

Beatles Song Authorship

Mark Burkart

Alec Strauss

Kait Walko

# 1 Introduction

The Beatles, a band more popular than Jesus, as some people say. This band hailed from England and took over the entire world during the 60s. Its member were John Lennon, Paul McCartney, George Harrison, and Ringo Starr. Famously, these members agreed to share authorship of all their songs, but who really wrote the songs?

John Lennon and Paul McCartney contributed roughly equally to the Beatles’ total song output. George Harrison was a significant but less prolific contributor, and Ringo contributed two songs and shared credit for a third. McCartney and Lennon shared song authorship in the album credits as ‘Lennon/McCartney’, with no distinction regarding the main composer. The two had agreed to share all songwriting credits years before the Beatles ever became famous1.

However, based on an interview done with Lennon in 1972, it is apparent that the two wrote many of the songs separately2. Based on that interview, the ‘main’ composer of most of the Beatles songs is generally known. McCartney generally agreed with Lennon’s recollections with a single exception, “In My Life”, which he credited to himself. The work of music historians has backed-up many of these assertions, but not without some controversy3.

Given that both composers wrote music and lyrics, would it be possible to predict which of the two composers was the main composer of any given song using only the lyrics? Are there certain words or phrases that are used more by one Beatles composer than the other?

Another question, aside from knowing who authored the lyrics of each song, would be what the sentiment of those songs were. Did John Lennon write more cheerful songs than Paul McCartney, or vice-versa? Would it be possible to predict the overall positive or negative sentiment of the songs? Could a computer algorithm read a song’s lyrics and label them as being positive or negative with the same ability as a human reader?

Finally, again using the song lyrics, would it be possible to computationally discover certain underlying themes or topics that are common in Beatles songs? Could songs be grouped or clustered together based on certain shared features?2 Analysis and Models

## 2.1 About the data

### 2.1.1 Data Sources

1. Two sources with main composer attributions for every Beatles song were compared:
   1. an interview with John Lennon from *Hit Parader* in 19722 <https://www.beatlefan.net/b208394-lennon-mccartney-who-wrote-what.html>
   2. an article in *Soundscape* in 1999 by Per Myrsten4 <http://myrsten.nu/worldnet/beatlesongs.htm> (Figure 1).
   3. Both sources agreed on all songs except for seven songs which Lennon attributed to both himself and McCartney whereas Myrsten attributed them to one artist or the other. In these cases, the Myrsten attribution was used. The song list from Myrsten was used as a ‘ground truth’ for the main composer labels for data preparation.
   4. This data was prepared as “beatles\_song\_list\_utf8.txt” (Table 1).
   5. Note - for additional evidence and to add confidence to the main composer label obtained from the preceding sources, song attributions from “Who Wrote the Beatles Songs: A History of Lennon-McCartney”, by Todd M. Compton3 <http://www.toddmcompton.com/beatlesongsbywriterprint.htm> were consulted to check the comparisons from the first two sources. This source generally agreed with the previous two sources. This source was not used as a source of data for the study.

Figure 1 Screenshot of song attributions from Soundscapes.info volume 2 October 1999, by Per Myrsten

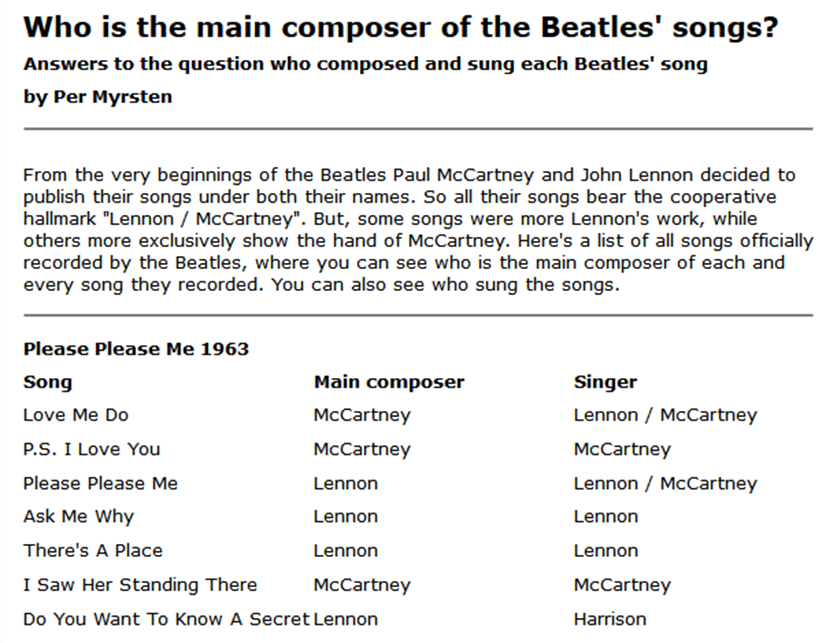
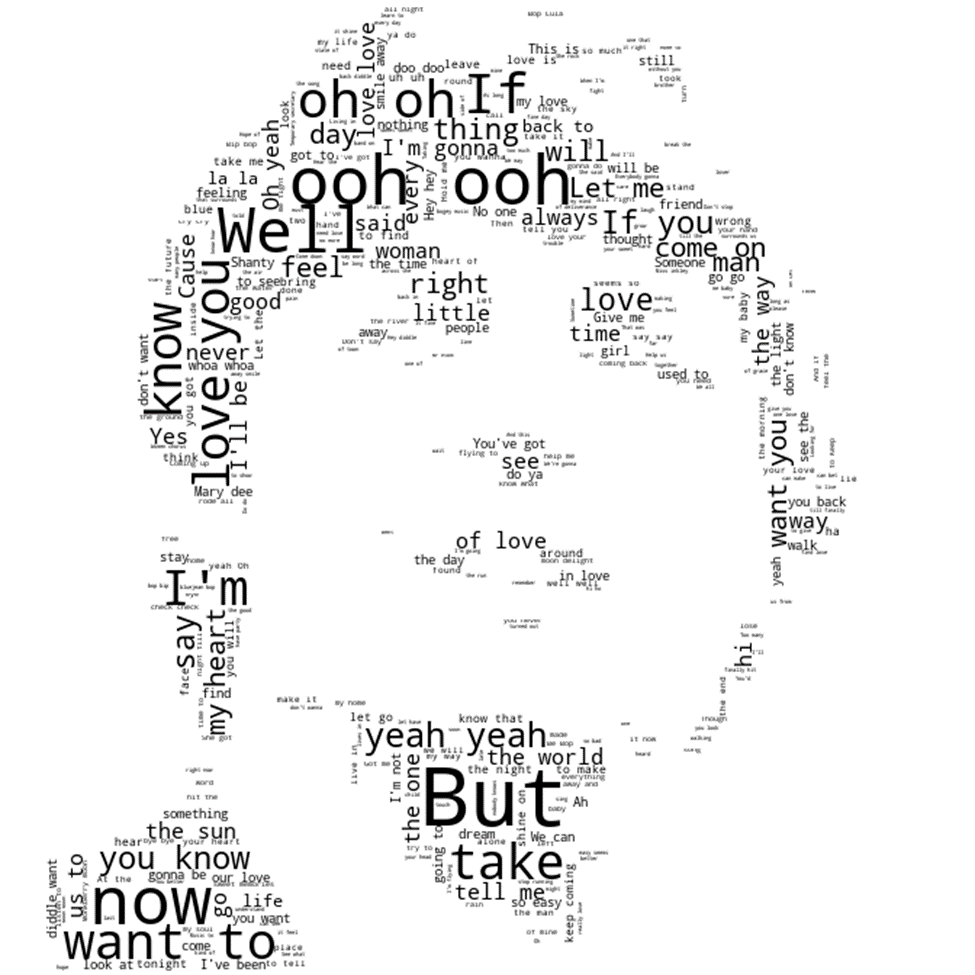


Table 1 beatles\_song\_list\_utf8.txt – sourced and prepared from Myrsten



1. API Queries – an API with song lyrics to popular music, Canarado-lyrics, was used to obtain song lyrics for all Beatles songs.
   1. registration at rapidapi.com was required to obtain an API key
   2. GET requests were created using the song names from ‘beatles\_song\_list\_utf8.txt’
   3. Endpoint: <https://canarado-lyrics.p.rapidapi.com/Lyrics/>
   4. Query example: <https://canarado-lyrics.p.rapidapi.com/Lyrics/Blackbird%2520the%2520beatles>
   5. Only relevant query results were kept, and special characters were removed
   6. All results were saved as ‘songsDF\_lyrics.txt’
2. Lyrics for John Lennon solo and Paul McCartney solo songs were extracted using the python package PyLyrics as ‘PM\_CR.csv’ and ‘JL\_CR.csv’.
   1. Duplicates were eliminated within each artist’s solo songs using manual editing
   2. Duplicates were eliminated so that there were no duplicates in each artist’s solos songs that were also in the Beatles songs.
      1. This was done using the fuzzy-matching python package Fuzzywuzzy
      2. <https://github.com/seatgeek/fuzzywuzzy>
      3. Any titles from the Lennon or McCartney solo songs having >= 70% similarity ratio to titles of any Beatles songs were manually examined to determine if they were the same song. If it was determined they were the same as a song in the Beatles songs, the song duplicate was removed from the solo material, as the duplicates were consistently Beatles songs mistakenly listed as solo work.



