Midterm Report

Introduction

The overall objective of our project is to create a program that will use both features from the instrumental and features from the lyrics to create an overall sentiment analysis for a given song. However, for the current scope for our midterm report, we wanted to attempt to create an unsupervised model that will output an overall sentiment analysis for the lyrics only. Our project is motivated by the need to classify and gain insight on the characteristics of songs that we listen to everyday, whether it's for others to make use of these characteristics in products, or for individuals to gain knowledge about the songs they enjoy.

Methodology

For our data, we gathered the songs from the Billboard Top 100, and gained the lyrics as .txt files (66 files total, 147 KB) from the genius API. However, one issue we ran into was that some of the songs on Billboard Top 100 were not English songs, so we were not able to use them. Another issue we ran into was that when the song lyrics were generated into text files, some of the files did not contain the lyrics we were looking for; as a result, we were not able to use those files either. At the end of the data collection, we ended up with 66 usable text files that had song lyrics.

To clean the data, we started by excluding "stop words" and skipped lines that indicate song structure (ex: [Chorus], [Verse 2], etc.). This was to remove any words that would affect the analysis even though these words did not hold any sentiment or were irrelevant. We also replaced shortened word forms (ex: -in' turns into -ing) to make sure the shortened words did not affect the analysis. Finally, we removed punctuation and any other unnecessary characters that were present after API gathering.

Once the data was cleaned, we obtained 5680 unique words, and we were able to pass it into a pre-trained Word2vec model to create continuous-valued vectors out of the words that were passed in. Our result is a 5680 x 300 matrix, as each word from Word2vec is encoded with 300 dimensions. This program was crucial for our project because its specific context-based encoding situates each similar word similarly within the vector space. Our intuition is that due to this encoding, clustering analysis can isolate words of positive and negative sentiment, as we expect positive words to be least similar to negative words. This program was also beneficial for reducing the amount of dimension in our data. Each word is initially represented by millions of dimensions, encoded one-hot for each word in the vocabulary, but Word2vec allows the reduction of the number of dimensions (features) to 300 for each word. Although the number of dimensions is large, the program is effective in significantly reducing the dimensions and allowing context-based encoding. However, with dealing with 300 dimensions, visualizing our data and how words are situated in the vector space has proven to be difficult without significantly reducing necessary information within the vectors.

To evaluate sentiment using unsupervised methods, we make use of various clustering algorithms to partition sets of words within the vector space. The algorithms used were K-means, GMM, and DBSCAN. To assist in evaluation, we also made use of various sentiment classifiers to produce sentiment values of words within our dataset, providing a ground-truth comparison in our clustering.

Results

For our results, we expected similar words to be situated similarly (smaller distance) in their vector space, therefore in the same cluster. This would result in two or three clusters where one cluster represents positive sentiment, the other cluster represents negative sentiment, and an

optional third cluster that represents neutral sentiment. In general, we planned for the sentiment score for each word to be based on its assigned cluster and weighted based on its distance from the cluster center. Using sentiment values for each word, we can take the lyrics of each song, find the total sentiment, and average the total sentiment using the amount of words of each song to gain an overall sentiment of the songs. The three models that were used for training are K-means, GMM, and DBSCAN.

For K-means, we were able to assign sentiment to a single word based on Euclidean distance to a cluster center, and to obtain polar sentiment of a word, we make use of two clusters to represent positive and negative sentiment and the distance of the word from its cluster center to weight this sentiment. To evaluate our clustering, we used the sentiment values obtained from various sentiment classifiers to determine ground truth assignment. Ground truth clustering will assign words to clusters based on whether the word sentiment is more positive or negative. Without results from K-means, Various evaluation metrics were derived from the confusion matrix, but they indicate poor precision and accuracy. This means that in the context of being compared with other sentiment classifiers, our model generally fails to make positive predictions accurately, and correct predictions are not likely significant. Our recall score is good, though. This indicates that the model returns most of the relevant results. Our F1 score is ok, but not indicative that our model was very successful

