Music, Lyrics, and Mood

Vilas Winstein
winstein.1@osu.edu

James Enouen
enouen.8@osu.edu

Nate Gerjets
gerjets.2@osu.edu

Sarah Flanagan
flanagan.177@osu.edu

***Abstract*—This project explores the problem of classifying a song by emotion based on its lyrics. A Naïve Bayes model is used as a baseline for comparison, and the conditional probabilities it generates are used to determine the importance of individual words. This data is then used to create more advanced classifiers, including an adapted Naïve Bayes model and a Neural Network.**

Classifier Method	Accuracy
Naïve Bayes Model	**36.80%**
Adapted Naïve Bayes	**38.95%**
Mood-Balanced Naïve Bayes	**42.40%**
Deep Neural Network	**47.03%**

***Index Terms*—music, mood, sentiment analysis, artificial intelligence, classification, word embeddings.**

I. Introduction

Music plays an important role in culture and society, both as a method of entertainment and a mode of communication. Different songs can evoke different emotions, and the range of meaning and feeling that can be captured in a single song is immense. Picking out the sentiment of a song is relatively easy for humans. Our goal is to pin down exactly what gives a song the emotion it conveys, thereby shedding some light on the process of capturing emotion through song. Specifically, we aim to build a classifier that can determine the mood of a song, given its lyrics.

In general, emotion is difficult to model or classify, but one simplistic model introduced by Posner et. al. [2] focuses on just two aspects of emotion: valence (a measure of positivity) and arousal (a measure of energy). In its simplest form, this model asserts the existence of four main emotions: happy (positive valence and positive arousal), relaxed (positive valence and negative arousal), angry (negative valence and positive arousal), and sad (negative valence and negative arousal). This model is easy to encode in a computer due to its simplicity, and is also easy to visualize (Fig. 1). Thus, it is potentially easy for some machine-learned classifier to categorize songs according to these four moods.

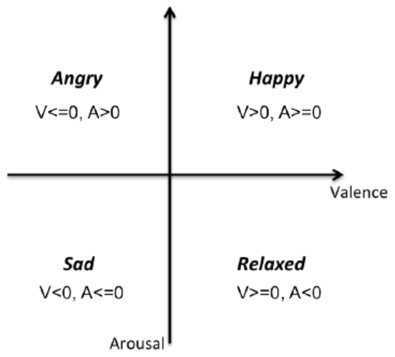


Fig. 1. Valence/Arousal model of emotion.

II. Problem Specifics

A. Data Set

For this project, we used the "Music Mood Classification Dataset" provided by Xue et. al. [4], which has lyric files of popular songs tagged with their emotion in the valence/arousal model. The data set contains about 400 training data points with about 100 songs in each of the four moods. The data set also contains about 400 testing data points, also split equally among the four moods. Each data point consists of the lyrics of a song and the mood that the song represents. For our work in this project, we simply transformed these lyrics into a "bag-of-words" which only keeps track of counts for each word in a song and forgets about the order of the words. In the entire training data set, there are 6138 unique words, so each song could be represented by a 6138-dimensional word count vector.

B. Problem Formulation

Our task is to build a classifier, using various machine-learning techniques, that can correctly classify the tagged mood of a song in the data set discussed above. The main performance measure of such a classifier is simply classification accuracy, which is the number of correct classifications over the total number of testing data entries. The model of emotions we are using allows us to investigate the accuracy of a classifier in terms of valence and arousal as well. When calculating the valence accuracy, we consider the classification to be a success if it simply guessed the correct valence. For example, if the classifier guessed that a "Sad" song was "Angry" instead, this is still counted as a success, since both emotions have the same valence. The arousal accuracy is computed in an analogous manner.

C. Models and Methods

We chose to start by creating a Naïve Bayes classifier for our task. This is because this classifier is relatively easy to construct, and can often provide a surprisingly high accuracy in many situations. In addition to this, a Naïve Bayes classifier is a good baseline to compare our later (and more advanced) classifiers against. For instance, if a more advanced classifier performs worse than the Naïve Bayes model, then that model of classification is clearly not suited to the task of classifying songs by mood. On the other hand, a classifier that beats a Naïve Bayes model must be worthwhile, and can perhaps shed light on key factors that determine the mood of a song.

Another reason for starting with a Naïve Bayes classifier is that such a classifier yields a conditional probability table. This can be analyzed to find out which words are "important" in the sense that they are a good predictor of mood. By restricting the dimensionality of the data set by only considering these most important words, we were able to eliminate useless information, which speeds up classifiers and can also provide an improvement to their accuracy.

