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

Christoph Emunds (i6146758)
Richard Polzin (i6145946)

June 11, 2017

Contents

1 Introduction

Leaving melody aside, there is a lot of information that can be extracted from a song's lyrics. In this work, we aim to measure the similarity of artists with respect to emotions and topics of their songs based on their lyrics. A suitable application of this could be the usage as an alternative to data driven recommender systems in music streaming services. The developed techniques can be used to recommend artists invoking similar emotions or singing about related topics.

In Section 2, we give an overview over our dataset and explain how it was gathered. Section 3 describes the preprocessing that has been done. We mention the tools we used for the visualization in Section 4. The actual emotion mining is described in Section 5 and the topic extraction in Section 6 together with their respective results.

2 Dataset

The *pylyrics3* package (Wenzel, 2017) is able to download the lyrics for all songs of a given artist if they have an entry on *lyrics.wikia.com.* This website, online since April 2006, contains more than 1.5 million songs based on crowd-sourcing. It is integrated into many different media players including Windows Media Player, iTunes, Winamp and Amarok (Colombo, 2017).

The lyrics are returned in a dictionary, where the keys are song names and the values are the lyrics. The usage of this package is straightforward and removes the necessity to deal with 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 of the top 100 Billboard artists could not be resolved when querying *pylyrics3.* This could be due to spelling variations or incomplete/currently maintained entries. Moreover, we scraped the names of 30 metal bands from *gotdjent.com* and requested their lyrics via *pylyrics3.* Some of them could also not be resolved, either due to spelling variations or because they are not popular enough to be listed. Altogether, we ended up with approximately 11,000 songs from 120 artists.

Some of the lyrics are not very clean. For example, they contain markers for the chorus and verse part or additional information like *INTRUMENTAL* or *SOLO.* In some lyrics contact information of the creator is included and the formatting often varies between different authors. We did not filter out these tokens, as they did not seem to have an impact on our results.

3 Preprocessing

All of the programming is done in Python 3.6 using different modules to interface existing NLP toolkits. Preprocessing the lyrics consists of several parts. The lyrics are tokenized, stemmed and stopwords are removed. Furthermore a TF-IDF feature matrix is calculated and the language of songs is detected. Most of the preprocessing steps are independent and can be executed in parallel.

For tokenization and stemming, the *nltk* Python package (NLTK-Project, 2017) has been used. The lyrics are split into sentences and the sentences are split into words. For stemming, *nltk* implements several algorithms. In our implementation, we use the Porter algorithm.

The *nltk* package also provides a stopword list. However, it did not yield satisfying results, so we sourced a list of stopwords that are more appropriate for lyrics. Moreover we tailored it to the specific lyric domain by iteratively adding stopwords that we found to obfuscate the results.

In addition to tokenization, stemming and stopword removal we built TF-IDF features utilizing the *scikit-learn* package (Pedregosa et al., 2011). With the *TfIdfVectorizer*, class a matrix of TF-IDF features can be created from a collection of raw documents.

For the topic extraction, all songs should be in the same language. Otherwise specific topics for the languages will be created, which is not the desired result. Therefore we needed to filter out lyrics that are not in English. We implemented a rudimentary language detection using the *langdetect* Python package (Danilak, 2014). This package is a port of Google's language detection library.

To summarize, the *nltk* toolkit is used for tokenization and stemming. A custom stopword list was developed and *scikit-learn* was used to calculate the TF-IDF features. Finally *langdetect* was used for language detection. After preprocessing we have the original text, estimated language and word tokens without stopwords that have been reduced to the word stems available for every song.

Various metrics, like flesch score, fog-score and flesch-kincaid levels, have been implemented, but were not used later. Therefore, they will not be discussed in this report.

4 Visualization

All the graph visualizations were created using the open-source platform *Gephi* (Bastian, Heymann, & Jacomy, 2009). Gephi is able to display a plethora of graphs and networks that can also be explored interactively. It provides a rich set of features, including filtering out edges with low weight, calculating modularity classes for the nodes in a graph, and coloring the nodes according to their modularity class.

