Information Retrieval and Text Mining
Emotion Mining and Topic Extraction on Lyrics

Christoph Emunds (i6146758)
Richard Polzin (???)

May 27, 2017

Contents

1 Introduction

Measure the similarity of artists with respect to emotions and topics We give information about our dataset and how it was gathered Next, we describe the preprocessing that has been done on the data In the following two sections, we discuss the emotion mining and the topic extraction Afterwards, we show how the results have been visualized We explain the tools we used All of the processing is done in Python 3.6

2 Dataset

The *pylyrics3* package is able to download the lyrics for all songs of a given artist if it exists on *lyrics.wikia.com*. The lyrics are returned in a dictionary, where the keys are the song names and the values are the songs' lyrics. The usage of this package is straightforward and removes the necessity to deal with the raw HTML data. The retrieved songs are stored in JSON format, each song having a title, the artist's name and the actual lyrics.

Some (how many?) of the top 100 Billboard artists could not be resolved when querying pylyrics3. This could be due to spelling variations. Moreover, we scraped the names of 30 metal bands from

got-djent.com and requested their lyrics via pylyrics3. Some (how many?) of them could also not be resolved, either due to spelling variations or because they are not famous enough to have an entry in die LyricsWiki. Altogether, we ended up with 11.??? songs from 120 artists.

In addition to the dataset we gathered ourselves we were given a database of more than 16.000 artists and 228.000 originally built by Sibirtsev and Schuster (???). In the remainder of this document, we call this dataset ???

Some of the lyrics are not very clean. For example, they contain markers for the chorus and the verse part...

3 Preprocessing

For tokenization and stemming, the *nltk* Python package has been used. It provides easy to use function for splitting a document into sentences and splitting the sentences in words. Furthermore it implements several stemming algorithms. In our implementations, we use the Porter algorithm. We downloaded a list of stopwords that are more appropriate for lyrics. Moreover, in the process of topic extraction, we iteratively added words to this stopword list whenever we found...

The tf-idf features are built with the help of the *sklearn* package. It offers the *TfIdfVectorizer* class, which...

For the topic extraction, all songs should be in the same language, since otherwise the used algorithm will create an extra topic for every language.

In our dataset, all songs, except for one, are in English. For the ??? dataset, we implemented a rudimentary language detection using the *langdetect* Python package (Danilak, 2014). This package is a port of Google's language detection library.

4 Visualization

Used Gephi to visualize the results as a graph The *pygraphml* package was used to dump the extracted information to a GraphML file, that can be imported by Gephi. Calculate modularity classes in Gephi

5 Emotion Mining

The emotion mining is done by first tokenizing every song's text. Afterwards, every token is looked up in an emotion lexicon for its emotional value. For our experiments, we used the *NRC Emotion Lexicon* (EmoLex) by... This lexicon specifies the emotional value for the eight emotions *anger*, *anticipation*, *disgust*, *fear*, *joy*, *sadness*, *surprise*, and *trust*.

The emotion mining we apply to the lyrics is only rudimentary. It does not handle negations...

We define the *strength* of an emotion by...

5.1 Results

Figure 1 shows a graph of all 120 artists, where each artist is connected to every emotion. The edges themselves have weights depending on the strength of the emotion in an artist's song. The weights are normalized between 0 and 1.

The full graph is of course quite confusing and does not yield any useful insight. Therefore, Figure 2 shows a graph that contains only those edges with weights higher than 0.3.