1. Positive/Negative sentiment scores were given to each song using the Python package VADERs described here:
   1. <https://github.com/cjhutto/vaderSentiment>
   2. The ‘Compound score’ from Vader results was used as the sentiment score. While ‘positive’, ‘negative’, and ‘neutral’ scores between 0 and 1 were available, the compound score conveniently places the positive-negative spectrum of scores in one metric:
      1. -1 = very negative
      2. 0 = completely neutral
      3. +1 = very positive
2. The following files were output with labels for both main song composer and song sentiment. An example of the beatles\_songs.txt file is shown in Table 2.
   1. beatles\_songs.txt
      1. Beatles songs
      2. This file was used as the **‘Beatles’** song catalogue for author attribution predictions
      3. Includes ‘Lennon’ and ‘McCartney’ labels for composer
   2. jl\_songs.txt and pm\_songs.txt
      1. Lennon and McCartney solo songs
      2. These files were combined and used as the **‘solo LM’** song catalogue for author attribution predictions
      3. Includes ‘Lennon’ and ‘McCartney’ labels for composer
   3. LM\_Beatles\_pos\_neg.csv
      1. Includes the top 20 highest and top 20 lowest songs by sentiment for Beatles work written by Lennon or McCartney
      2. This file was used as the **‘Beatles-SENT ’** song catalogue for song sentiment predictions
      3. Includes ‘P’ for positive and ‘N’ for negative sentiment
   4. LM\_solo\_pos\_neg.csv
      1. Includes top 20 highest and top 20 lowest songs by sentiment for Lennon
      2. Also includes top 20 highest and top 20 lowest songs with sentiment labels for McCartney solo work
      3. This file was used as the ‘**solo LM-SENT**’ song catalogue for song sentiment predictions
      4. Includes ‘P’ for positive and ‘N’ for negative sentiment

Table 2 screenshot of Excel view for beatles\_song\_list\_utf8.txt – composer label sourced from Myrsten4 and Compound sentiment score generated by the VADER python packages.



## 2.2 Analysis (models or techniques)

### 2.2.1 Preparation of the song catalogues for Author Prediction

The Beatles dataset was 209 songs, but after dropping songs whose main composer was either Lennon or McCartney this left 142 total songs in the Beatles catalogue. Within this catalogue there were 72 songs for which Lennon is considered the main author, and 70 songs for which McCartney is considered the main author.

For the solo works, there were 353 total Paul McCartney solo songs and 143 John Lennon solo songs. To use a roughly equal amount of Lennon and McCartney solo lyrics, the Paul McCartney solo songs were reduced to 143 by removing 210 songs at random. This left a total of 286 songs (143 Lennon, 143 McCartney) in the solo LM catalogue. There were no duplicate songs within or between the song catalogues. The ‘Beatles’ and ‘solo-LM’ song catalogues are shown in Table 3.

Table 3 Song catalogues used for author prediction



### 2.2.2 Preparation of the Lyrics corpuses for Author Prediction

Using the song catalogues above, three lyrics corpuses were created for training and testing author prediction models. Each corpus was randomly divided into training and testing samples. The train and test sets contained none of the same songs (Figure 2).

Figure 2 The three corpuses used to train and test author attribution predictive models. The number of songs from each composer used from each song catalogue is shown at the right of each table.



### 2.2.3 Preparation of the song catalogues for Sentiment Prediction

The ‘Beatles-SENT’ song catalogue was a subset of the ‘Beatles’ catalogue created by rank sorting the songs by VADER sentiment and selecting only the top 20-ranked and bottom 20-ranked songs by compound score. The top 20-ranked songs were given ‘positive’ labels and the bottom 20-ranked songs were given ‘negative’ labels.

The ‘solo LM-SENT’ song catalogue was a subset of the ‘solo LM’ catalogue created by first dividing the catalogue into two sets by main composer (Lennon or McCartney – 143 each) and rank sorting each set by the VADER composite sentiment score. For each of the two authors, the top 20-ranked songs were given ‘positive’ labels and the bottom 20-ranked songs were given ‘negative’ labels. The top 20 and bottom 20 ranked-songs from both the Lennon and McCartney halves were merged to create the ‘solo LM-SENT’ catalogue which consisted of 80 songs (40 positive labels, 40 negative labels). The ‘Beatles-SENT’ and ‘solo LM-SENT’ catalogues are shown in Table 4.

Table 4 Song lyrics corpuses used to construct the Sentiment Prediction Lyrics corpuses



The purpose of dividing the ‘solo LM-SENT’ lyrics corpus before rank-sorting them was to obtain an approximately equal ratio of Lennon and McCartney lyrics within the corpus. The VADER package would have given the same score to the songs irrespective of how they were divided, but by first dividing them and then selecting the top-20 and bottom-20 scored songs from each, an equal proportion of each author’s songs was assured.

The range of scores and labels for both catalogues, along with the average VADER sentiment score for all Beatles and Lennon and McCartney solo songs is shown in Table 5.

Table 5 Sentiment score ranges for labeled songs and average score for all songs in the Beatles and the solo LM catalogues.





### 2.2.4 Preparation of the Lyrics corpus for Sentiment Prediction

The lyrics corpus used for predicting song sentiment was created by combining the Beatles-SENT catalogue with the solo LM-SENT catalogue. The solo LM-SENT portion was used for model training and the Beatles-SENT was used for testing the model predictions. The train and test sets contained none of the same songs (Table 6).

Table 6 The corpus used to train and test song sentiment predictive models. The number of songs for each category is shown to the right of each category.



The reason that the solo LM-SENT catalogue (which consisted of Lennon and McCartney solo work) was used for training was because the solo LM catalogue contained a larger number of songs with highly positive and highly negative VADER scores relative to the Beatles catalogue. (This was not surprising since the solo LM catalogue contained 246 songs and the Beatles catalogue contained only 100 songs.) With highly positive and negative scores (close to +1 or -1) these songs could be labeled as ‘positive’ or ‘negative’ with greater certainty.

### 2.2.5 Vectorizers used to Construct Document Term Matrices

The data was tokenized and vectorized for all document term matrices in a single step using either CountVectorizer or CountTokenizer from scikit learn (Table 7).

Table 7 Vectorizers used for constructing document term matrices



Parameters for the vectorizers were adjusted to create different document term matrices. Separate DTM-matrices were created for each of the possible values in the range listed in Table 8.

Table 8 Parameter ranges used for vectorizers when constructing document term matrices. Any parameters not explicitly mentioned were left as default.



### 2.2.6 Naïve Bayes Classifiers used in Sentiment and Author Prediction

The Bernoulli naïve Bayes classifier and the multinomial naïve Bayes classifier from scikit learn were used to build models for sentiment and author predictions (Table 9).

Table 9 Naïve Bayes classifiers used for prediction models



### 2.2.7 Naïve Bayes Models used in Sentiment and Author Prediction

Models were created for a range of combinations of Naïve Bayes classifiers and document term matrices (Table 10).

Table 10 Naïve Bayes models. The Bernoulli classifier only used binary vectors, whereas the multinomial naïve Bayes (MNB) used both frequency and TF-IDF normalized vectors. The document term matrix parameters adjusted for each trial are highlighted in gold.



### 2.2.8 Support Vector Machine Classifiers used in Sentiment and Author Prediction

Linear SVC and SVC classifiers from scikit learn were used to build models for sentiment and author predictions. A linear SVM classifier was built using the LinearSVC class. Radial basis function (‘rbf’) and polynomial (‘poly’) were built using the SVC class. (Table 11).

Table 11 Support vector machine classifiers used for prediction models



### 2.2.9 Support Vector Models used in Sentiment and Author Prediction

Round 1 Selection Process for SVM models

To obtain an initial selection of models for three different types of SVM classifiers (linear, radial basis function, polynomial), SVMs with an input range of parameter settings (Table 12) and document term matrices (Table 13) were compared.

Table 12 Parameter ranges used with SVM classifiers. Any parameters not explicitly mentioned were left as default.



