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

Christoph Emunds (i6146758)
Richard Polzin (i6145946)

June 10, 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 could be 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

In this section an overview over the datasets used and the acquisition thereof will be given. Furthermore some comment about lyrics quality will be given.

The *pylyrics3* package (Wenzel, 2017) is able to download the lyrics for all songs of a given artist if it exists 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 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 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 (Sibirtsev & Schuster, 2014). In the remainder of this document, we refer to it as the external dataset.

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.

3 Preprocessing

This section goes into detail on the preprocessing that the lyrics went through. The different aspects and the toolkits are described.

All of the programming is done in Python 3 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 implementations, we use the Porter algorithm.

Nltk also provides a stopword list, but 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 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 was not the goal of the project. Therefor we needed to filter out lyrics that are not in English. In our dataset, all songs, except for one, are in English. For the external 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.

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

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

4 Visualization

In this section the tools used to visualize our results will be described and problems encountered using them will be discussed. Two different frameworks have been used for the visualizations which will be discussed in sequential order.

All the complex 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, ...

To be able to import our results to 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. We circumvented

this problem by simply exchanging a single line in the beginning of the GraphML file.

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 before 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 wordclouds generated can be seen in Figure 1.

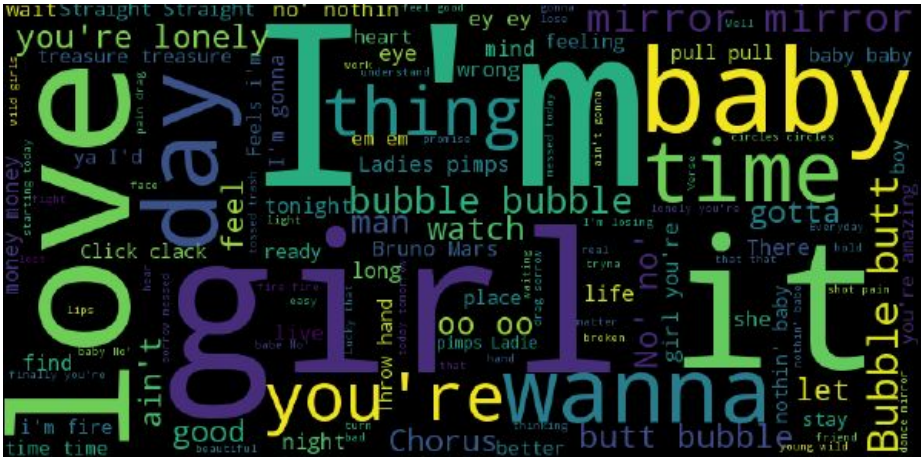

Figure 1: Wordcloud visualizing the most important terms in the lyrics of Bruno Mars. The size of the terms is related to their frequency in the lyrics. The color is only used for a more appealing visualization and does not convey any meaning.

To summarize pygraphml was used to export data to Gephi where visualizations involving the complete dataset were created. In addition wordclouds have been generated for the emotions and vocabularies of all individual artists. Those can be explored through the filebrowser ore more conveniently a local webserver created in flask.

5 Emotion Mining

An overview over the exact emotion mining workflow is given in this chapter. Furthermore first results are shown and discussed.

The emotion mining is based on the preprocessed data. The tokenized lyrics are used and every token 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 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, 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, or suitable work that is related to the our specific need for negated emotion words.

This analysis of the tokenized songs resulted in a vector containing values for the eight emotions. These eight-dimensional emotion vector were then summed up and normalized for every artist in the dataset. For every artist the result was a vector containing information about how strong each of the eight emotions is in his songs.

To calculate similarity between artists we used the cosine distance in the eight-dimensional space.

5.1 Results

Figure 2 shows a graph of all 120 artists, where each artist is connected to a varying number of emotions. The edges have weights depending on the strength of the emotions 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 ... 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 0.37 yielded a balanced mixture of complexity and conciseness.

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. In fact this kind of music is often more aggressive and ... Music associated with the Pop genre is closely connected to the positive emotions.

Figure 4 shows the emotion similarity between individual artists. The modularity 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.