To be able to import our results into Gephi, we made use of the *pygraphml* Python package. It provides a couple of methods to dump the extracted information into a *GraphML* file, that can be imported by Gephi. However, the pygraphml package is very rudimentary. An issue was that the weight attribute that has been added to edges is stored as a string, rather than a number, which makes it impossible for Gephi to recognize the edge weight properly. For now, we worked around this problem by simply exchanging a single line in the beginning of the generated GraphML file manually.

The second framework that was used is *word_cloud* (Mueller, 2017). It provides a Python interface to generate wordclouds from text. It was used in conjunction with *flask* (Ronacher, 2017) to spin up a local webserver that can be used to browse emotion and vocabulary wordclouds for the different artists. Wordclouds are created through the main scripts and are just displayed in HTML, as creating them in real time added significant delay when exploring artists interactively. An example of the generated wordclouds can be seen in Figure 1.

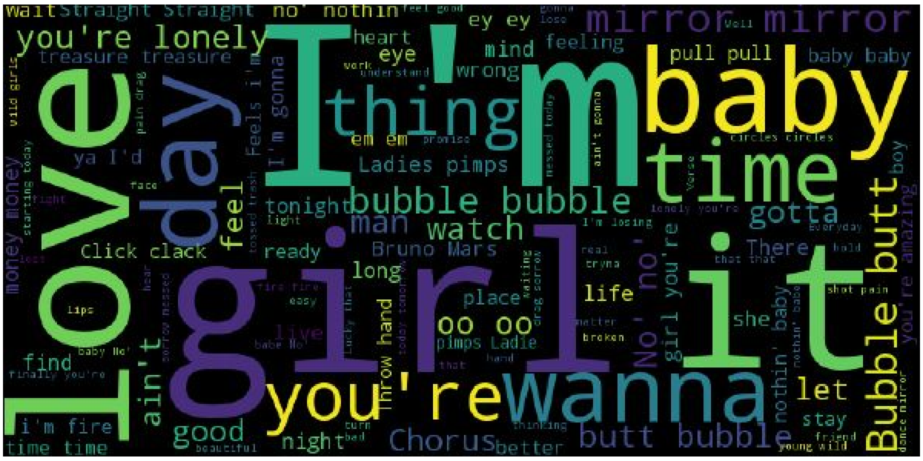

Figure 1: Wordcloud visualizing the most important terms in the lyrics of Bruno Mars. The size of the terms is in proportion to their frequency in the lyrics. The color is only used for a more appealing visualization and does not convey any meaning.

5 Emotion Mining

The emotion mining is based on the preprocessed data. Every token of the tokenized lyrics is looked up in an emotion lexicon for its emotional value. For our experiments, we used the *NRC Emotion Lexicon* (EmoLex) (Mohammad & Turney, 2013). This lexicon specifies emotional values for the eight emotions *anger*, *anticipation*, *disgust*, *fear*, *joy*, *sadness*, *surprise*, and *trust*.

The emotion mining we apply to the lyrics does not handle negations, sarcasm or complex syntactical structure. We were not able to find either an emotion lexicon that specifies the opposite emotional value for every common emotion word, nor suitable work that is related to our specific need for negated emotion words.

Through this analysis, we obtained an eight-dimensional vector for every song, specifying the strength of each of the emotions. These emotion vectors were then summed up and normalized per artist. For every artist, the result was a vector containing information about how strong each of the eight emotions is in their songs.

To calculate the similarity between the artists we used the cosine distance in the eight-dimensional space. The resulting real value represents the similarity of two artists with respect to the emotions that are expressed through their songs.

5.1 Results

Figure 2 shows a graph of all 120 artists, where each artist is connected to a varying number of emotion centers. The edges have weights depending on the strength of each emotion in an artist's song. The weights are normalized between 0 and 1.

The full graph of 128 nodes and 960 edges would be too overwhelming and would not yield any useful insight. Therefore, Figure 2 shows a graph that contains only those edges with sufficiently high weights. Our experiments showed that a threshold of 0.35 yielded a balanced mixture of complexity and conciseness.