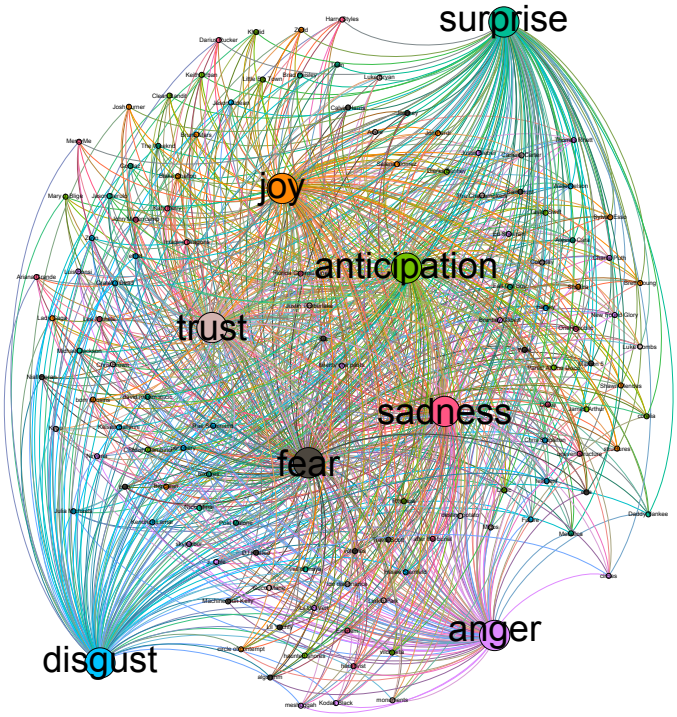

Figure 1: Full emotion graph

Figure 4 shows the emotion similarity between individual artists. The modularity score separates the graph into three main regions, which...