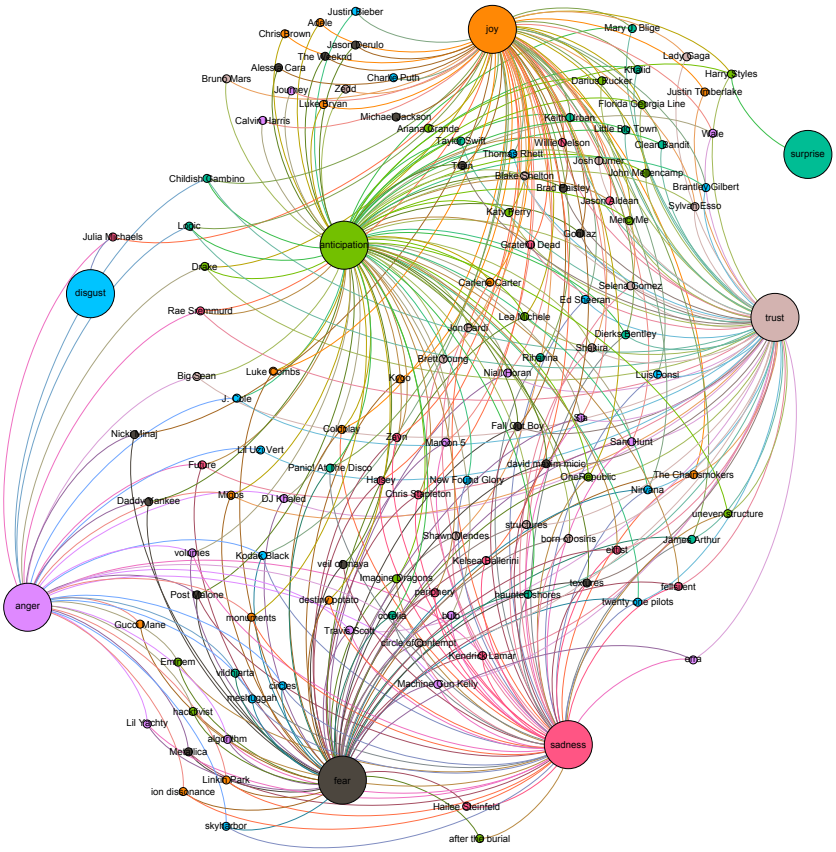

Figure 2: Emotion graph with edge weight greater than 0.37

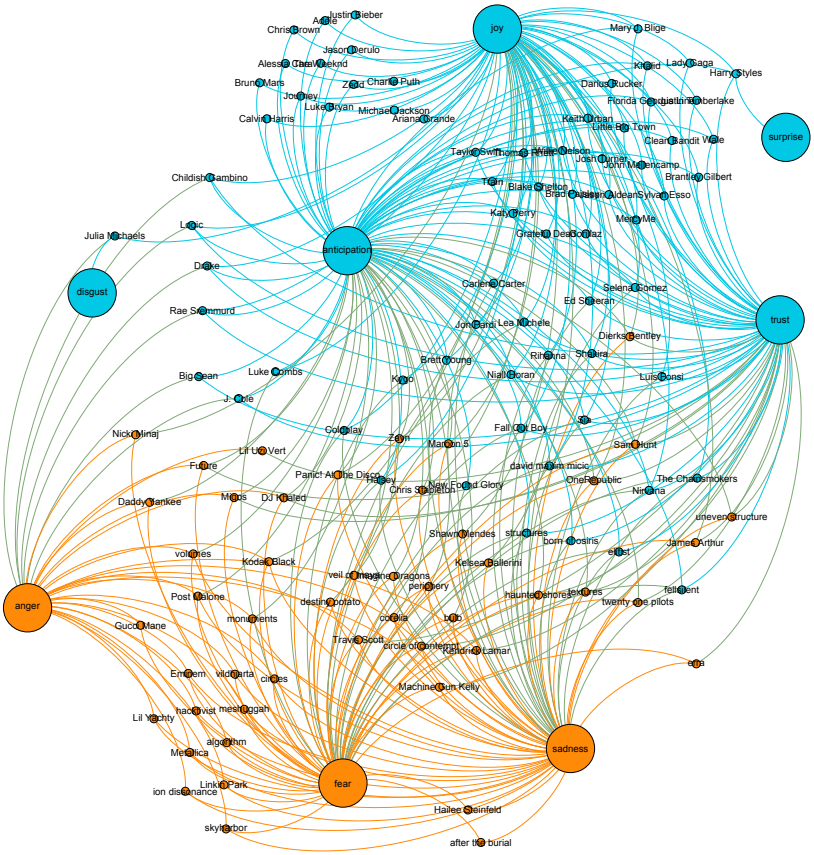

Figure 3: Emotion graph with recalculated modularity

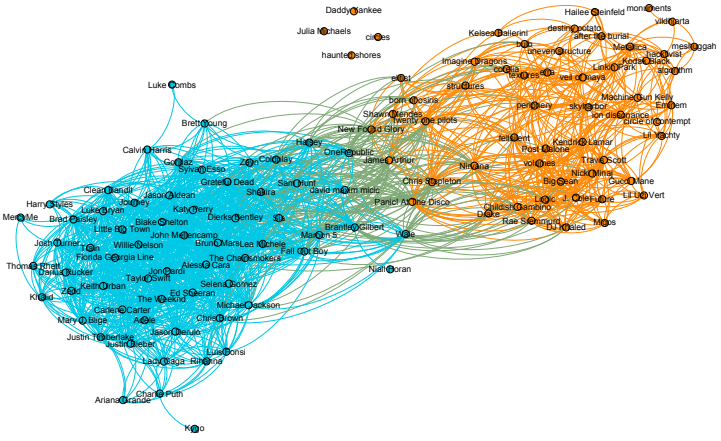

Figure 4: Emotion similarity between artists greater than 0.98

6 Topic Extraction

While grouping artists by emotions they express through their lyrics already provides fascinating results another part of this project was the topic based analysis of lyrics. This chapter describes the used technique and discusses the achieved results.

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. 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 and through experiments we determined ten topics to be appropriate for our dataset.

Generating more topic yielded overlap throughout the topics and creating fewer topics resulted in very packed clusters with much information contained. The amount 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 0.32

