CS224N Final Project
Chord Prediction Using Lyric Sentiments
Submitted: Dec 8, 2013
Ashwin Apte (ashwina), Dimuth Kulasinghe (dimuthk)

1 Introduction

1.1 Chordal Sentiment

Chords are one of the fundamental building blocks in music. They give harmony and character to the overlying melodies, and give an identity to a song. Chord progressions can often express sentiment as well: the progression `C-Am-Dm-G7-C-Fm-C` played in *No Surprises* by *Radiohead* is a sad sounding progression, and the melody and lyrics in the song reflect this. The progression `C-G-G7-C-F-C-G-C` played in *Obla Di Obla Da* by *The Beatles* sounds much happier by comparison.

1.2 Problem Statement

We attempted to build a classifier that, given lyrics for a song, outputs a chord progression for the song out of a set of N possible progressions. Given a dataset of songs with their associated lyrics (data) and a defining chord progression (label), we run various prediction algorithms and extract numerous features to see what level of accuracy we can achieve.

1.3 Prior Work

There hasn't been any previous work specifically regarding correspondence of chords to lyrics. Significant work has been performed in the realm of lyrical sentiment analysis, however, such as lyric-based song sentiment classification using vector space models [1] and maximum entropy classification [2]. Both authors note a data sparsity issue, as well as the relative shortness of lyrics making it difficult to accurately asses sentiment.

Tonal sentiment analysis has traditionally been performed through the transformation of acoustic signals into primitive musical constructs such as melodic lines, chord progressions, tempo, etc. A paper by Katayose, H., extracts these primitives, and then constructs sentiments for them based on a rule set developed by musical experts [3].

2 Data

There are complications in labelling a progression by a single progression; there is no predefined length or definition of a characteristic progression for a given song. Hence, we approached the problem by running queries into a website [4] that returns songs for

a given progression. The site holds the actual chords played for every song, and returns those for whom the progression is contained within. We did this for about 6-7 progressions (in each key, ultimately transposed to `Cmaj` and `Cmin`), and selected songs for which the progression was played at least 3 times in the song. Furthermore, we removed songs which had more than one classification.

We had some trouble collecting large amounts of data using this approach. While common chord progressions, such as `C-Am-F-G`, returned thousands of results, most progressions only returned a few hundred. Ultimately, we reduced the number of `C-Am-F-G`-classified songs in our database so as to keep the proportionate distributions of the labels more uniform.

The corresponding lyrics were then retrieved for each song by inputting the song and artist name into a lyric query engine [5]. Over half the songs inputted didn't return lyrics due to either translation errors or not existing in the database; out of our original 2000 chord-labelled songs, we received lyric data for around 600. Finally, we also obtained user-written tags for each song using the Last.fm API [6]; retrieval rates for this stage were over 95%.

Finally, songs written in foreign languages were removed, giving a total of 574 usable songs for our analysis.

Progression	**Usable Data**
`Cm-Ab-Eb-Bb`	157
`C-Am-F-G`	129
`G-F-C-G`	102
`C-Em-Am`	62
`C-Am-Dm-G`	48
`Cm-Fm-G`	45
`C-Bb-F-G`	31

Table 1: Statistics on usable data for each chord progression.

3 Feature Selection

3.1 Sentiment-Based Features

We tried 3 approaches for sentiment analysis of lyrics:

- Word level sentiment using NLTK movie reviews dataset
- Sentiment parsing based on the Stanford Sentiment Treebank parser
- User generated tags crawled from Last.fm

3.1.1 Word Level Sentiment Analysis

The simplest approach used the NLTK movie reviews corpus to measure the individual sentiment of each word in a set of lyrics. The movie reviews corpus consists of 2,000 movie reviews with manually labeled sentiment polarity for each review. For any word appearing in this corpus, we can count the number of times it occurs in positive reviews and negative reviews. The final polarity for each word is the greater of these two counts. Using this word level sentiment polarity, we can classify songs into positive and negative by counting the number of positive and negative words appearing in the lyrics. We use this song polarity as a feature to train our classifier.