Table 13 Three document term matrices with the parameter settings shown below were used with each SVM classifier type.



Initial SVM selection was by best accuracy according to the trials shown in Table 14.

Table 14 Round 1 of trials of SVM models. The classifier parameters and document term matrices (‘Vectors’) that were adjusted for each trial are highlighted in gold.



Round 2 Selection Process for SVM models

The three most accurate SVM classifiers of each type (linear kernel, radial basis function kernel, polynomial kernel) obtained in round 1 of SVM testing were used. For each of these, new model variants were created using a range of classifier - DTM matrix combinations (Table 15).

Table 15 Support vector machine models. The document term matrix parameters adjusted for each trial are highlighted in gold.



2.2.10 K-Means Clustering Analysis

To prepare the data for k-means clustering analysis, first, the data itself is obtained. Lyrics for both John Lennon’s and Paul McCartney’s solo career song works are retrieved from an online repository of song lyrics for many different music artists, via a Python package called PyLyrics3. This package retrieves matching song information from its database of song titles and lyrics; users can retrieve song lyrics by querying within Python code for the desired artist(s). PyLyrics3 returns matching artists’ songs in the form of a Python dictionary, in which the song titles are the keys (equivalent to the word itself in a physical dictionary) and the lyrics are the values (equivalent to a word definition in a physical dictionary).

The lyrics fetched from PyLyrics3 overall tend to be quite clean, especially compared to text found through web crawling or Tweets. Nevertheless, minor corrections were made to the text via the steps next outlined. A label indicating who wrote which song is added: for John, a zero is added; for Paul, a 1. This is a vital step for model training as it will tell the model which person wrote which song.

To prepare the song lyrics, they are first saved to CSV format and then re-uploaded to Python as a dataframe. This allows for quick and easy editing and manipulation of the text itself.

For John Lennon’s solo songs, PyLyrics3 returned a repository of 175 songs. For Paul McCartney, however, this package returns over 400 different sings. Since Paul and John shared authorship of many songs, however, checking the song lyrics verified that there were some duplicates of songs within each song writer’s dataframe/repository of lyrics. To correct this, in Python, code is executed that checks for songs within John’s song lyrics that are in Paul’s song lyrics, and drops them from the dictionaries. This is repeated for the Beatles songs – thus ensuring that no songs are duplicated across the Beatles, John, and Paul song dataframes. This will avoid confusing models with repeated text and improve results.

Next, all three of the dataframes, John, Paul, and the Beatles, have their label information removed. The label information for these takes the form of a column in the lyrics dataframe that indicates who wrote each song. K-means clustering analysis does not need label information and in fact this might ruin any clustering results.

Then, each repository of songs is converted into a list of complete file paths. TFIDF Vectorizer and CountVectorizer each require as an input option content, or a list of files to vectorize. This is accomplished using a function that first retrieves in Python the lyrics for each song, saves the song to an individual text file -- which the song title is the name of the text file, and the lyrics form the body of the text file – and then builds a list of a complete file path for each individual song. These lists are saved to three separate variables, storing them for John, Paul, and the Beatles songs repositories separately.

Next, to give the k-means model something to cluster, Sentiment Analysis is performed on all three song dataframes. This is accomplished using the AFINN sentiment lexicon. Each of John’s, Paul’s, and the Beatles songs is scored using AFINN scoring. The resulting sentiment scores, which vary greatly per song, are plotted using Python’s histogram function, to show the distribution of song scores for each artist and the Beatles songs. This exploratory analysis gives a good indication of what might be expected from clustering results. The results are also plotted per song, which shows variations among John’s, Paul’s, and the Beatles’ song lyrics sentiments.

With this exploratory sentiment analysis done, the list of complete file paths are input into CountVectorizer and TFIDF Vectorizer. Both are used in this instance to have a chance to see how their results differ. A separate instance of each vectorizer is built and used for John, Paul, and the Beatles.

For vectorization, each song is converted to lowercase as per usual text mining standards, to help eliminate duplicate words caused by capitalization differences. Stop words are also removed to help give an idea of more important, rarer words. A custom stop word list is also added to the standard stop words list native to CountVectorizer and TFIDFVectorizer, as both of these functions vectorized the file names’ paths as part of the song, which is incorrect. The custom stop words list removed the file path words from vectorization, ensuring only the song lyrics themselves were vectorized.

Next, each list of vectorized words for Paul, John, and Beatles lyrics are converted to term document matrixes. This is an array (essentially a spreadsheet) which lists each word as a column, and counts its occurrence across all three song lyrics groups/dataframes. A list of each word present in all three song lyrics groups is built for each CountVectorizer and TFIDF vectorizer result.

After this step, the array of CountVectorizer words and TFIDF words and their TFIDF scores is built. This will feed into the model. For this clustering, TFIDF vectorizer results are used, as this should be a truer representation of which words are meaningful across the different songwriters’ lyrics, since TFIDF allows one to take into account and adjust for the length of each document in a corpus. Since some songs are much longer than others, this may be an important factor to consider.

Following this, the k-means model is instantiated. A separate k-means model is built for John, Paul, and the Beatles lyrics. Then, each k-means instance is fitted to each lyrics term document matrix.

This completes the k-means clustering steps except for visualizing the results. The results are plotted using a standard scatter plot function.

2.2.11 Decision Tree Model

Since decision trees tend to be a powerful tool for differentiating authorship among documents and corpuses, a decision tree model is built. For this model, John and Paul lyrics are combined into one dataframe, with a label column added. A zero label is added to each John song, and a one label is added to each Paul song. The Beatles songs lyrics, however, have no labels. For this model, the combined John and Paul lyrics serve as the training data, and the entire Beatles dataframe/repository of lyrics is the test data. The model should be able to identify key features of John and Paul labeled songs, and in theory, use what it learns to differentiate who wrote which song in the Beatles lyrics.

To begin the model, first, all of the song titles and lyrics in each separate lyrics dataframe is converted to string format. Then, each song and song lyric set are all converted to lowercase, again helping the model to avoid duplications of words caused by differences in capitalization. The label of who wrote which song is then dropped from the Beatles set to prepare it to be used as the test data.

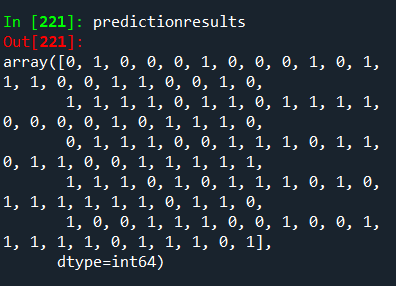
To give the model more of an idea what differentiates which song, sentiment results from the AFINN sentiment lexicon are then added to each dataframe for each song.

Next, the John and Paul AFINN-scored and labeled datasets are combined into one set. This creates the training data on which to train the model.

The AFINN score is set as the X variable in the Decision Tree model data; the label is the Y for the model. This shows the model that the label is what should be used to “train” itself.

Because, however, it isn’t really possible to feed text into certain Decision Tree models in Python’s DecisionTree classifier, Python’s SKLearn preprocessing module is imported, and its Label Encoder is used and applied to each X and Y dataset.

Finally, the Decision Tree classifier is loaded into Python. It is fed the X and Y data, which has been split into X training and test and Y training and test data. This split allows the model to first train on labeled data for which it has all of the “correct” label answers. Then, it can be used on test data for which it does not have a label answer and must determine which label belongs to which song.



The trained model is then fed the test data, in this case all of the Beatles songs, and then the results are obtained.

# 3 Results

## 3.1 Sentiment Exploratory analysis, topic modeling (LDA) and prediction

3.1.1 Sentiment Exploratory Analysis

The initial exploratory analysis began with sentiment visualizations, hoping to see something jump out that differentiated the authors.

These plots depict sentiment for Beatles albums John Lennon’s solo albums and Paul McCartney’s solo albums (Figure 3, 4, 5). For each album, the VADER-generated compound sentiment score for the most positive song on the album, the compound sentiment score for the most negative song on the album, and the mean compound sentiment score for the whole album are plotted. The albums are listed in chronological order.

Overall, the lyrics across the board were quite positive, but nothing really stood out as far as Lennon and McCartney having different writing styles. Paul McCartney’s solo work did have the highest average compound sentiment at about 0.65, with the Beatles next at 0.53, and Lennon was only a little bit behind with 0.52.

An interesting discovery was that only one out of 46 albums had a negative average sentiment and that was ‘Magical Mystery Tour’. The most positive albums for the Beatles, Lennon, and McCartney respectively were ‘Yellow Submarine’, ‘Rock ‘N’ Roll’ and ‘Ecce Cor Meum’.