Figure 6 shows the partitioning of the graph after calculating the modularity score. The graph is partitioned into four clusters. The first one consists of the topics *young & rich*, and *bitch, fuck 'n money*. The second cluster has the topics *feel real* and *good & bad*. The third cluster contains the topics *time* and *party*. The fourth and biggest cluster is made up of the emotions *breakup*, *girl & boy*, *crazy baby*, and *love*.

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].

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 splitted the songs for each topic into a training and a test set and trained and SVM with the songs' TF-IDF features.

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

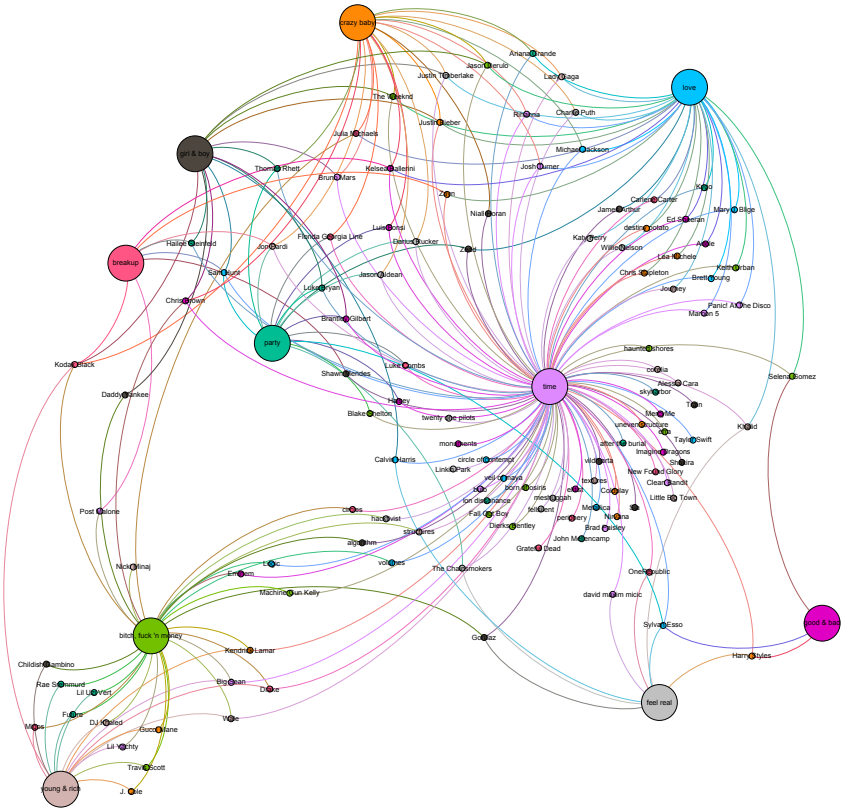

Figure 5: Topic graph with edge weights greater than 0.32

The result of this approach...