Progression	Positive Sentiment	Negative Sentiment
`Cm-Ab-Eb-Bb`	88	69
`C-Am-F-G`	84	45
`G-F-C-G`	63	39
`C-Em-Am`	42	20
`C-Am-Dm-G`	30	18
`Cm-Fm-G`	34	11
`C-Bb-F-G`	19	12
Total	360	214

Table 2: Positive and negative word level sentiment classifications for each chord progression.

3.1.2 Stanford Sentiment Treebank Parser

The word level approach detailed above has a number of shortcomings. It looks at words in isolation, ignoring information about the intended meaning that exists in the structure of the sentence. The order of words is not taken into consideration, neither is colocation of specific words, like negators.

The Stanford Sentiment Treebank consists of fine grained sentiment labels for 215,154 phrases in the parse trees of 11,855 sentences. A Recursive Neural Tensor Network is trained on this treebank. This model takes into account compositionality of a sentence, and is thus a powerful tool for sentiment analysis at a sentence level. The trained model is available for use from the Stanford NLP group's website [7].

The trained model was used to predict sentiment for the song lyrics in our dataset. The sentiment output of the model is on a 5 point scale: {`Very negative, Negative, Neutral, Positive, Very Positive`}. The song lyrics are passed a sentence at a time to the sentiment parser and the sentiment is recorded. Based on the number of positive and negative sentences present in the song, the song's sentiment polarity and strength is determined.

Very Negative	Negative	Neutral	Positive	Very Positive
95	330	106	36	7

Table 3: Stanford sentiment treebank parser classifications.

This appears to be a very skewed distribution, with negative and very negative songs making up the bulk of the dataset. On inspection, the sentiment of the songs do not seem to represent such a skew. As discussed later, this is an artefact of the structure of many songs.

Another intuitive factor is that the song chorus might be thought of as the lyrics' equivalent of a signature of the song. The chord progression would be the musical signature. To predict one from the other, we could focus only on the chorus. Additionally, we would expect the chorus to express the sentiment of the song much more accurately, than the entire lyrics.

For each song in the dataset the chorus was was extracted from the lyrics. The chorus was defined as the "most frequently repeating sequence of 4 or more sentences in the lyrics". This ensures that smaller sentence fragments that may repeat often through the song aren't mistakenly chosen as the chorus. (for example, sentences that end in the same words to rhyme). Once the chorus was extracted, the sentiment parser was run as above to extract sentiment for each song. The following distribution resulted:

Progression	Very Negative	Negative	Neutral	Positive	Very Positive
`Cm-Ab-Eb-Bb`	7	81	46	18	5
`C-Am-F-G`	9	71	30	19	0
`G-F-C-G`	3	55	34	10	0
`C-Em-Am`	5	34	15	6	2
`C-Am-Dm-G`	1	27	14	6	0
`Cm-Fm-G`	5	27	12	1	0
`C-Bb-F-G`	2	15	10	4	0
Total	32	310	161	64	7

Table 4: Stanford sentiment treebank parser classifications for each chord progression based on chorus lyrics.

This gives a slightly more balanced distribution, but it is still heavily biased towards negative. The sentiment derived from the entire lyrics as well as from the chorus alone were both used as features to train the classifier.

3.1.3 Sentiment Based on User Generated Tags

This method of extracting sentiment information is not derived from the lyrics of the song, and is thus different from the other two methods described above. Also, this method incorporates information about factors other than sentiment, such as genre, period (90's, 80's) etc.

Last.fm is a music discovery service that allows users to interact with songs, and recommends songs that they may like. Amongst other things, users may annotate songs with tags that they feel best describe the song. These tags thus are a potential source of information about the sentiment of the song.

Last.fm has an API that allows developers to get information related to songs, artists, albums etc. The API was used to get user annotated tags for the songs in our dataset. The tags were stemmed to ensure uniformity. There were a total of 4,795 distinct tags for the songs within our dataset. Some tags consist of multiple words. These were broken down to single words to reduce sparsity of the data. For example, "alternative rock", "alt rock" and "rock" all contain "rock". If we consider the entire tags, these are three distinct tags. Breaking them down to word level results in 3 songs with "rock" associated with it. This results in a total of 7,805 tags associated with the songs in our dataset, but the sparsity of the top tags reduces significantly.

