Web Mining Project: Music Lyrics Sentiment Analysis

Group 8
Martin Böckling
Chih-Yen Ou
Yi-Hsuan Peng
David Probst
Fabio Westphal

Abstract

This report is part of an educational project in the field of web mining. The objective of this work is to build a classifier that predicts the sentiment of a song solely based on its lyrics. Such a classifier might help music services recommend a song based on a desired mood. Additionally, keyword extraction is done to provide more insights into the context of a song. The lyrics data for training and testing was taken from Genius, a popular website hosting song texts and information. To keep it simple, the target classes are whether a song conveys a positive or a negative sentiment. A neutral class is added in an alternative approach. Following a supervised approach, the songs are labeled using the valence score provided by Spotify. It turns out that a classifier using bidirectional encoder representations from transformers (BERT) yields the best performance on these classification task on lyrics sentiment. Nevertheless, performance is limited by the fact that a song's textual data does not necessarily convey the same sentiment as the entire song including acoustical data.

After presenting related works, this report walks through different classification algorithm candidates. The experimental setting regarding the construction of data and models is described before the results are evaluated and discussed.

1 Introduction

Music streaming services like Spotify have drastically changed the way we listen to and interact with music. With artists' work available on demand anywhere at any time, it not only changed our behavior but also the music itself. In times where there seems to be an increased space and need for a musical presence in everyday human life, one could argue that music encodes contemporary lifestyle and zeitgeist. It may affect the human body, impact cognition and behavior, empower physical activity as well as regulate moods and feelings, making music a powerful instrument (Leijonhufvud, 2018; Swaminathan and Schellenberg, 2015; Rentfrow, 2012; Hagen, 2016). Most analyses regarding music streaming in particular do not consider the lyrics' sentiment, so this is where new insights might be discoverable. Especially for recommending songs for a desired mood, knowing the sentiment of a song is beneficial. In the end, such a model could help music services providing better song recommendations to increase usage and customer loyalty.

An underlying assumption is that lyrics are more descriptive regarding sentiment than the melody of a song, which is why we base our song sentiment analysis solely on the song text. Using text mining techniques on lyrics, we seek to decode sentiment and extract key words for each song to understand the structure of the data. The goal is, given just the lyrics of a song, to perform binary and ternary sentiment classification based on the valence score from Spotify. In search of a sophisticated lyrics sentiment model, several models are evaluated on their performance. For the sentiment analysis we consider different approaches in data preprocessing which will be evaluated with the different models we examine in this context. We also use the extracted keywords and the valence score to gain insights into past musical culture and deepen the understanding of how lyrics developed over time. A common challenge in lyrics mining arises from the text being relatively long and informal, which is why sophisticated feature extraction has to be developed.

The project behind this report was conducted as part of the lecture Web Mining at the University of Mannheim. In this report, we will outline the procedure and results of the project. In chapter 2 we will summarize different related works that are relevant for our project. In chapter 3 we will outline the different algorithms used to classify the lyrics sentiment. In particular, classic classifiers such as Support Vector Machine, Multinomial Naive Bayes, Random Forest, and Xgboost, as well as advanced Natural Language Processing (NLP) models with BERT Transformer and XLNet Transformer were used as model architectures. Parallel project tracks aimed to work on one approach each. More information about the underlying data from Genius and how it was preprocessed can be found in chapter 4, together with the particular model configurations we used. In chapter 5.3, the results of the used approaches are compared, evaluated and discussed. After the discussion of the results we will provide in chapter 6 a conclusion of our work and give an outlook on future extensions from our results.

2 Related Work

With regard to the analysis of lyrics there are several other works providing valuable insights. In this section, a few of them are briefly outlined including their impact on our work.

An important model regarding emotions, which is frequently referenced in works about sentiment analysis, is Russell's Circumplex Model of Affect (Russell, 1980). It divides 28 affect words, each representing a distinct human emotion, into four main quadrants by assessing arousal and valence. Emotions like excitement are characterised by high valence and high arousal, whereas contentment is lower in arousal while having a similar valence. Feeling somehow distressed, on the other side, is low in valence and high in arousal, and depression is low in both. These four key categories of emotions can help classifying sentiment, although they are often named differently.

The paper by Gamez and Mercer (2019) describes the most similar approach to ours, aiming at developing a classifier for song sentiment based on lyrics. They retrieve the lyrics data from two different online sources, one with human annotated tags and one with word valence annotations. The selected models are a Naive Bayes classifier, a Convolutional Neural Network, a Recurrent Neural Network and a pretrained BERT model. Notably, a very simple Naive Bayes model is able to beat both CNN and RNN models. It is outperformed only by BERT, which yields an accuracy of 0.78 for a binary sentiment classification task. They also conducted a classification for four emotions, namely happy, angry, sad and relaxed. These emotion categories are from Russel's Valence-Arousal model, which was described above. Interestingly, the authors point out that angry songs are the easiest to classify, which might be due to the frequent use of explicit vocabulary.

Another research group looked into the deep sequential correlation between audio and lyrics (Yu et al., 2019). First, they extract lyrics and audio features separately. The audio is represented as a spectrogram, which is highly dimensional and therefore extracted by a convolutional model. For lyrics' textual data a vectorised representation is created by a Doc2vec model. Both are then mapped to a common semantic space. In the end, this architecture even allows bidirectional retrieval between lyrics and audio: using audio to retrieve lyrics and using lyrics to retrieve audio. A deep correlation between audio and lyrics is evident, which justifies our approach to determine the sentiment only based on one of the two.