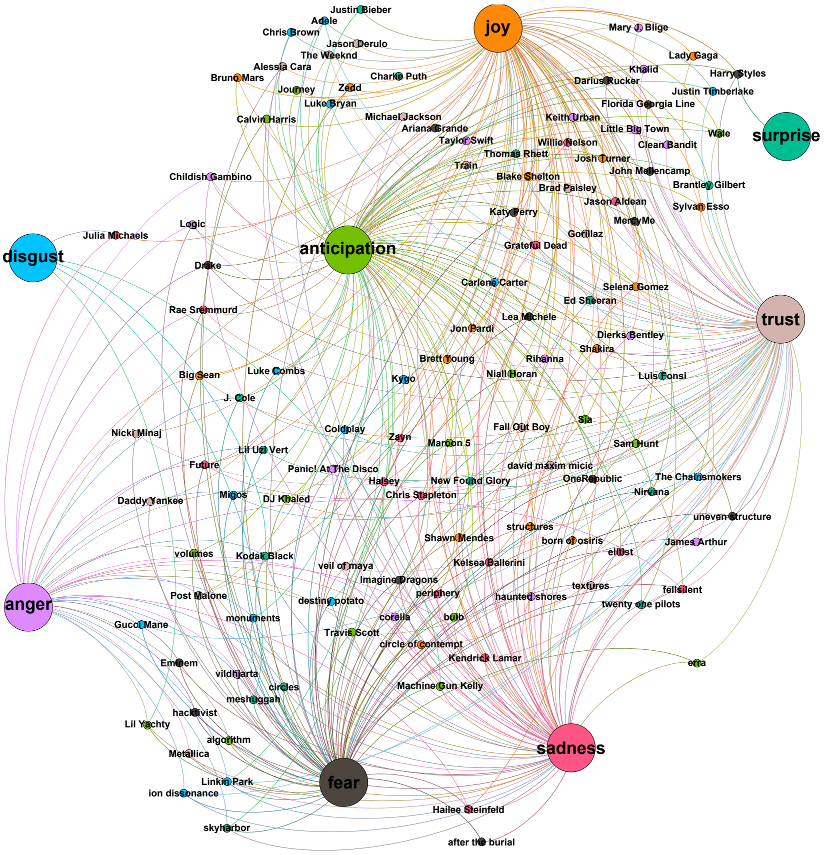

Figure 2: Emotion graph with edge weight of at least 0.35

Figure 3 shows the previous graph with a newly calculated modularity score. The graph decomposes into two clusters. The first cluster is made up of the emotions *anger*, *fear*, *sadness*, and *disgust*. The second cluster consists of the emotions *joy*, *surprise*, *trust*, and *anticipation*. This partitioning seems reasonable, since it somewhat follows an intuitive understanding of positive and negative emotions.

Towards these more negatively connotated emotions, there are a lot of Hip Hop, Rap, and Metal artists. Music associated with the Pop and Mainstream genre is closely connected to the positive emotions.