The tags were used as indicator features to train the classifier.

3.2 Bag of Words Feature

A bag of words feature was also extracted from the lyrics. The NLTK movie reviews corpus was used as a master vocabulary. This contained 25,859 distinct words after stemming. The song lyrics in our dataset had 4.769 distinct words after stemming. Of these, 818 words were not present in the master vocabulary, and were thus ignored. Though this results in loss of valuable data, it also guards against overfitting due to use of words that appear only in the training set.

3.3 Other Features

Other features such as word counts and line counts for the entire lyrics and the chorus, Part of Speech counts etc. were extracted and used to train the classifier. As discussed below, these features did not result in any significant improvement and were thus dropped from our final model.

4 Feature Selection

The machine learning model over our dataset was a multi-class classification problem over 7 classes (chord progressions). We used the following machine learning classifiers to train and test our model:

- Multinomial Naive Bayes (with Laplace smoothing): This model is traditionally used in text classification problems with bag of words features, like spam classification. Laplace smoothing was used to prevent overfitting.
- Logistic regression (max-ent): Logistic regression has been shown to have good results in text classification, especially in natural language applications.
- SVM: Same as above. Can be used since number of categories is small (7).
- Decision trees: Since we had a small dataset with a large number of features, decision trees might prove to be useful
- Random forest: This would eliminate tuning problems associated with decision trees, giving more robust results

The `sklearn` library was used to implement all the classifiers above.

5 Results

Since we had a small dataset, 5-fold cross validation was used to ensure robustness of results. The results of the classifiers were compared against a random baseline, that is, what would be the accuracy of a predictor that chose chord progressions at random from the set of 7 possible progressions. The metric of interest is accuracy (identical to precision and recall in this context).

Classifier	**Accuracy**
Random Baseline	14.44%
Multinomial Naive Bayes	22.12%
Logistic Regression	19.98%
Linear Support Vector Classifier	22.05%
Decision Tree Classifier	23.82%
Random Forest Classifier	20.67%

Table 5: Performance of each classifier. The random baseline accuracy is an average of 100 runs. The Random Forest classifier accuracy is based on an average over 10 runs using 5-fold cross-validation in each run.

6 Discussion

First and foremost, it is clear that the classifier does not do a very good job of predicting the chord progression. An accuracy of 22% is too low for the model to be of any practical use.

On the other hand, we see that our model does give significant improvement over a random baseline. This means that the model does have some predictive power. We analyze here the reasons for this predictive power, and avenues to explore for further improvement in the classifier's performance.

Taking the Linear Support Vector Classifier as the model for illustrative purposes, we analyze the feature importances. The accuracy scores for using individual feature types are as below:

Classifier	**Accuracy**
Bag of Words	22.39%
Lyrics Sentiment	6.63%
Chorus Sentiment	7.38%
Last.fm Tags	24.41%
Lyrics Line Count	14.32%
Lyrics Word Count	15.46%
Chorus Line Count	10.39%
Chorus Word Count	7.45%

Table 6: Performance of Linear Support Vector Classifier using various features.

It is clear from this that the sentiment parser by itself is a very poor predictor for chord progressions. We discuss possible reasons for this below. Bag of words and Last.fm tags are the only reasonable features. For both these features, the dataset we are working with is extremely small. With only 574 songs, we face severe sparsity issues. We hypothesize that a much larger dataset might be able to predict chord progressions much more accurately.

The sentiment feature based on the Stanford Sentiment Treebank parser model fails miserably to predict chord progression. On careful analysis of the predicted sentiment for a song and the lyrics of the song, we find numerous examples where the lyrics of the song taken even at the sentence level give misleading results in terms of sentiment.

