'stop' word) provided no noticeable performance increase as compared with selecting the proper features. The results of our project seem to suggest that much of the intuition that underlies the removal of so-called 'stop' words is not applicable in this setting as many of the words we suspected of being content free were in fact the highest precedence features with regard to the forward search feature selection.

Regarding future work, our binary classifier could in principle be expanded to a multi-class classifier. This multi-classifier could include all of the members of the Beatles: Paul McCartney, John Lennon, George Harrison and Ringo Star. A potential application of this classifier is the identification of the principal author of songs written by multiple individuals and for songs with disputed authorship. A further extension to the project would be to include audio data in the feature space or to examine if audio data alone (without lyrics) could be used to determine the author of a song.

References

[1] Mehran Sahami, Susan Damais, David Heckerman, Eric Horvitz. *A Bayesian Approach to Filtering Junk E-Mail*. 2014 Dec 3.

[2] Glen Fung, *The Disputed Federalist Papers: SVM Feature Selection via Concave Minimization*Available at `http://pages.cs.wisc.edu/ gfung/federalist.pdf` 2014 Dec 3.

[3] Fred. *THE BEATLES: Complete Lyrics of all Songs* Available at `http://www.gratuit-cours.com` 2014 Dec 3.

[4] Bird, Steven, Edward Loper and Ewan Klein (2009), *Natural Language Processing with Python. OReilly Media Inc.* 2014 Dec 3.

[5] John Cast, Chris Schulze, Ali Fauci *Music Genre Classification*. Available at `http://cs229.stanford.edu/proj2013`. 2014 Dec 3.

[6] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. *LIBLINEAR: A Library for Large Linear Classification, Journal of Machine Learning Research 9(2008)*, 1871-1874. Available at `http://www.csie.ntu.edu.tw/ cjlin/liblinear`

[7] Tommi Jaakkola, *course materials for 6.867 Machine Learning, Fall 2006*. MIT OpenCourseWare, Massachusetts Institute of Technology. Available at `http://ocw.mit.edu/` 2014 Dec 3.

[8] George Forman (2007), *Feature Selection for Text Classification* Available at `http://www.hpl.hp.com/techreports/2007`

[9] Dawen Liang, Haijie Gu, and Brendan O'Connor (2011)*Music Genre Classification with the Million Song Dataset* 2014 Dec 3.