Figure 4 shows the emotion similarity between individual artists. The modular-

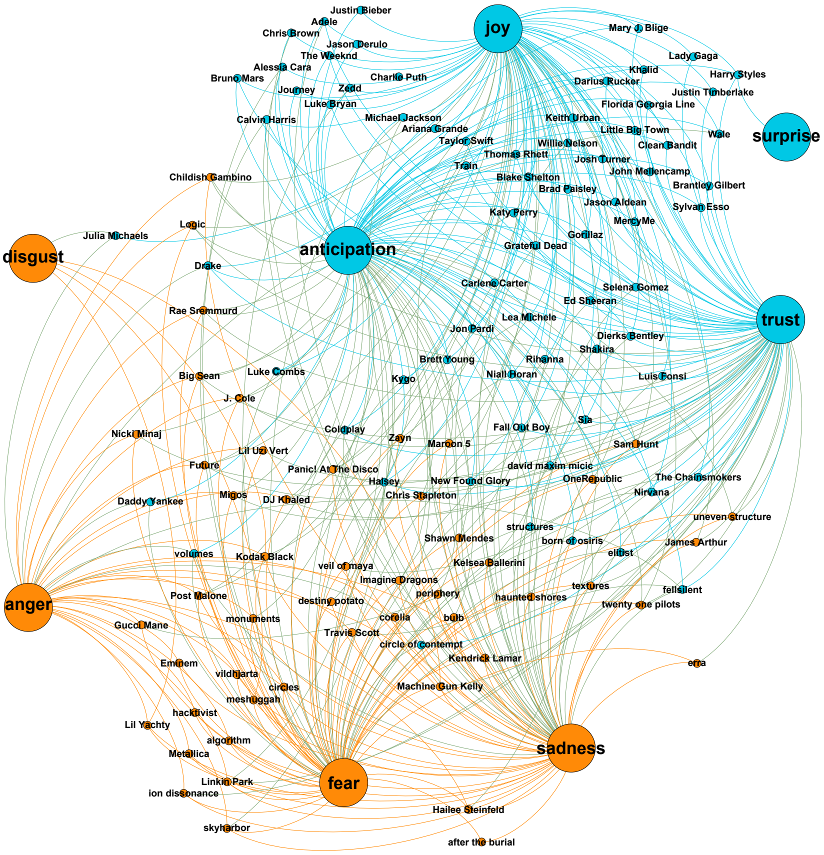

Figure 3: Emotion graph with recalculated modularity

ity score partitions the graph into two main regions, which relate to the results shown before in Figure 3. Pop artists are tightly clustered in one side of the graph, while Hip Hop, Rap and Metal make up the other part. Few artists whose edges have no strong enough weights are isolated from the graph.

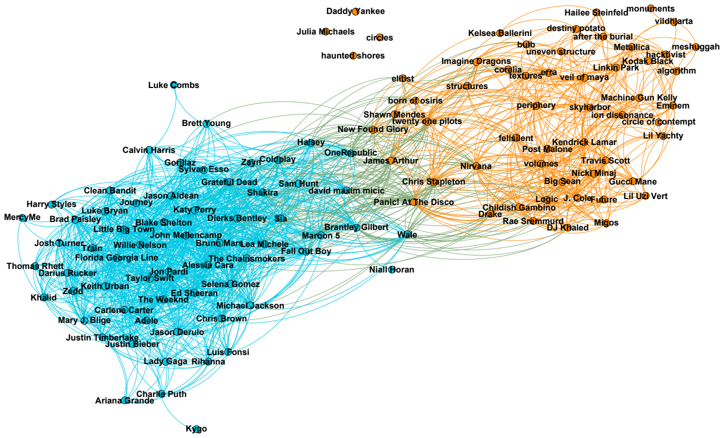

Figure 4: Emotion similarity between artists with edge weight of at least 0.98

6 Topic Extraction

While grouping artists by the emotions their songs express already provides interesting results, another part of this project was the topic based analysis of lyrics. This section describes the used technique and discusses the obtained results.

The Non-negative Matrix Factorization (NMF) algorithm has been used to extract potential topics from the lyrics. The *scikit-learn* Python package provides everything necessary to apply NMF to the data. In the preprocessing TF-IDF features have been calculated, which can now be used in the NMF algorithm. The NMF algorithm requires to predefine the number of expected topics. Through experiments we determined ten topics to be appropriate for our dataset.

Generating more topics yielded too much overlap between the topics, creating fewer resulted in very packed, broad topics. The number of topics is of course subject to the experimentalist's taste. For future work one could also experiment with hierarchical topics, but we felt the topics were already very focused such that splitting them up any further would not increase the informativeness of the results.