Specifically, we compared the performance of a Naïve Bayes classifier which uses this restricted-dimension data to the original Naïve Bayes classifier. There were also some improvements to the way that important words were selected, and this resulted in a third implementation of the Naïve Bayes model. Finally, we created a deep neural network which operates on vectors obtained by converting each song into a vector using the restricted-dimension data and a Word2Vec model [1].

III. Naïve Bayes Model

A. Implementation of a Naïve Bayes Classifier

The first Naïve Bayes classifier we created uses the occurrence or non-occurrence of words in a song as the features of the song. Specifically, each word that appears in any song in the training data set is used as a feature. Songs are converted to feature vectors, where the value corresponding to a particular feature (a word) is 0 if the word doesn't appear in the song, and 1 if the word does appear. Then a conditional probability table is constructed, storing the values

$$P\left(\text{the word } w \text{ is in } S \mid S \text{ has mood } m\right)$$

for each pair (w, m) of a word and a mood (here S denotes a song). Laplace smoothing, as suggested by Provost et. al. [3], is then used to remove zero probabilities to avoid immediate rejection of certain mood possibilities due to a very large vocabulary. Finally, in order to classify a test data element, Bayes' rule is used to compute the conditional probability of each mood, given which words appeared (or did not appear) in the song, and the mood with the highest probability is returned.

B. Results of Naïve Bayes Classifier

The Naïve Bayes classifier performed better than randomly guessing. The overall accuracy of this classifier was 36.80%, whereas one would expect a "random guessing" algorithm to achieve an accuracy of 25% since there are four moods to choose from. The valence accuracy of the Naïve Bayes classifier was 51.47%, which is only barely greater than 50%. Needless to say, this classifier cannot tell good from bad. However, the arousal accuracy of this classifier was more impressive at 69.07%. This indicates that the intensity of emotion is more readily apparent from the lyrics of a song when using this particular method of classification.

C. Important Words

The conditional probability table used in the Naïve Bayes model contains information that can be used to determine which words are more useful than others. Specifically, using Bayes' rule, for each fixed word w, one can determine the values, as m varies across the four moods, of

$$P\left(S \text{ has mood } m \mid \text{the word } w \text{ is in } S\right),$$

and then determine the entropy of the resulting probability distribution (with moods m as the possible events). Words which lead to a higher entropy in this calculation give less information: they appear in similar numbers of songs across all moods. By the same token, words with lower entropy give more information. Below are some snippets of the list of words ranked in order of lowest entropy:

Rank	Word	Entropy
1	fuck	0.514995
2	ass	0.838099
3	shit	0.856571
4	motherfucker	0.886686
5	fuckin	0.886686
6	fucking	0.896392
7	bitch	0.917232
8	fact	0.938280
9	drown	0.940450
10	eminem	0.941549

Rank	Word	Entropy
6101	for	1.385540
6102	it	1.385544
6103	be	1.385657
6104	im	1.385788
6105	say	1.385819
6106	and	1.385967
6107	the	1.386026
6108	in	1.386117
6109	i	1.386123
6110	you	1.386139

This shows a few of the most influential words and a few of the least influential words. The least influential words are those which appear in similar numbers of songs across all moods. As can be seen in the table, this includes many common words like 'the' and 'it,' which are not helpful in determining the mood of a song because most songs include words like this, regardless of mood. On the other hand, words like 'fuck,' 'ass,' and 'shit' are very good indicators of the mood of a song. The word 'fuck' in particular has much lower entropy than any other word on the list by a large margin, and variants of 'fuck' (such as 'motherfucker' and 'fucking') are also very high on the list of good indicator words.

One can also notice that almost all of the lowest-entropy words are negative and intense, which hints that they represent anger. When we took a closer look at the data, we found that all of the top ten most indicative words appeared primarily in angry songs, except for 'drown' which appeared in sad songs. This indicates that the "angry" mood stands out significantly more than the other moods and is easier to classify.