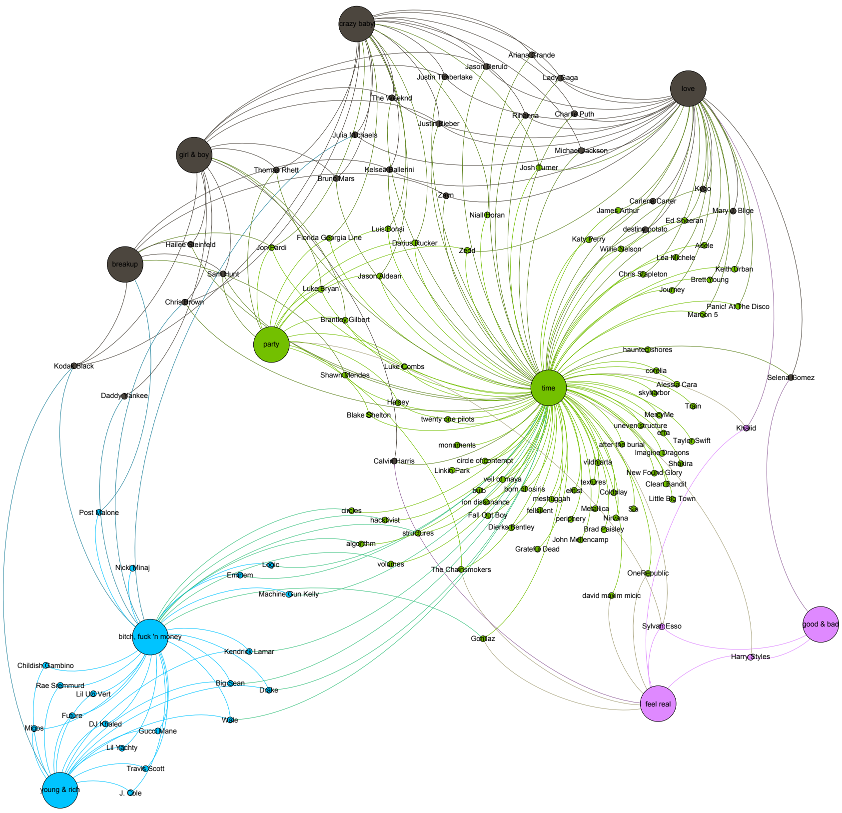

Figure 6: Topic graph with recalculated modularity

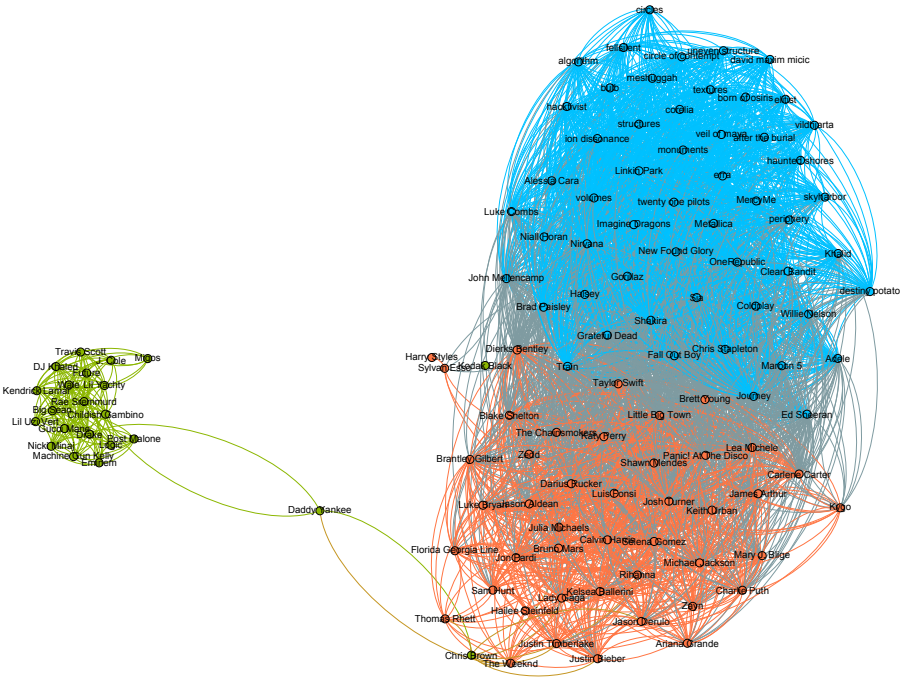

Figure 7: Topic similarity between artists greater than 0.82

7 Other techniques

In this section we want to mention some of the techniques we used that were neglected during the development. Some were already mentioned earlier and some have not yet been discussed at all. While we did decide not to include these techniques in our project no generality should be inferred from this as we rejected some techniques based on lack of experience, time complexity or simply the quality of the API and documentation.

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 non 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 to the user they don't actually help increase our performance in general. Only working with specific parts of speech or named entities distorts the results of emotion mining and is required in topic extraction. 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 yields 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 or 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. While the correlation between such statistics and the topics/emotions lyrics convey this was also not in the scope of this project. Furthermore the statistics could vary greatly within the songs of the same artist as they touch different styles of writing.

The last technique we also applied was Rapid Automatic Keyword Extraction (RAKE). It helps, like named entities, getting 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 have any benefit for the final results.

To sum it up 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

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`

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/`

Sibirtsev, & Schuster. (2014). *Information retrieval and text mining with lyrics.*

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