6.1 Results

Similar to the results of the emotion mining before, the graph in Figure 5 shows all 120 artists, each of them connected to a varying number of topics, depending on the strength. All weights are again normalized between 0 and 1. The graph shows only those edges that have a weight of at least 0.28

Figure 6 shows the partitioning of the graph after calculating the modularity score. The graph is partitioned into three clusters. The first one consists of the topics *young & rich*, and *bitch, fuck 'n money*. The second cluster has the

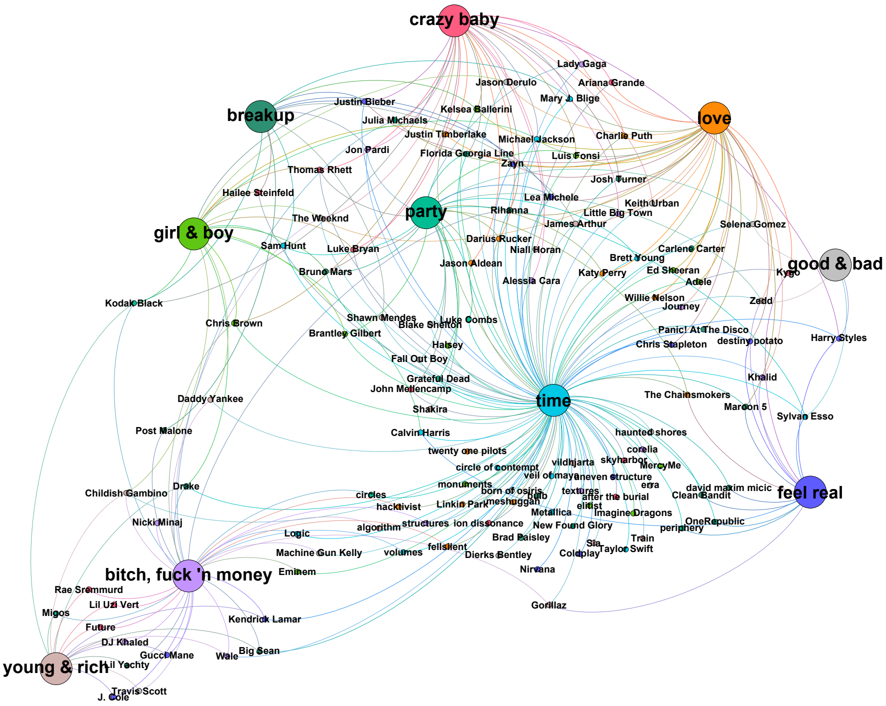

Figure 5: Topic graph with edge weights greater than 0.28

topics *feel real*, *time*, and *party*. The third and biggest cluster is made up of the topics *breakup*, *girl & boy*, *crazy baby*, *love*, and *good & bad*.

We felt that this way of partitioning the graph is reasonable and intuitive. The first cluster in the lower left corner of the graph inhabits a lot of Rap and Hip Hop artists, whose music is often about money, being rich and sexual intercourse. The second cluster is all about partying and having a good time while somewhat feeling alive or real. The exception to this are the Metal bands, a lot of which are associated with the topic *time*. These do not actually sing about partying and having a great time, but rather about the meaning of time itself. The third cluster comprises everything about human relationships, from love to breakup.

Figure 7 shows the similarity of topics between artists. The modularity score partitions the graph into three components. It is interesting to see that a lot of the rappers form their own subgraph, that is only connected to the rest of the graph by the artist *Daddy Yankee*[1]. In the right hand part of the graph, we again find the Pop and Mainstream artists to form one cluster, and Rock and Metal to form the other.

[1] Of course, the edge weight filter is chosen such that this strongest connection between the two parts survives.