IV. Improvements to Naïve Bayes Models

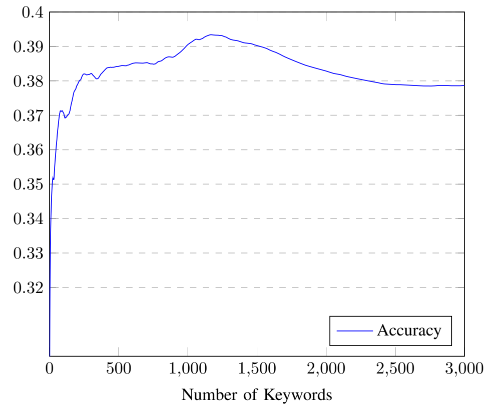
A. Restricting the Word Set for the Bayesian Network Model

The original Naïve Bayes model utilizes every words in the training set and uses Laplace smoothing to mitigate the effects of extremely rare words. This method still focuses on overly commonplace words such as 'the,' 'and,' and 'it,' which are not useful in classifying the mood of a song. In order to improve the Naïve Bayes model, we discard the words which are not important for classification, considering only the words with high importance. We introduce a hyperparameter, n, which is the number of important words to consider from each song. This time, the Naïve Bayes classifier considers a song to be just its n most important words, but otherwise the classification algorithm is the same as before. One challenge for this model lies in finding the correct number of words to use. If too few words are used, the classifier will not have enough information to make a reasonable prediction, whereas if too many words are used, the classifier will be very similar to the original Naïve Bayes model.

B. Results of This Restriction

The overall accuracy of this model was calculated on a validation set. Graph 1 below depicts the accuracy as a function of the hyperparameter n, and it shows a clear maximum.

Graph 1: Accuracy by Number of Keywords



The optimal value for n, the number of words used, was calculated to be 1167 on this validation set. Using the 1167 most important words, the altered Naïve Bayes model then scored a test accuracy of 38.95% with a valence accuracy of 55.80% and an arousal accuracy of 65.79%. This slightly altered model was able to improve the accuracy of the Naïve Bayes classifer, but it still follows the trend that the arousal of a song is easier to predict than the valence.

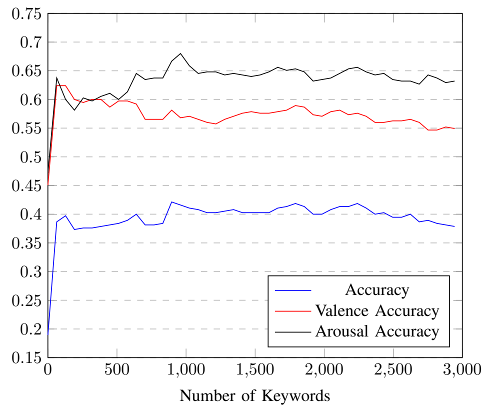
C. Mood-Balanced Keywords

As discussed earlier, the words with the lowest entropy are mostly 'angry' in our data, which skews the model to favor choosing 'angry' as the mood of the songs. As a solution to this problem, attempted to ensure similar representation of each mood in the list of indicative words. To do this, we again started by ordering the list of words by entropy and taking a sampling from the beginning of this list. This time, while selecting the words from the list, we skipped a word if the mood it conveyed was already represented in at least $\frac{n}{4}$ words, where n is the desired number of important words.

D. Results of Mood-Balancing

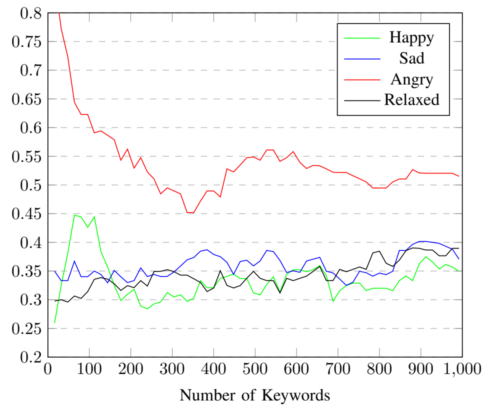
We sampled the data once again with different numbers of important words and found the maximum accuracy to be 42.40%, an increase of 3.45% from the previous model. These results can be seen in Graph 2.

Graph 2: Mood-Balanced Keyword Accuracies



The results were also highly dependent on the number of indicative words used to run the model. Just by adding one more word, the results sometimes improved, and sometimes became more biased towards the mood of the added word. Although the words are now mood-balanced, because the entropies for the 'angry' indicator words were still significantly lower than the other moods' indicator words, 'angry' remained the mood with the top precision, as seen in Graph 3.

Graph 3: Precision by Mood



V. Neural Network Model

A. Implementation

This approach uses a mixture of techniques to embed a song into a high-dimensional vector space, and then the resulting vector is fed into a neural network. As with previous models, this model prioritizes words with low entropy, using a hyperparameter m as the number of words selected from each song. The first step in pre-processing is to take the m words of lowest entropy from a song's bag of words (taking repeated words up to three times). This list of m words is then converted into m 300-dimensional vectors using Google's pre-trained model for Word2Vec, which embeds a word in a 300-dimensional space. [1].