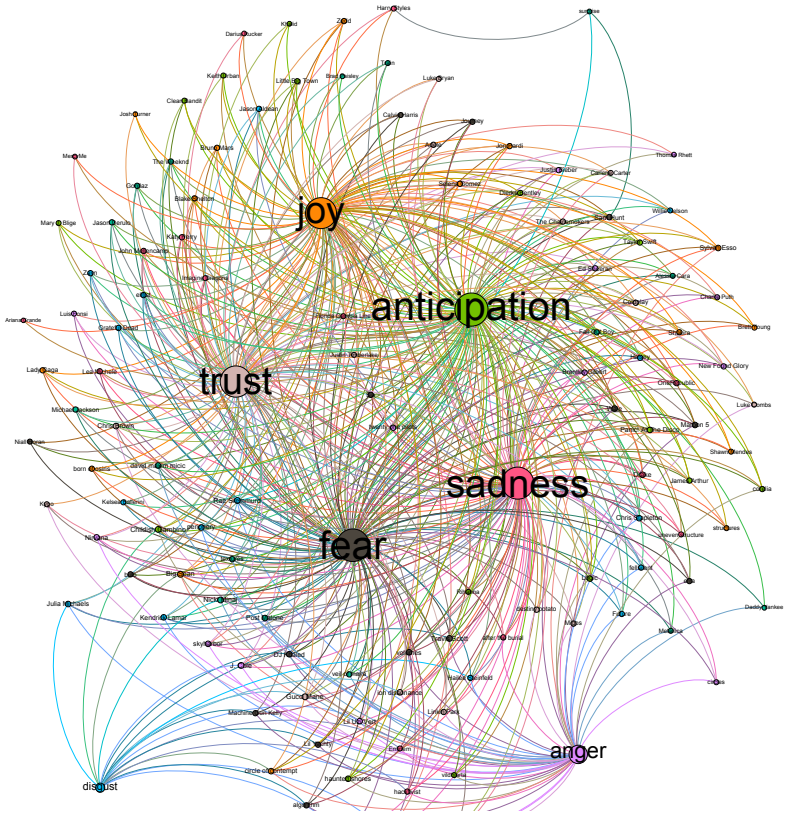

Figure 2: Emotion graph with edge weights higher than 0.3

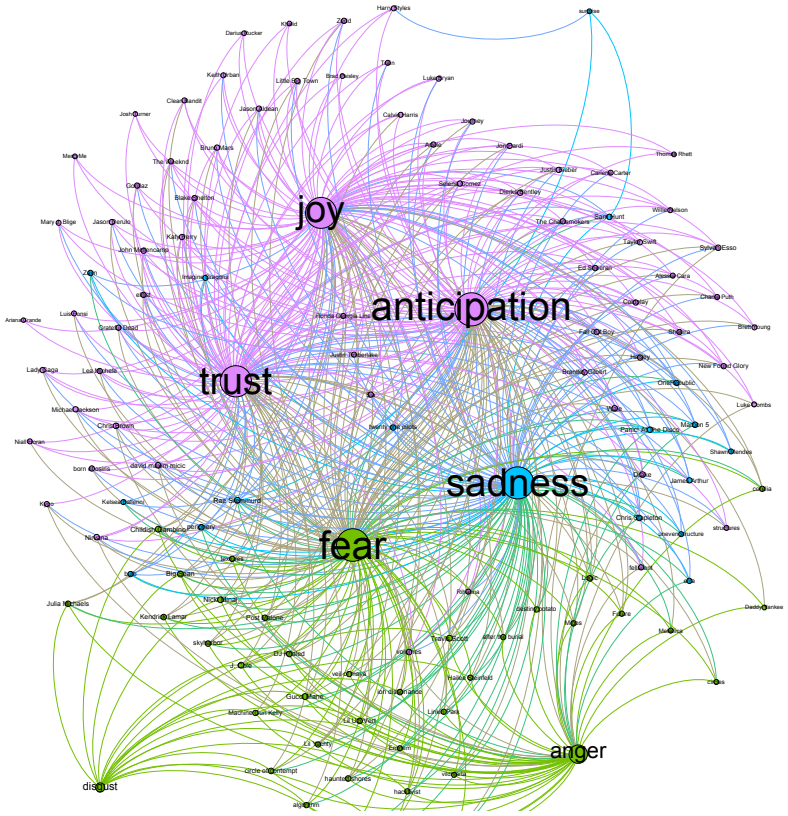

Figure 3: Reduced emotion graph with recalculated modularity

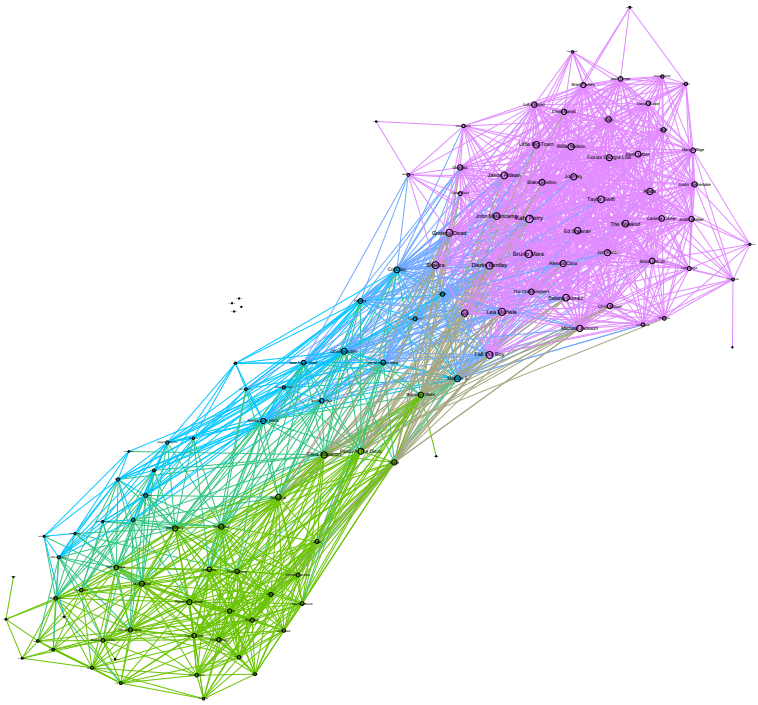

Figure 4: Emotion similarity between artists

6 Topic Extraction

The Non-negative Matrix Factorization (NMF) algorithm has been used to extract potential topics from the lyrics. The *sklearn* Python package provides everything necessary to apply NMF to the data. First, a tf-idf feature vector for each song text is calculated. Afterwards, these are fed into the NMF algorithm, which returns two matrices that...

We found that a number of ten topics is appropriate for our dataset. When choosing more topics, they start to overlap too much. When choosing fewer topics, too much information is put into every single topic. This is of course subject to the experimentalist's taste.