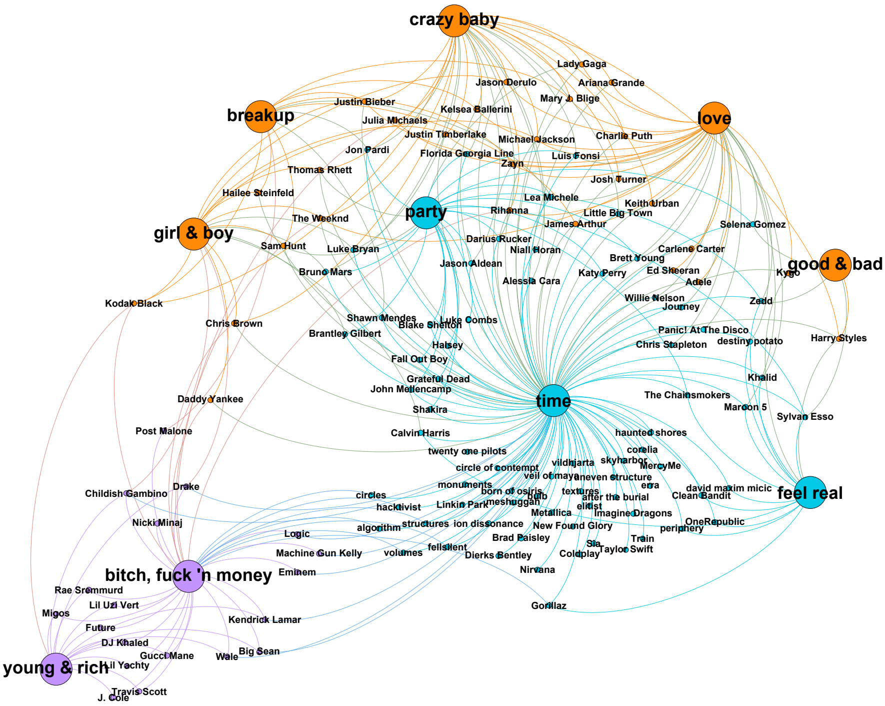

Figure 6: Topic graph with recalculated modularity

6.2 Evaluation

We tried to evaluate the resulting topics via held-out tests. For this, we assigned a label from 0 to 9 to each song depending on the highest component in its ten-dimensional topic vector. We split the songs for each topic into a training and a test set and trained an SVM with the songs' TF-IDF features.

Unfortunately, the best of the ten SVMs only achieved an accuracy of approximately 42%. This could have several reasons, the most likely being that the assignment of a song to a topic depending on the maximal value in its topic vector is too crude. For example, the topic vector of *Kendrick Lamar*'s song *Growing Apart* has a value of approximately 0.0208 in the first and 0.0206 in the second topic. With our approach, where each song is assigned exactly one label, this song would get the label 0, which is a very insensitive classification.

This could cause the feature space to not be linearly separable anymore. Since we only experimented with linear SVMs, these cannot achieve high accuracies in this particular domain.

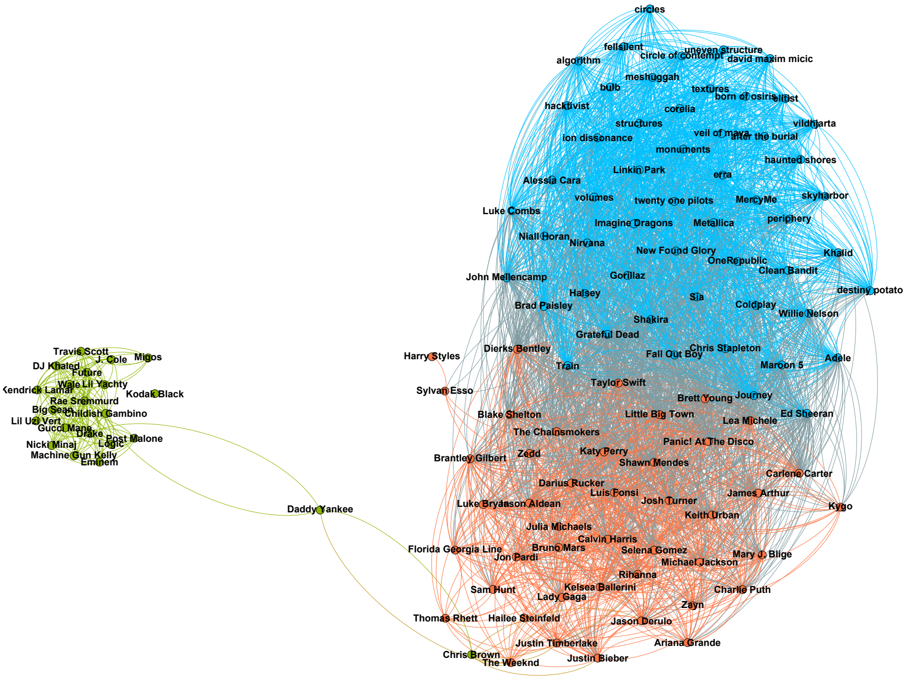

Figure 7: Topic similarity between artists greater than 0.82

7 Other techniques

In this section we want to mention some of the techniques we used that were neglected during the development, some of which were already mentioned earlier.