The Beatles have said that Yellow Submarine was written for kids, so it was nice to see that the average sentiment was positive. Lennon’s Rock ‘N’ Roll, also having a very positive sentiment, was said to be his tribute to the rock genre of music and what it had done for his life. It made sense that he would be write positive songs about a subject that had offered him so many great opportunities. McCartney’s Ecce Cor Meum was one of his classical albums, and the lyrics are mostly in Latin. However, it does include English lyrics which are generally about love, peace, and happiness, so the positive sentiment score made sense in this light.

Figure 3 Beatles song sentiment for each album. The blue bar is the most positive song of the album, the orange bar is the average sentiment of the album, and the green bar is the most negative song on the album

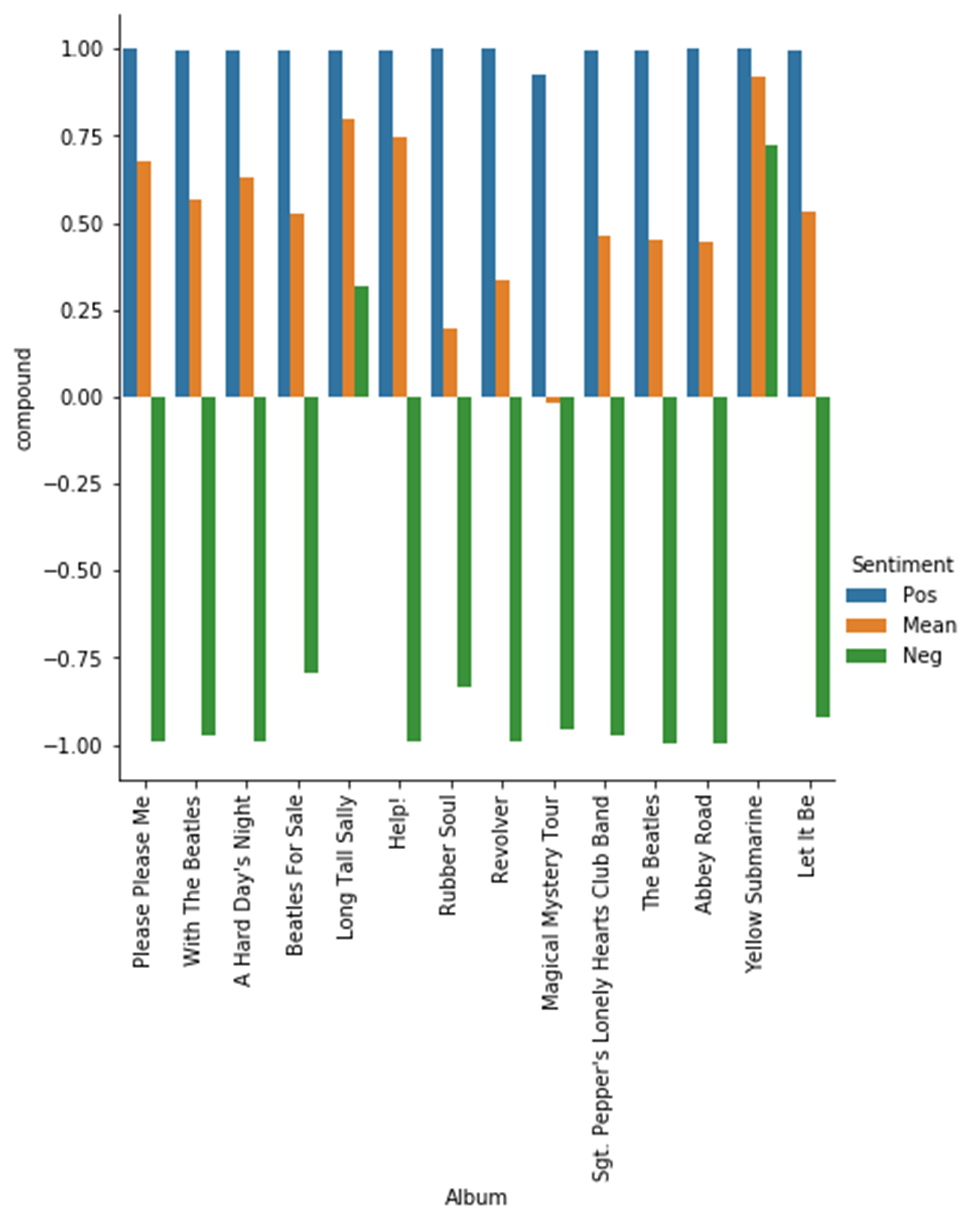


Figure 4 Lennon song sentiment for each album. The blue bar is the most positive song of the album, the orange bar is the average sentiment of the album, and the green bar is the most negative song on the album

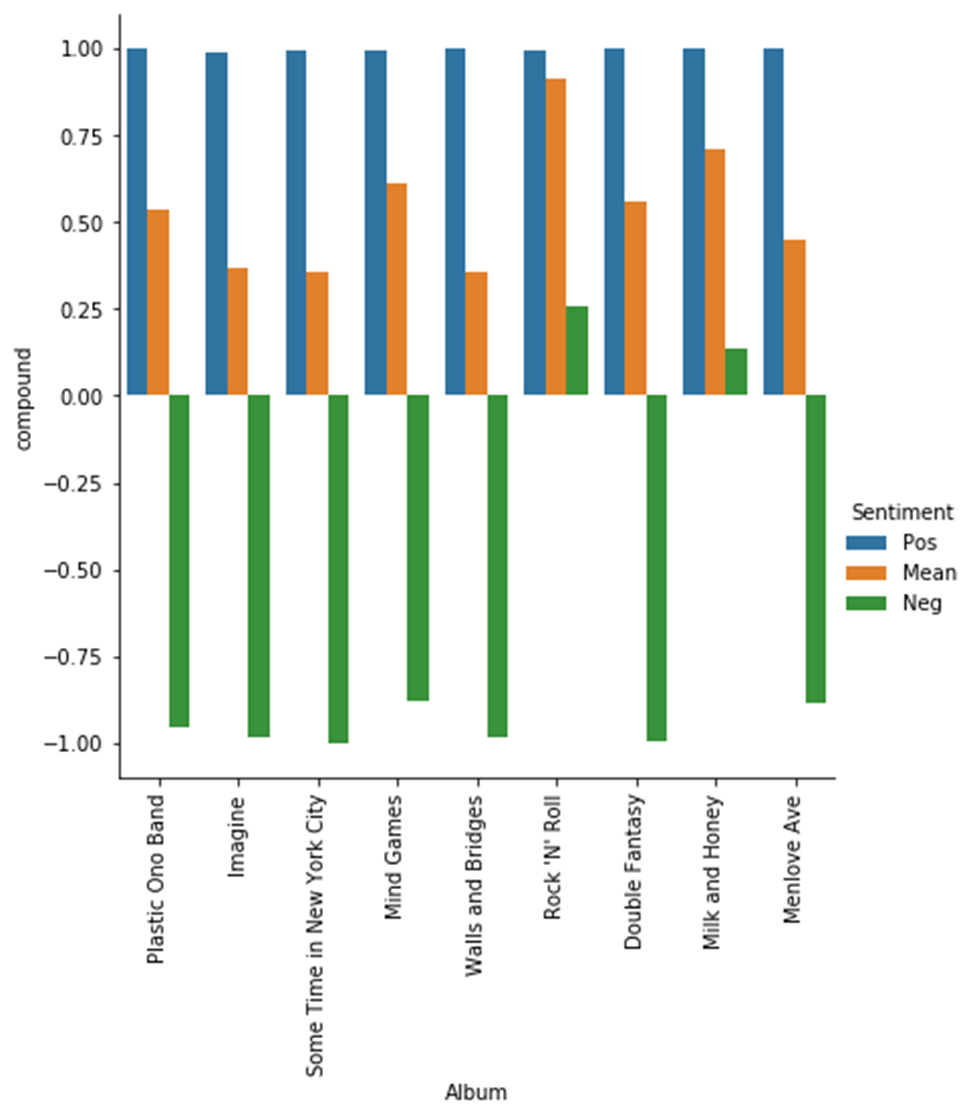
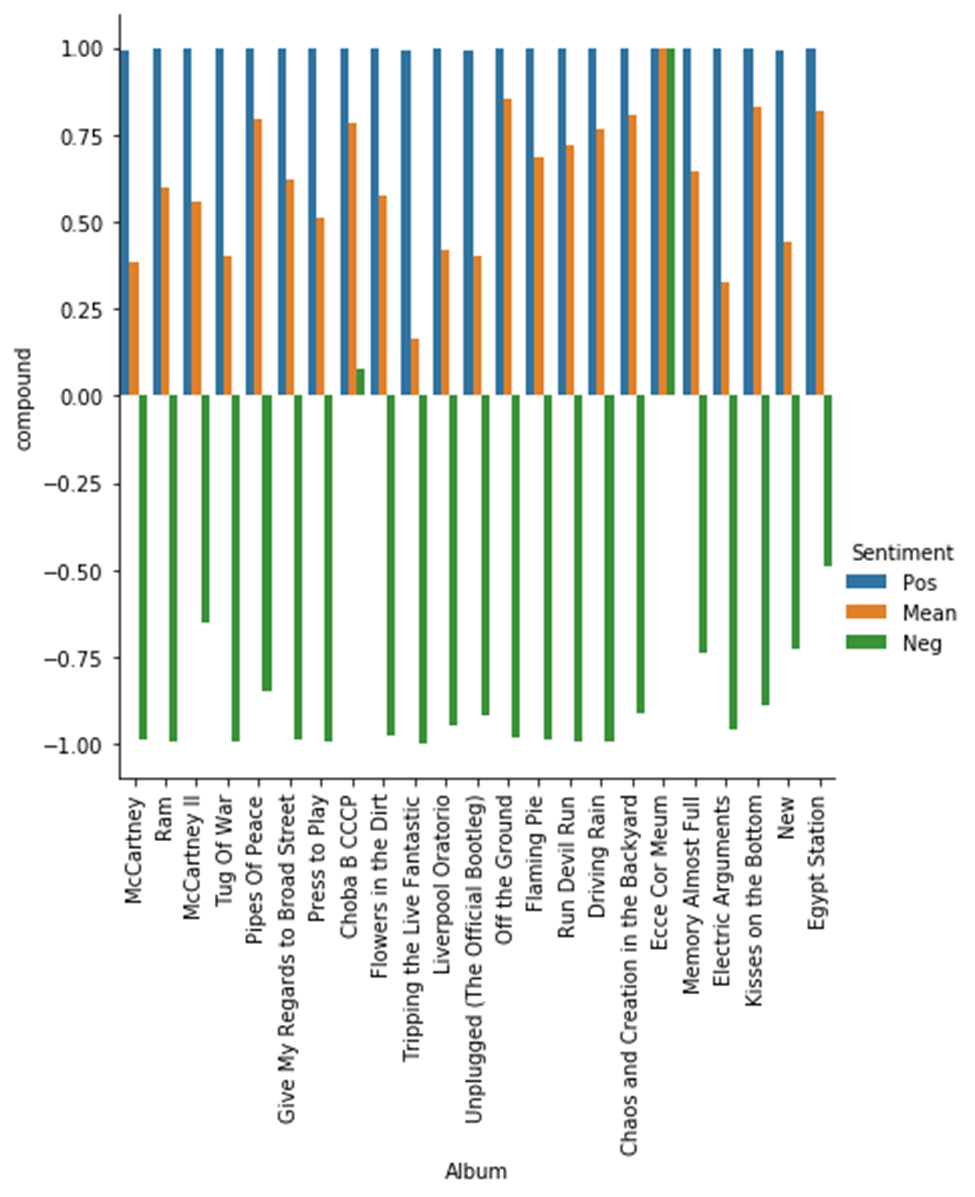