An example is below: (entire lyrics are not given for brevity) The song Again by Bruno Mars is classified as `Negative` based on entire lyrics as well as chorus alone. It has a large number of negative sentences, such as *Hands over my head thinking*, *What else could go wrong*, *Would've stayed in bed*, *How could a day be so long?*. All these, and

more, are classified `Negative`. There are only a few positive classified lines in the song, like *That for you I'd do it all over again*. But on listening to the song, it is clearly a positive song. Thus, songs may contain a large number of lines of one polarity, but a few lines of the opposite polarity might be enough to decide the actual polarity of the entire song.

This problem is apparent in a number of songs in the dataset. This is the same problem that we encounter in word based sentiment analysis, but now at sentence level. Thus, song sentiment analysis from lyrics fails at sentence level. We need to find a way to analyze lyrics of the song as a whole, rather than sentences or words. Using the sentiment parser based on the Sentiment Treebank on the whole lyrics or chorus gives very poor results, with most songs being classified as `Negative` or `Very Negative`. Thus, a different approach might be needed.

The Last.fm tags can be used much more effectively if we could reduce sparsity by unifying different tags that mean the same thing. For example, `joyful, joyous, happy, jovial, jolly...` etc. might all indicate the same mood or sentiment. But these are different tags. Unifying them would give a much better mood based tagging. Unfortunately, much of this has to be done manually by looking at tags and identifying possible synonymous candidates. Another factor is that the tags are user generated, so typos, spelling variations, punctuation etc, complicate matters.

Also, chord progressions may have ambiguous sentiments when considered alone. The progression `C Am F G` is extremely common in music and has been used in songs encompassing a large range of emotions; it is difficult to attribute a single sentiment to it. We believe a more effective approach would be to consider other factors to accompany the progression in labelling the music; for example, the tempo of the song, paired with the associated progression, may be more indicative of the involved sentiment. It is difficult to obtain this sort of data, however, and a more effective approach may be to extract the progressions and tempos from the acoustic signals themselves, as performed in [3].

7 Conclusions and Further Work

To conclude, we see that our model though not practically usable in the current form, points us in some interesting directions. Firstly, if a larger dataset with chord progressions and lyrics could be crawled, even the existing model may give much better results, as sparsity of the bag of words and Last.fm tags reduces. Secondly, unification of Last.fm tags as discussed above might turn out to be an interesting approach. WordNet synsets might be able to help reduce the manual labor involved in unifying tags.

Lastly, sentiment analysis of song lyrics requires further attention. It is a problem that needs to be solved taking into account the entire body of a song's lyrics' as a whole.

References

[1] Xia, Yunqing, Linlin Wang, and Kam-Fai Wong. "Sentiment Vector Space Model for Lyric-Based Song Sentiment Classification." *World Scientific* 21.04 (2008): n. pag. IEEE Explore. Web. 2 Dec. 2013.

[2] He, Hui, Jianming Jin, Yuhong Xiong, Bo Chen, Wu Sun, and Ling Zhao. "Language Feature Mining for Music Emotion Classification via Supervised Learning from Lyrics." *Lecture Notes in Computer Science* 5370 (2008): 426-35. Springer. Web. 2 Dec. 2013.

[3] Katayose, H., M. Imai, and S. Inokuchi. "Sentiment Extraction in Music." *Pattern Recognition, 1988., 9th International Conference on 2* (1988): 1083-087. IEEE Xplore. Web. 2 Dec. 2013.

[4] *Chord Hunter*. N.p., n.d. Web. 2 Dec. 2013. <http://www.chordhunter.com>.

[5] *Metro Lyrics*. N.p., n.d. Web. 2 Dec. 2013. <http://www.metrolyrics.com>.

[6] "Last.fm Web Services." *Last FM*. N.p., n.d. Web. 2 Dec. 2013. <http://www.last.fm/api>.

[7] Socher, Richard, Alex Perelygin, Jean Y. Wu, Jason Chuang, Chirstopher D. Manning, Andrew Y. Ng, and Christopher Potts. "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank." (n.d.): n. pag. Nlp.stanford.edu. Web. 2 Dec. 2013. <http://nlp.stanford.edu/ socherr/EMNLP2013_RNTN.pdf>.