6.1 Results

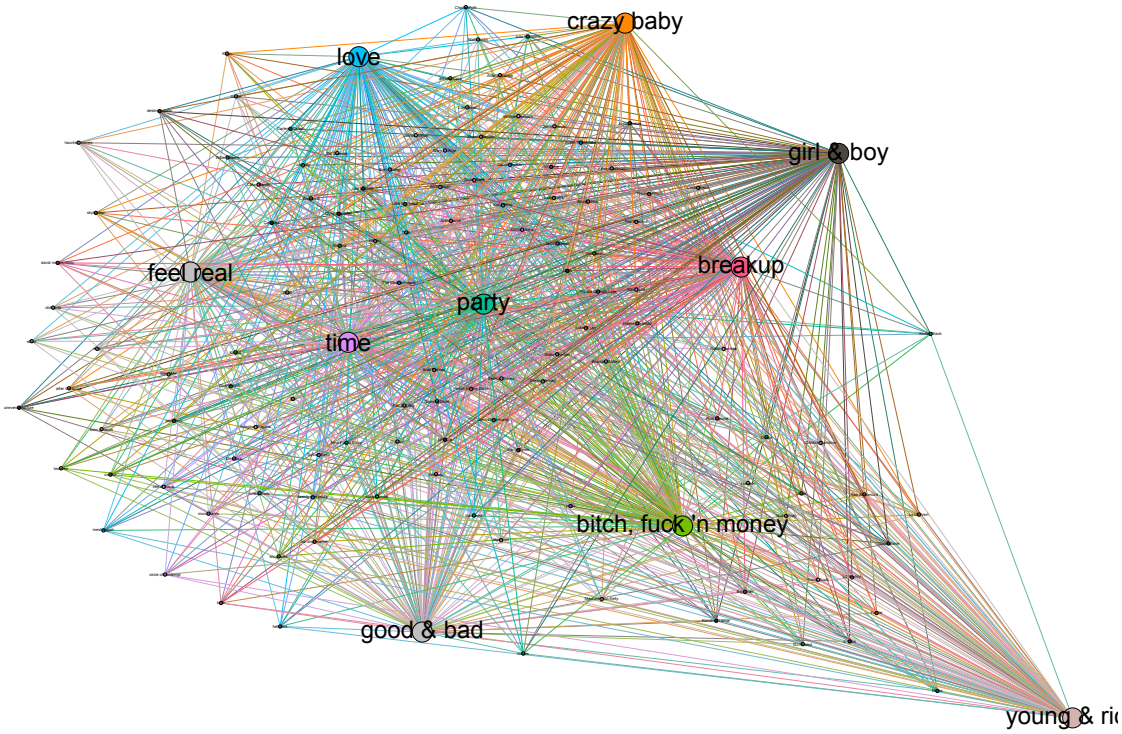

Figure 5: Full topic graph

Figure 8 shows the similarity of topics between artists. It is interesting to see that a lot of the rappers form their own subgraph, that is completely isolated from the rest of the graph.