Figure 5 McCartney song sentiment for each album. The blue bar is the most positive song of the album, the orange bar is the average sentiment of the album, and the green bar is the most negative song on the album



3.1.2 Topic Modeling (LDA) Analysis

With that understanding of sentiment, the next thing to explore was the various topics of all the music. Overall, the topic modeling was not very informative, since there was never really a complete topic that made sense. Occasionally, a song title would pop up in a topic, but how this was related to other terms in the topic was unclear.

Some trends were notable, however. For McCartney, the word love seemed to be extremely popular, and this coincides with him having the highest average sentiment. For Lennon, terms related to the 1960s peace movement like peace, love, and beautiful, as well as negative events of the time such as the Bloody Sunday massacre of 1972 appeared in the results.

Figure 6 Paul McCartney did appear to sing about women a lot.

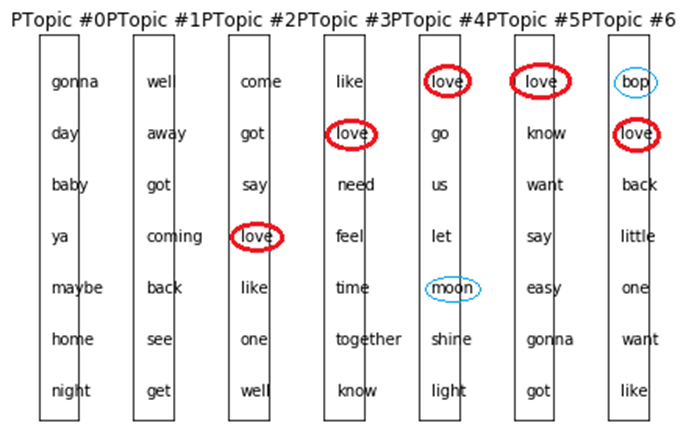


Figure 7 The Beatles sang about women a lot. Some song titles (Bungalow Bill, Yellow Submarine) in the results as terms.

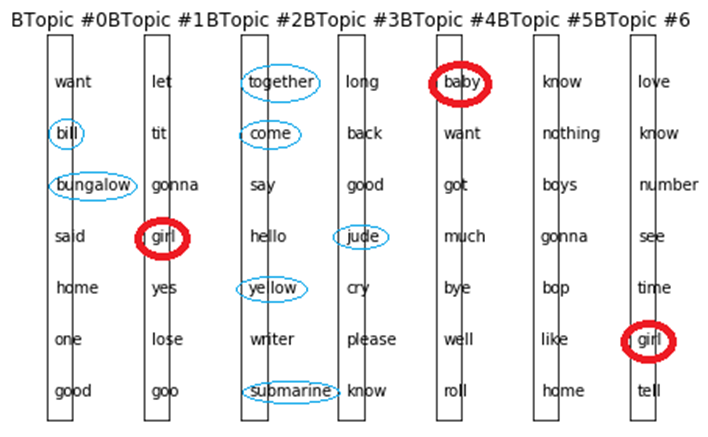
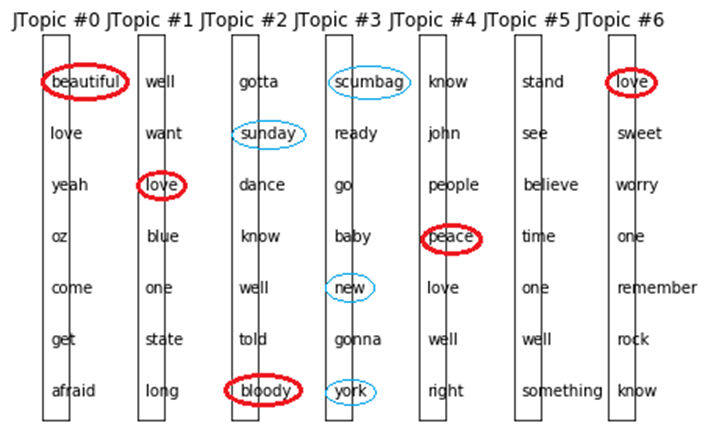


Figure 8: For Lennon, love appeared a lot but so did many negative words like scumbag, bloody, and worry appeared a lot.

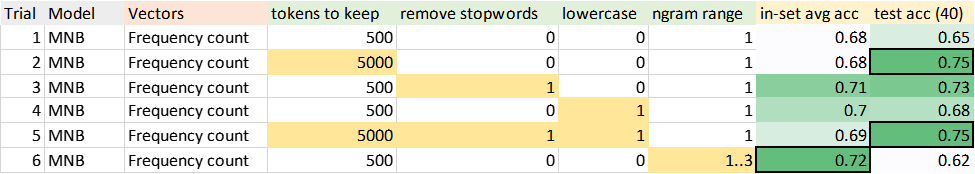


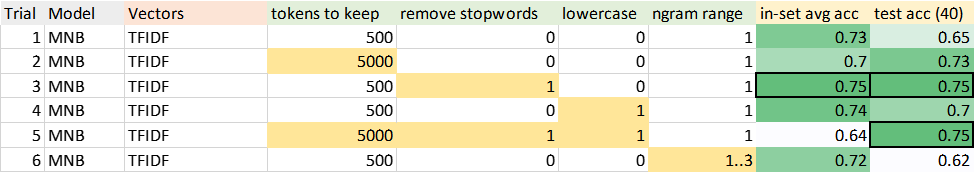
3.1.3 Prediction by Naïve Bayes Models

Term matrices generated with Binary, Frequency, or TFIDF vectors were fit to Bernoulli and Multinomial Naïve Bayes models. For each trial, one vectorizer parameter (tokens to keep, remove stop-words, use lowercase, or n-gram range) was changed. The in-set average accuracy (cross-validated) and test set accuracy for each trial are given in Table 16.