	T0	T1
C0	1041 (TP)	830 (FP)
C1	147 (FN)	146 (TN)

Precision = TP / (TP + FP) = 1041/1871 = 0.5564
Recall = TP / (TP + FN) = 1041/ 1188 = 0.8763
Accuracy = TP + TN / (Total) = 1187/2164 = 0.5485
F1 Score: 0.6806

For GMM, it allowed us to assign a probability based on vicinity to the polar sentiment cluster, giving more details to a specific word clustering rather than assigning it into one of two discrete categories. The ground truth clustering process was identical to the process used for our K-means implementation. The sklearn GMM implementation was used and covariance matrices were set to full to allow for hyperellipsoid Gaussians.

From our GMM results, the evaluation metrics analyzed were the same as those analyzed for K-means. Precision, recall, accuracy, and f1 score were all low and lower than those for the K-means implementation. The final probabilities for all words in the current implementation are either <0.00001 or >0.99999, with no words in between. This is likely due to using only 2 components in the implementation and using 300 dimension vectors, resulting in very low covariances and a model with little use.

	T0	T1
C0	539 (TP)	692 (FP)
C1	649 (FN)	284 (TN)

Precision = TP / (TP + FP) = 539/1231 = 0.4379
Recall = TP / (TP + FN) = 539/ 1188 = 0.4537
Accuracy = TP + TN / (Total) = 823/2164 = 0.3803
F1 Score: 0.4457

With our attempt with DBSCAN, the goal of gaining two or three clusters to represent polar sentiment, with noise words possibly representing more neutral words. When clusters are determined through DBSCAN, we don't know if clusters will correspond with sentiment, or what clusters might represent without observing the results intuitively. Since we are focusing on polar sentiment gathering from clustering, DBSCAN might not provide the best algorithm to find significant results

As for our results, running DBSCAN with eps equal to 3 and minPts equal to 2, our result is 93 separate word clusters. Analyzing each cluster and all the words it contains manually would

be time-consuming, and upon a first glance, words within a cluster don't necessarily correspond to one similar topic or subject. Increasing minPts allows for our cluster count to reduce (ex: minPts = 8 reduces to 5 clusters), but most points are assigned the same cluster. Since our goal is to indicate polar sentiment, from the intuition of having a goal of two or three clusters to represent distance from a 'positivity', 'negativity', or a possible 'neutral' center, DBSCAN to detect noise and arbitrary cluster 'shapes' in the vector space likely won't provide anything significant with further adjusting of parameters. To further analyze this, we could use the elbow method with the minPts nearest neighbor to choose the most optimal eps value.

Discussion

Overall, there were various aspects that we believe limited our ability to not produce the results we wanted in this half of the project. One of these aspects was how our current method for analyzing sentiment using clustering is seemingly ineffective for K-means. Upon further research, Word2vec might produce better results by using cosine distance instead of Euclidean distance to determine clustering results. Also, our clusters might represent sentiment better if we provide a greater amount of unique words to our lyric vocabulary, so we will be increasing the sample size and ensuring that the data is appropriate for use. Another aspect that limited us was the application of other algorithms such as DBSCAN and GMM to our word vectors that have also resulted in similarly insignificant results. Finally, Since we were dealing with data in 300 dimensions, visualizing the data was proven to be difficult without eliminating significant information, and applying internal evaluation metrics and dimensionality reduction on such high dimensional vectors is also difficult for similar reasons.

In the future, our current plan is to make use of derived sentiment values with a trained model that uses these sentiment values as target variables. Also, our clustering results are

proving to be insignificant and against our intuitions, but further effort into our unsupervised methods might yield more significant results. When training, we will implement additional features by including Spotify characteristics of each song. This will allow us to incorporate additional features that might assist in correlating a song to its polar sentiment. With the data, we will include a larger sample size that only includes English. This will allow us to properly train our models. Additionally, applying our algorithms with a greater number of clusters may yield more promising results.