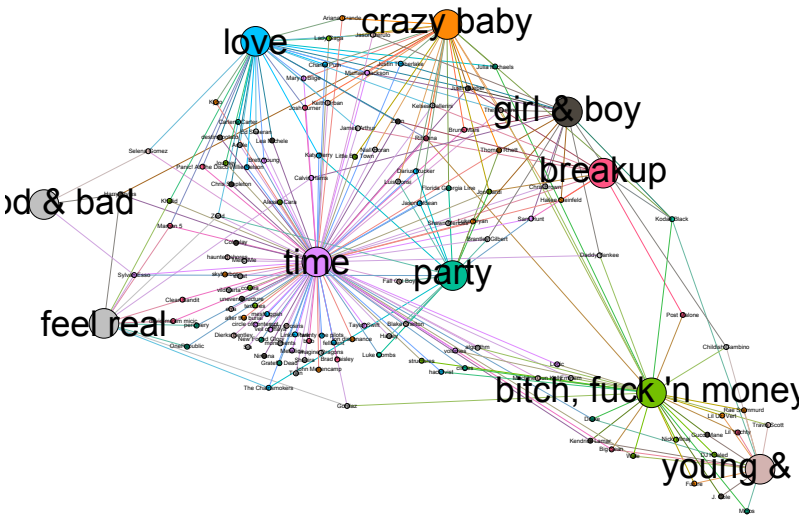

Figure 6: Topic graph with edge weights higher than 0.3

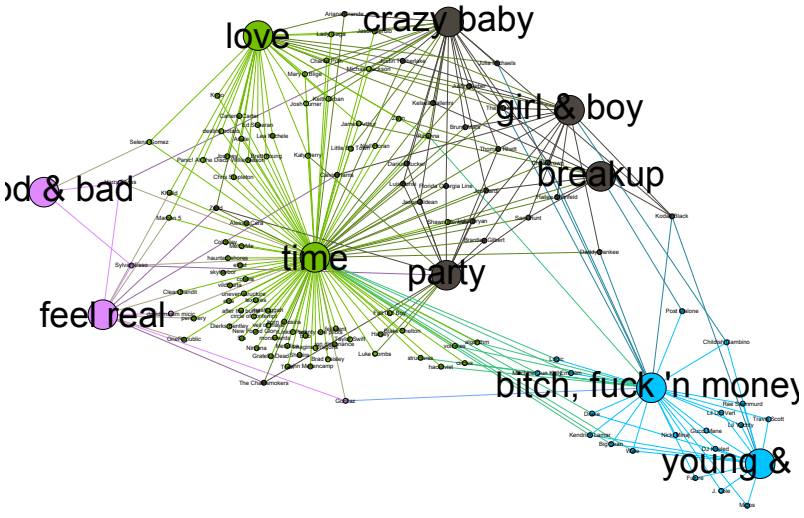

Figure 7: Reduced topic graph with recalculated modularity

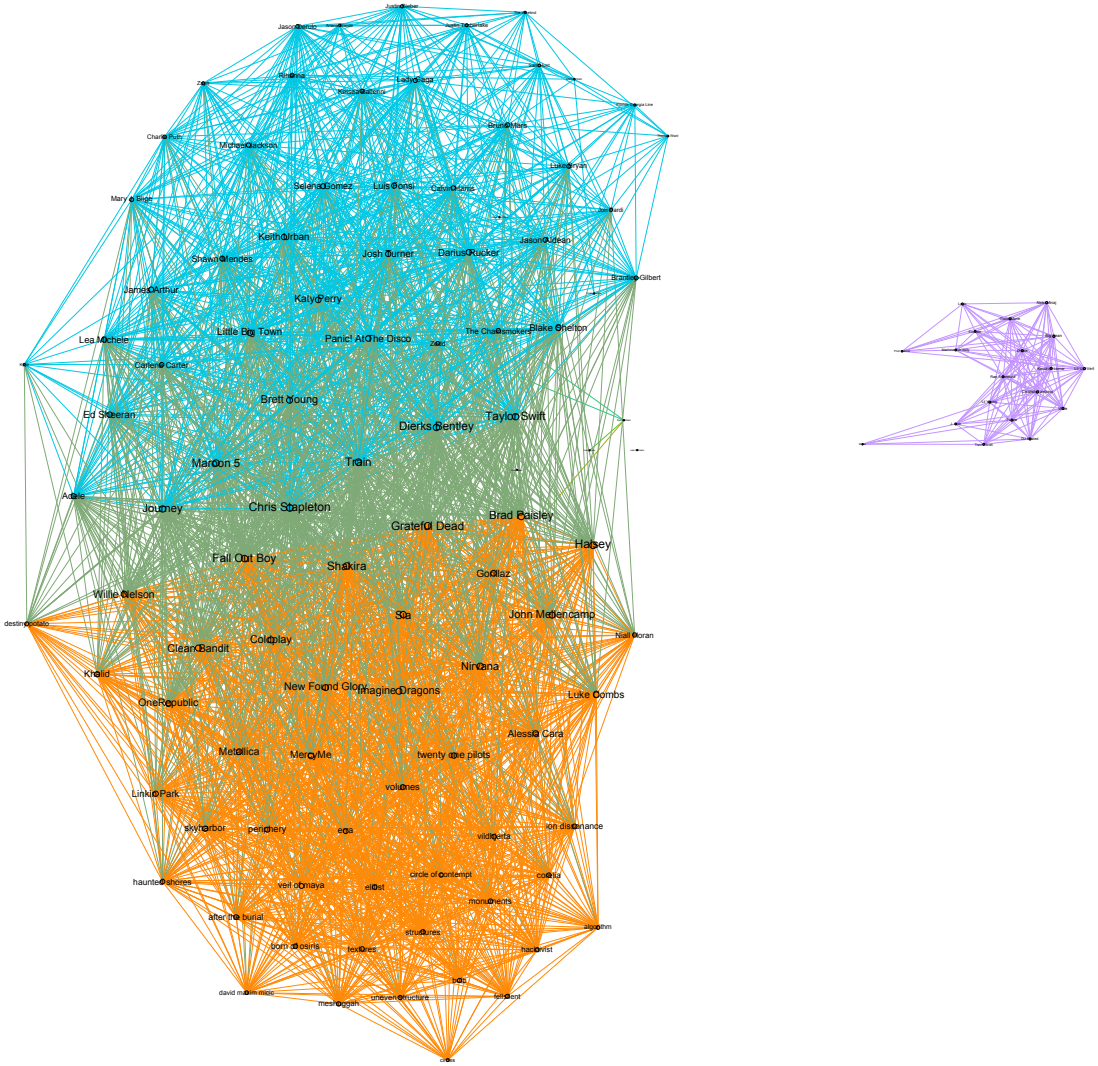

Figure 8: Topic similarity between artists

7 Other techniques

RAKE Summarization Text statistics (Flesch Score, etc.)

8 Conclusion

References

Danilak, M. (2014). *Language detection library.*