Table 16 For each trial, the optional parameter that was modified from the previous setting is highlighted in gold. The accuracies observed for each model and parameter setting is shown at right, with the highest accuracies (x-val and test) in bordered boxes.







The best accuracy scores for both cross-validated in-training sample accuracy and test sample were from the Bernoulli Naïve Bayes. This result was somewhat surprising since binary term matrices convey less complex information than frequency vectors or TFIDF vectors.

Another observation is that MNB with frequency and TFIDF vectors seemed to perform slightly better when stop-words were removed, and words were converted to lower case. For Bernoulli Naïve Bayes using binary vectors, however, this was not the case. The cross-validation accuracy was best for the unmodified term-document matrix when both the stop-words and upper-case letters were preserved. (There was a slight improvement in the test accuracy when words were converted to lower case).

3.1.3 Sentiment Prediction by SVM Models

Round 1 SVM Selection

SVM models were compared separately by kernal type (linear, rbf, poly) . For each trial, one SVM model parameter or vector count parameter (e.g., cost, gamma, exponent, vector type) was changed. The in-set average accuracy (cross-validated) and test set accuracy for each trial are given in Table 17.

Table 17 For each trial, the optional SVM parameter that was modified from the previous setting is highlighted in gold. The accuracies observed for each model and parameter setting is shown at right, with the highest accuracies (x-val and test) in bordered boxes







All models performed similarly. The best RBF-kernal models as judged by cross-validation used binary vectors. The best linear and poly models as judged by cross-validation were used TFIDF, but these did not do as well on the test set.

Additionally, C-penalty of 10 and 100 performed better than C-penalty of 1 for all models, Polynomials of 2nd degree performed best, and RBFs using scaled gamma worked best.

Round 2 SVM Selection

Next, as second round of testing was done using the best performing SVM models for each of the three kernel types. In this round of testing, the rest of the document-term matrix parameters were adjusted for each of the models.

Since binary vectors had worked best for the test set in the previous round of testing, those were used for all models in the second round. Additionally, a C-penalty of 10, a ‘scaled’ gamma = 1 / (n\_features \* X.var()), and 2nd degree polynomials were used.

The other parameters adjusted were tokens to keep, remove stop-words, set to lowercase, and n-gram range. The in-set average accuracy (cross-validated) and test set accuracy for each trial are given in Table 18.

Table 18 For each trial, the optional DTM parameter that was modified from the previous setting is highlighted in gold. The accuracies observed for each model and parameter setting are shown at right, with the highest accuracies (x-val and test)in bordered boxes







3.1.4 Sentiment Prediction Model Comparison

The best Naïve Bayes model and the best SVM were the Bernoulli Naïve Bayes and the SVM model with RBF-kernal. Both models used binary document-term vectors (Table 19, 20).

Table 19 SVM Confusion Matrix

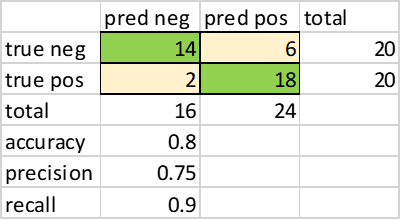


Table 20 Naïve Bayes confusion matrix



Both models used binary document-term vectors. A side-by-side comparison (Table 21) shows the SVM had better accuracy by +10%, better precision (for positives) by +19% and better recall (for negatives) by +25%. On the other hand, the Naïve Bayes had better recall (for positives) by +5% and better precision (for negatives) by +1% than the SVM.

Table 21 Results comparison of the best Naïve Bayes and SVM models

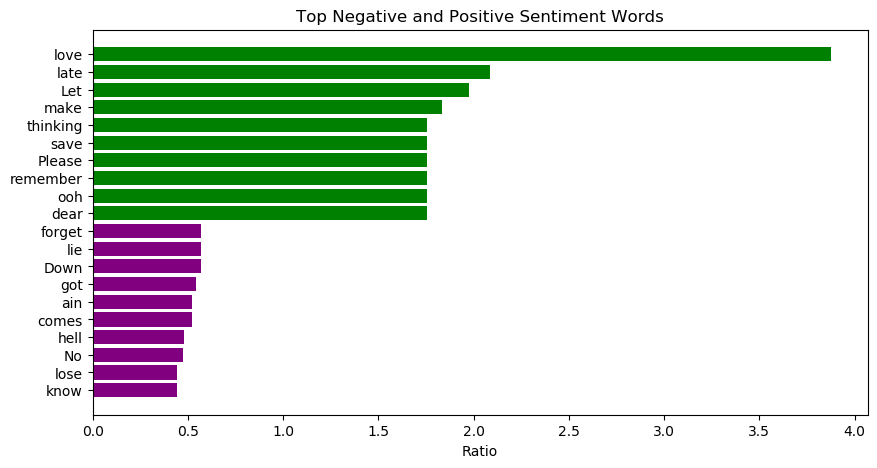


Additionally, most of the SVM models, after two rounds of tuning, performed slightly better than the Naïve Bayes models. An average test accuracy for the three best NB Models was 80% whereas for the three best SVM models the average test accuracy was 89%. The better results for the SVM models may be due simply to the fact that it handles the structure of this specific data set better. It may also be that it handles the general structure of song lyrics better, but this would require much more extensive testing to determine.

3.1.5 Sentiment Prediction - Indicative Terms

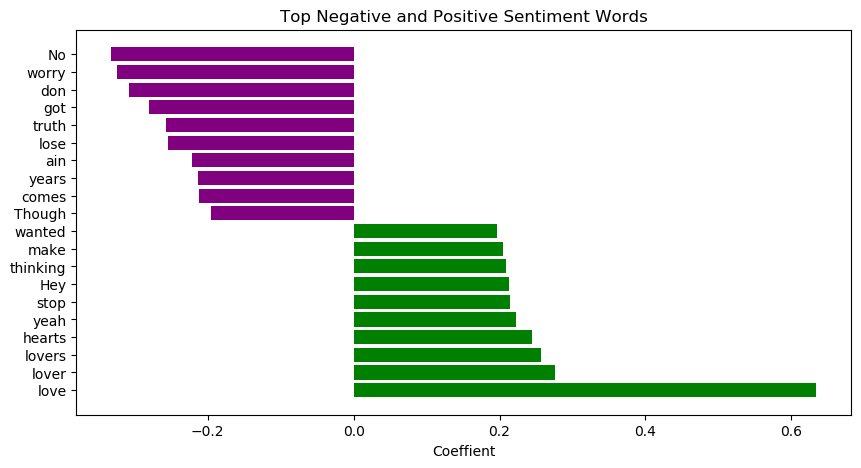
For the Naïve Bayes model, examining the words in the song lyrics for positive documents, the word ‘love’ is most indicative in the positive labeled documents, followed by ‘make’, ‘save’, ‘remember’, and ‘dear’. The words ‘know’, ‘lose’, ‘hell’, ‘ain’t’, and ‘comes’ are most indicative in negative documents (Figure 9).

Figure 9 Top 10 positive and negative sentiment words from Naïve Bayes by log ratio



For the SVM model, examining the words in the song lyrics for positive documents, the word ‘love’ is most indicative in the positive labeled documents, followed by ‘lover’, ‘hearts’, ‘supposed’, and ‘lovers’. The words ‘worry’, ‘truth’, ‘ain’t’, ‘got’, and ‘lose’ are most indicative in negative documents (Figure 10).

Figure 10 Top 10 positive and negative sentiment words by coefficient in linear SVM model

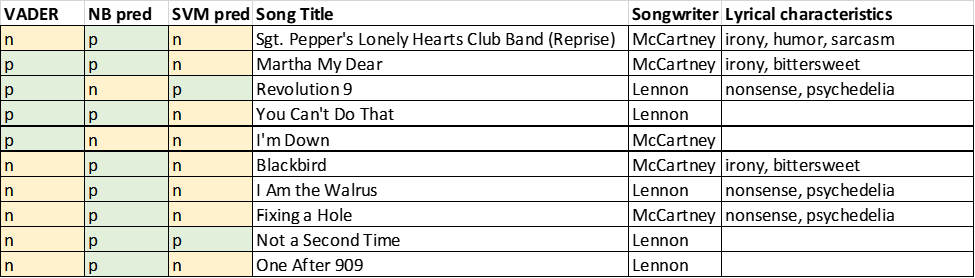


3.1.6 Sentiment Prediction Misclassifications

There were ten songs which were incorrectly predicted by either model or both models.