These m vectors are what the neural network takes as input. Let f the size of the feature vector extracted from each word, which is the number of neurons per word in the second layer of the network (see Fig. 2). The first step of the network is to take each 300-dimensional vector and convert it into an f-dimensional feature vector using the first layer of the network. These m feature vectors are combined to make an $m \cdot f$ dimensional vector which is fed into an m-dimensional hidden layer (note that this parameter is independent of the number of words chosen to represent the song, but it is convenient to use the same number in both cases). This hidden layer feeds into the final 4-dimensional prediction layer. The activation function at each stage was ReLU and the final prediction is obtained by taking the argmax of the 4-dimensional prediction layer. This process was repeated for different values of the hyperparameters m and f.

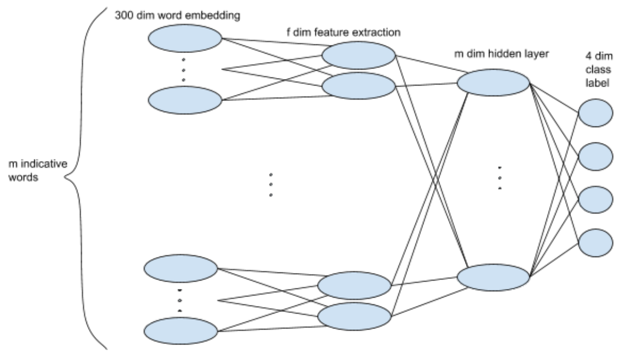


Fig. 2. Depiction of the deep neural network used

B. Results

The accuracies for class prediction, valence prediction, and arousal prediction attained by the model after training under different hyperparameters are all displayed in the following table:

Words (m)	**Features** (f)	**Accuracy**	**(Valence)**	**(Arousal)**
5	5	45.41	60.00	67.03
5	10	44.86	62.16	65.41
5	15	47.03	62.70	67.57
5	30	46.49	64.86	67.57
10	5	43.78	54.05	68.11
10	10	44.32	59.46	65.41
10	15	44.86	58.92	69.19
10	30	44.32	58.38	66.49
20	5	40.00	64.86	61.08
20	10	43.24	68.11	62.16
20	15	43.78	66.49	61.08
20	30	43.78	65.95	62.70

This table shows a general trend of increased accuracy as the number of features is increased. In addition, the accuracy increased with the training time of the model. However, this increase was overshadowed by the random initialization of the neural network; surprisingly, as the number m of words used to represent a song increased, the accuracy quickly decreased. Similarly, when the dimension f of the feature vectors increased past 15, the accuracy decreased. On the validation set, the error did not steadily decrease as the loss function of the neural network was minimized. This is the result of a number of factors including: the regularization term on the network's weights, the random initialization of the network, and the disparity between the training and validation data sets. The trend which remains visible about all networks, however, is the improvement in accuracy when compared to the Naïve Bayes models.

Interestingly, when $m = 20$, the networks saw a large improvement in valence accuracy. This is unique to the neural network approach, as all previous models suffered from very low valence accuracy. On the other hand, the networks which performed best overall were those with small m, and these, as with previous models, had higher arousal accuracy. This variance, as well as the networks' general failure to strictly improve as f increases, could be accounted for by the choice of loss function, as well as the method of preprocessing.

VI. Conclusion

There is more work to be done in improving the accuracy of the models obtained here. In particular, the neural network model could be less dependent on the order of the features passed in from the preprocessing step. For this, the technique of parameter sharing, as used in convolutional neural networks, could be employed. Additionally, more computation time could be invested to search the parameter space more thoroughly. Tweaking the network in this way should result in higher accuracy of the model. We believe that with a more delicate treatment of the structure of the neural network, overall accuracies could exceed 50%.

As a topic for future research, one could apply the technique of balancing the number of important words used by mood, as was done in the second enhancement of the Naïve Bayes model presented here. This would likely improve the results of a neural network model since it improved the results of a Naïve Bayes model.

References

[1] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[2] Jonathan Posner, James A Russell, and Bradley S Peterson. "The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology". In: *Development and psychopathology* 17.3 (2005), pp. 715–734.

[3] Foster Provost and Pedro Domingos. "Well-trained PETs: Improving probability estimation trees". In: (2000).

[4] Hao Xue, Like Xue, and Feng Su. "Multimodal music mood classification by fusion of audio and lyrics". In: *International Conference on Multimedia Modeling*. Springer. 2015, pp. 26–37.