Part-of-Speech Tagging as well as Named Entity Extraction are techniques often used in the NLP domain. The *nltk* toolkit offers endpoints for both and can use different engines like the components created by the *Stanford NLP Group* or can even be trained on a custom text corpus. We implemented both techniques to extract additional information from the lyrics, but none of our high-level goals (emotion mining and topic extraction) benefited from the invested effort.

While especially the NER tags can provide some intuition about the topic of a song, they do not actually increase our performance in general. Only working with specific parts of speech or named entities distorts the results of emotion mining. As we use NMF for the extraction of topics, the TF-IDF values of words are used and the importance of words is already contained. NER tags could be useful if we would require information about e.g. persons or locations, but none of that has any relevance in the context of this project.

Another technique that partly suffered from the same issues was summarization. While interesting, it did not really fit the scope of this project and would only obfuscate results for emotion mining and topic extraction. Furthermore, due to the complexity/simplicity of some lyrics, this could make for a whole project on it’s own.

During preprocessing we initially included some metrics that contained information about the lyrics readability or the complexity of the vocabulary. These yield interesting insights into the repetitiveness of the lyrics. However, they did not contribute to improving our final results and were not part of any visualization we obtained in the end.

The last technique we applied was Rapid Automatic Keyword Extraction (RAKE). It is useful to get some understanding about the context of a song without actually having to read the lyrics. While we used it to quickly verify our results and to verify the word clouds, it does not yield any benefit for the final results.

To summarize, many of the techniques that are often applied, like POS and NER tagging are not required in emotion mining and topic extraction. They could even reduce the quality of the results by obfuscating the actual truths through badly handcrafted rules. The correlation of topics or emotions and the complexity of lyrics is an interesting topic, but not within the scope of this project.

8 Conclusion

To sum up the achieved results fit the intuition and are rather reasonable. The genres of Rap and Metal convey different emotions than Pop and Mainstream. Moreover, the detected topics clearly separate those genres. Rap and Metal also seem to be closer to each other than they are to the Mainstream music. However, this whole analysis is only based on the songs' lyrics. For future work the incorporation of melody or other aspects of music into the analysis would certainly lead to interesting results. Furthermore the correlation between genre /topics / emotions and an artist's net-worth would be interesting to explore.

The code as well as the dataset is publicly available on GitHub (Emunds & Polzin, 2017) and can be used for further projects under the GPL license.

References

Bastian, M., Heymann, S., & Jacomy, M. (2009). *Gephi: An open source software for exploring and manipulating networks.* Retrieved from `http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154`

Colombo, S. (2017). *Lyricwikia - fandom powered by wikia.* Retrieved from `http://lyrics.wikia.com/wiki/Lyrics_Wiki`

Danilak, M. (2014). *Language detection library.* Retrieved from `https://pypi.python.org/pypi/langdetect`

Emunds, C., & Polzin, R. (2017). *Lyrical analysis for emotion mining and topic modeling.* Retrieved from `https://github.com/DeastinY/text-mining`

Mohammad, S. M., & Turney, P. D. (2013). Crowdsourcing a word-emotion association lexicon. , *29*(3), 436–465.

Mueller, A. (2017). *A little word cloud generator in python.* Retrieved from `https://github.com/amueller/word_cloud`

NLTK-Project. (2017). *Natural language toolkit.* Retrieved from `https://www.nltk.org/`

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Ronacher, A. (2017). *A microframework based on werkzeug, jinja2 and good intentions.* Retrieved from `http://flask.pocoo.org/`

Wenzel, J. (2017). *Pylyrics3 - lyric scraper based on py-lyrics, updated for python 3.* Retrieved from `https://github.com/jameswenzel/pylyrics3`