There is some grey-area when predicting song lyrics because these algorithms do not detect common lyrical devices such as sarcasm, irony, or bitter-sweet sentiment that are not easily placed on a two-dimensional spectrum of negative <> positive sentiment. (Table 22).

Table 22 Songs for which the human reader and VADER disagreed on sentiment label from within the ten songs ‘incorrectly’ predicted by either or both NB and SVM models.



Other errors in prediction resulted from certain decisions made when generating the document term matrix. For example, while the best SVM model performed better in general when stop-words were removed, one of the stop-words, ‘cry’, was actually a negative term that was removed from the matrix, thus the song was not predicted to be negative.

Similar errors may be generated by the assumptions made when generating the document-term matrix, but given the variety of possible choices, there is likely to be some amount of loss that occurs when optimizing matrix-model combinations.

## 3.2 Author Prediction

3.2.1 Author Prediction by Naïve Bayes Models

Term matrices generated with Binary, Frequency, or TFIDF vectors were fit to Bernoulli and Multinomial Naïve Bayes models. For each trial, one vectorizer parameter (tokens to keep, remove stop-words, use lowercase, or n-gram range) was changed. The in-set average accuracy (cross-validated) and test set accuracy for each trial are given in Table 23.

Table 23 For each trial, the optional parameter that was modified from the previous setting is highlighted in gold. The accuracies observed for each model and parameter setting is shown at right, with the highest accuracies (x-val and test) in bold and underline



The best accuracy scores for both cross-validated in-training sample accuracy and test sample were from the Bernoulli Naïve Bayes with the single exception of a multinomial naïve Bayes test score that was produced using TFIDF scores for the solo LM train -> predict soloLM test.

The cross-validation accuracy was generally better for the unmodified term-document matrix when both the stop-words and upper-case letters were preserved.

3.2.2 Author Prediction by SVM Models

Round 1 SVM Selection

SVM models were compared separately by kernal type (linear, rbf, poly) . For each trial, one SVM model parameter or vector count parameter (e.g., cost, gamma, exponent, vector type) was changed. The in-set average accuracy (cross-validated) and test set accuracy for each trial are given in Table 24.

Table 24 For each trial, the optional SVM parameter that was modified is highlighted in gold. The accuracies observed for each model/parameter setting are shown at right, with the highest accuracies (x-val and test) printed in bold and underlined







The linear-kernel models appeared to work best for Beatles predictions, the RBF-kernal models appeared best for solo predictions. Binary vectors worked best for all models.

Additionally, C-penalty of 1 was best for linear kernel, whereas a C-penalty of 10 worked best for RBF-kernels. and 100 performed better for polynomial. Polynomials of 2nd degree performed best, and RBFs using scaled gamma worked best.

Round 2 SVM Selection

Next, as second round of testing was done using the best performing SVM models for each of the three kernel types. In this round of testing, the rest of the document-term matrix parameters were adjusted for each of the models.

Since binary vectors had worked best for the test set in the previous round of testing, those were used for all models in the second round. A C-penalty of 1 was used for linear models. Additionally, a C-penalty of 10, a ‘scaled’ gamma = 1 / (n\_features \* X.var()), and 2nd degree polynomials were used for RBF and poly models.

The other parameters adjusted were tokens to keep, remove stop-words, set to lowercase, and n-gram range. The in-set average accuracy (cross-validated) and test set accuracy for each trial are given in Table 25.

Table 25 For each trial, the optional SVM parameter that was modified is highlighted in gold. The accuracies observed for each model/parameter setting are shown at right, with the highest accuracies (x-val and test) printed in bold and underlined



3.2.3 Author Prediction Model Comparison

Beatles predictions

Similarly to sentiment prediction, the models were the Bernoulli Naïve Bayes with binary document term matrix and the SVM model with RBF-kernel and binary document term matrix.

Table 26 SVM and Naïve Bayes Confusion Matrix (top SVM, bottom Naïve Bayes)





Table 27 Results comparison of the best Naïve Bayes and SVM models for Beatles songs predictions



Solo work (solo LM) predictions

For solo work predictions, the SVM model with RBF-kernel and binary document term matrix was again the best SVM model, but for Naïve Bayes it was the multinomial naïve Bayes using TFIDF vectors that provided the highest test prediction.

Table 28 SVM and Naïve Bayes Confusion Matrix (top SVM, bottom Naïve Bayes)





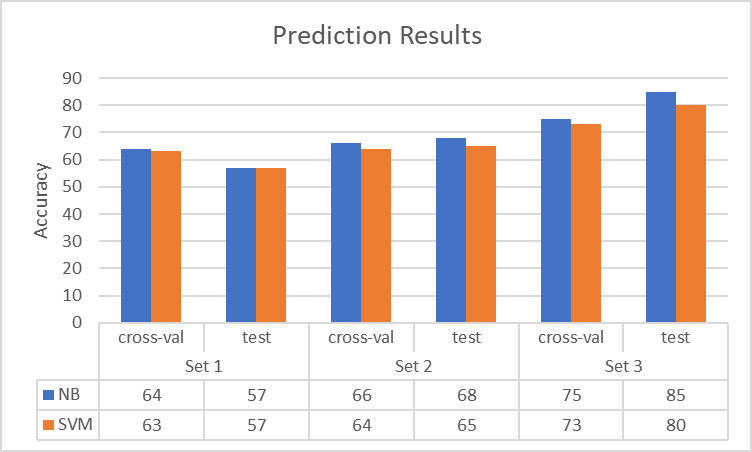
Table 29 Results comparison of the best Naïve Bayes and SVM models for Beatles songs predictions



Comparison of all prediction results

Overall, the best prediction scores were for training sets that included songs written by solo artists, and for test sets that also included songs written as solo artists (Solo-LM predict-> Solo-LM). Adding solo songs by Lennon and McCartney to the training set that included Beatles songs improved the predictions somewhat (Beatles+soloLM predict-> Beatles) (Figure 11).

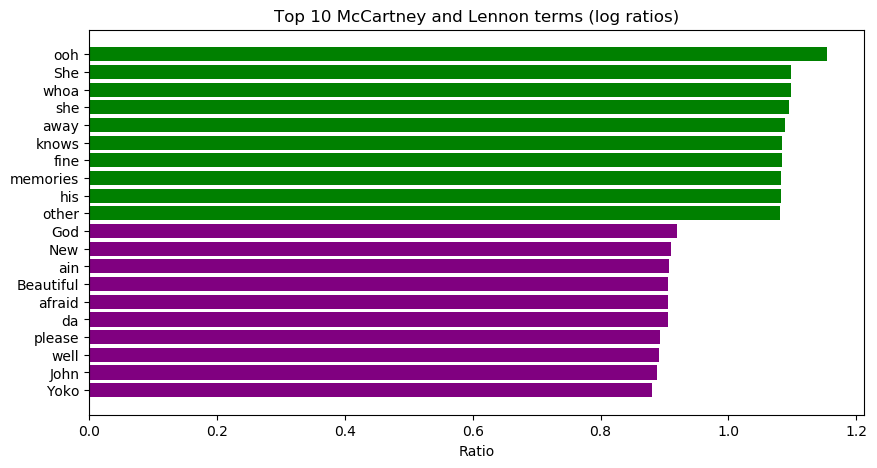
Figure 11 Comparison of best accuracies for cross-validated and test set predictions for each of the author prediction lyrics corpuses



3.2.4 Author Prediction - Indicative Terms

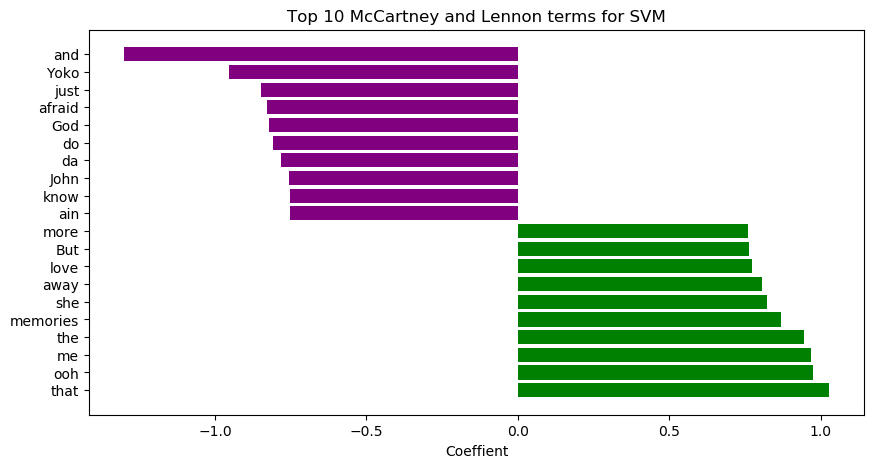
For the Naïve Bayes model, examining the words in the song lyrics for Lennon documents, the word ‘Yoko’ is most indicative, followed by ‘John’, ‘well’, and ‘please’. For McCartney, the words ‘Ooh’, ‘She’, ‘whoa’, ‘she’, and ‘away’ are most indicative (Figure 12).

Figure 12 Top 10 Lennon and McCartney words from Naïve Bayes by log ratio



For the SVM model, examining the words in the song lyrics for Lennon documents, the word ‘and’ is most indicative, followed by ‘Yoko’, ‘just’, and ‘afraid’. For McCartney, the words ‘that’, ‘ooh’, ‘me’, ‘the’, and ‘memories’ are most indicative (Figure 13).

Figure 13 Top 10 Lennon and McCartney words from SVM by coefficient



3.2.5 Author Prediction Misclassifications

There were songs that may have stronger predictability based on the lyrics. The probability difference for each label was used to sort the songs. Certain songs had a higher probability of being in one class or another (Table 30, 31).

Table 30 Naïve Bayes and SVM predictions for each Beatles song to belong to a given class. Naïve Bayes probability scores are included.



Table 31 Naïve Bayes and SVM predictions for each solo-LM song to belong to a given class. Naïve Bayes probability scores are included.



3.2.6 K-means clustering results

Figure 14: K-means results for Paul solo song lyrics:

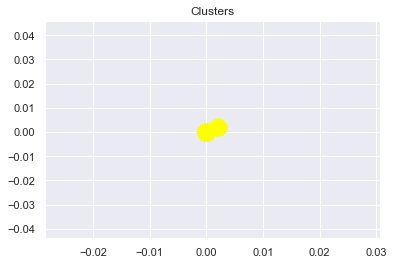


Figure 15: K-means results for John’s solo lyrics

A screenshot of a cell phone

Description automatically generated

Figure 16: Beatles (combined attribution to John and Paul): K-means results

A picture containing food

Description automatically generated

3.2.7 Decision Tree analysis results

For the Decision Tree classifier, results for classifying John and Paul lyrics follow:

Figure 17: Decision tree training data output results:

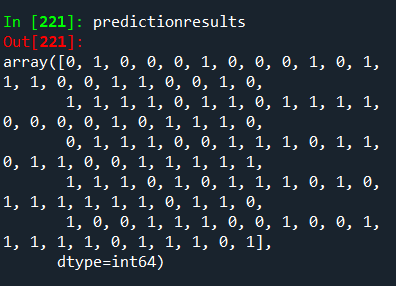
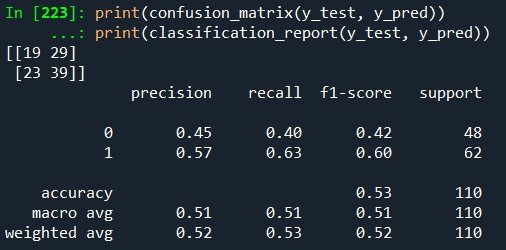
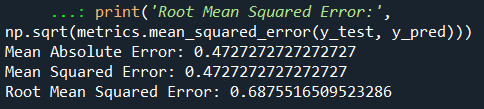


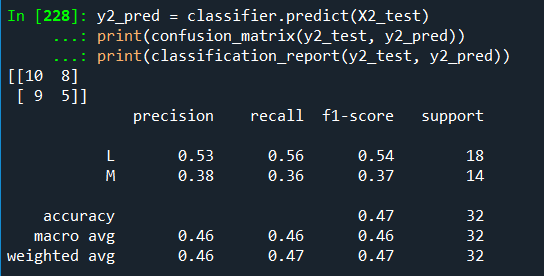
Figure 18: Confusion matrix for decision tree training data (Paul/John solo lyrics)





Next, the trained model is applied to Beatles lyrics:

Figure 19: Confusion matrix results for using trained decision tree classifier to predict John or Paul authorship of Beatles songs lyrics:



# Conclusions

Certain questions were asked at the beginning of this study. These questions were:

* Knowing *only* the lyrics of each song (and no other information about the music itself), would it be possible to predict which of the two composers was the main composer of any given song as determined by musicologists?
* Are there certain words or phrases used more by one Beatles than the other?
* Could further evidence of authorship be provided based on the lyrical elements alone?
* Are there words identified by topic modeling that are commonly used in topics related to each composer?
* What words or phrases does the algorithm use to determine if a song has positive or negative song sentiment?
* Does sentiment analysis show different overall sentiment for Lennon solo, McCartney solo, or Beatles songs?
* Could a computer algorithm to read a song’s lyrics and label them with author or sentiment with the same ability as a human reader?
* Does clustering of song lyrics provide any additional insight into the differences in songs written by Lennon and McCartney? What other insights does clustering song lyrics provide?

It was shown in this analysis that using only the lyrics of songs, it is in fact possible to predict which composer, Paul McCartney, or John Lennon, composed many of those songs. While it cannot be said that all the predictions could be made with perfect accuracy, the concept itself was convincingly demonstrated.

Moreover, it was shown that certain words or phrases were used more commonly by one Beatle composer or the other. Lennon’s lyrics are peppered with ‘Yoko’, ‘John’, ‘God’ and ‘beautiful’. McCartney’s lyrics contain a proliferation of emotive words like ‘ooh’ and ‘whoa’ as well as ‘she’ and ‘memories’.

Based on the analysis, some songs appear to have a *higher likelihood* of being written by one composer or another. These songs may be examples of songs where lyrics are more relevant to who composed the song. Showing the results of this analysis could provide additional evidence to Beatles music historians wishing to bolster their studies.

Topic modeling yielded some interesting results. While none of the artists had notably distinct topics amongst their own works, they did have a single overall theme that appeared. The Beatles sang about women a lot. Lennon sang about peace and love and had a lot of negativity within his topics. Lastly, McCartney sang about love significantly more than the other two.

It was also convincingly shown that positive and negative songs could be determined computationally. Song sentiment could be predicted with higher accuracy than author prediction, and when the algorithm was incorrect it was frequently due to a song being having ‘nonsensical’ lyrics or in some cases, subtle language elements like irony.

Moreover, it was shown that certain words or phrases were found to indicate positive and negative sentiment in Beatles and McCartney/Lennon solo work. These words included ‘Love’, ‘lover’, ‘lovers’, ‘hearts’, ‘yeah’ for positive words, and ‘No’, ‘worry’, ‘don’t’, ‘got’, and ‘truth’ for negative words.

Upon examining the different levels of sentiment alone, a difference amongst the groups can be seen. Paul McCartney is widely more positive than Lennon and the Beatles songs. Lennon and the Beatles however are nearly identical in their average sentiment.

Results from k-means clustering appear inconclusive so far. Further refinements and adjustments to the model will be necessary to fully plot out sentiment results for clustering each corpus. It does not appear based on the plotting results that sentiment can be used to distinguish John, Paul, and/or The Beatles lyrics jointly attributed to John and Paul, but this may be due to incorrect model set-up. Future clustering analysis should examine perhaps other attributes of their song lyrics, perhaps more advanced language processing techniques for example might generate better results.

For the decision tree analysis, surprisingly, the model appears to have difficulty distinguishing between Paul and John lyrics when it comes to who wrote which song in the Beatles songs. It does score higher when predicting John’s lyrics compared to Paul’s, which is an interesting result. However, when predicting who wrote which Beatles song, the decision tree model scores only slightly better than random chance. This is essentially equivalent to the baseline performance standard for this model, as there are only two classes from which to choose, which creates a probability of guessing who wrote which song at 50%.

This result shows that perhaps decision tree models may indeed have some promise for use as authorship attribution for Beatles songs; however, it may be necessary to use different features – perhaps deeper semantic structures, or, bi-grams (two words tokenized together) or tri-grams (three words tokenized together) to be truly able to distinguish John-written songs versus Paul-written songs.

# References

1. Sheff, David (2000). [*All We Are Saying: The Last Major Interview with John Lennon and Yoko Ono*](https://archive.org/details/allwearesayingla00lenn). New York: St. Martin's Press. *ISBN* *0-312-25464-4*.
2. <https://www.beatlefan.net/b208394-lennon-mccartney-who-wrote-what.html>
3. Todd M. Compton. “Who Wrote the Beatles Songs: A History of Lennon-McCartney” <http://www.toddmcompton.com/beatlesongsbywriterprint.htm>
4. <http://www.icce.rug.nl/~soundscapes/VOLUME02/Beatlesongs.shtml>
5. 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.
6. [www.beatlesbible.com](http://www.beatlesbible.com)