A paper by Yang and Lee (2009) proposes a music emotion extraction system (EMO) for music data mining. A basic assumption (Stratton and Zalanowski, 1994) is that lyrics are supposed to focus the listener's attention on particular emotion. This is also an important foundation for our work. In lyrics mining, even more than in regular text mining, many words occur with very low frequency, but have all the more power to distinguish texts in emotion recognition. An online library for music information carries 168 discrete mood categories, showing that music emotion ratings are hardly reducible. Nevertheless, the authors come up with 23 emotion descriptors in their EMO model (Yang and Lee, 2009). Since each social group in music tends to have their own distinct vocabulary, word feature vectors can become very large. This is why, rather than bag-of-words representation, psychological features are used as an effective transformation to reduce feature dimensionality. The Havard General Inquirer[1] covers 11,700 English words and tracks their psychological features, making it handy to identify emotions for each word in lyrics. That way, a large word vector can be transformed into a much smaller emotion vector encoding the incidence of each emotion occuring in the text. A

[1]http://www.wjh.harvard.edu/~inquirer/homecat.htm

decision tree model created based on these feature extraction steps proves successful with an accuracy of 67 %. They note that emotion identification in lyrics is more accurate than distinguishing them based on acoustic features, which is again a validation of our approach.

Another notable approach to music modelling is incorporating lyrics-melody aligned data in a language model (Watanabe et al., 2018). Aiming at offering precise syllable-note alignments for a song, the proposed language model automatically captures the relationship between lyrics and melody. It is able to generate fluent lyrics while maintaining consistency to an input melody. The authors use a standard recurrent neural network language model. They find that combining a limited-scale collection of melody-lyrics alignment data with a far larger collection of lyrics-alone data for training the model boosts the model's competence.

Since lyrics sentiment analysis requires different feature extraction than regular text sentiment classification, a sentiment vector space model (Xia et al., 2008) is applied as a document representation model in another relevant work. This approach considers only sentiment-related words by extracting them from a semantic dictionary. Additionally, words are contextually connected with negations and modifiers. They can strengthen, weaken or reverse a word's meaning and are found in a word's neighborhood using a sliding window. We considered three preceding and three succeeding words for the sliding window to find related negations and modifiers. A word forms a semantic unit with its modifier and negation, should there be any. With this approach, only relevant features are extracted and, equally important, disambiguated. The paper (Xia et al., 2008), besides showing that the Sentiment Vector Space Model allows higher F1-Score, also supports our claim that lyric is better than audio in song sentiment detection.

3 Methodology and Algorithms

3.1 Baseline

An obvious baseline model in classification is just using the most frequent group as the only solution, so that any more sophisticated algorithm should be able to beat this very basic one in order to be considered as providing any added value (Gamez and Mercer, 2019). In this sense, we have 54% and 40% for negative valence respectively in binomial classification and in multinomial 3-classes classification.

3.2 Keyword Extraction

3.2.1 TF-IDF

TF-IDF (Term Frequency - Inverse Document Frequency) is a statistical weighting scheme for n-grams. Although it is a rather simplistic approach, it usually leads to a good first baseline model in a lot of different areas (Firoozeh et al., 2020). For keyword extraction, the TF-IDF weights can be understood as a measure of uniqueness of a term towards a sentence, hence describing a keyword. Those weights are calculated as follows:

$$tf_{t,d} * idf_t \tag{1}$$

$tf_{t,d}$ is the term document frequency of the term t and the document d and idf_t is the inverse document frequency of term t, defined as:

$$idf_t = log\frac{N}{df_t} \tag{2}$$

with N being the total number of documents in the collection and df_t being the document frequency containing the respective term (Manning et al., 2008)[p. 119].

3.2.2 KeyBERT

The Bidirectional Encoder Representation from Transformers (BERT) is a language representation model architecture that was introduced in 2018 (Devlin et al., 2019). The model architecture builds upon the use of transformers, which is based on the work of Vaswani et al. (2017). KeyBERT is an easy-to-use but powerful, pretrained implementation of BERT embeddings and cosine similarities. The BERT embeddings are first created on a document level. Afterwards word embeddings are extracted and the cosine similarity is calculated between the word and the document embeddings to find the best fitting words for each document (Grootendorst, 2020).

More detailed insights into BERT, also in the context of sentiment analysis, are given in subsection 3.4.

KeyBERT and TF-IDF are going to be used during this project to get a better understanding of the data (see subsection 4.4).

3.3 Classic Algorithms

Support Vector Machine (SVM) has been extensively and successfully used as a sentiment learning approach (Colas F., 2006). Due to the fact that text analysis task typically represent instances by very high dimensional but sparse feature vectors, SVM is particularly helpful as it learns a classification hyperplane in feature space which has the maximal distance (or margin) to all the training examples (except a small number of examples as outliers). Consequently, on classification tasks SVM tends to have better generalisation capability on unseen data than other distance- or similarity-based learning algorithms such as k-Nearest-Neighbour(KNN) or Decision Trees (Li et al.).

In the process of selecting an algorithm, we use SVM as our first model approach. Algorithms such as Random Forest, Gradient Boosting and Decision Tree also tend to have a good performance when it comes to opinion mining in music lyrics (Ahuja and Sangal, 2018). Therefore, we will conduct and optimize the classic classifiers, including SVM, Multinomial Naive Bayes, Stochastic Gradient Descent, Random Forest, and Xgboost. After training, one optimized model that performs the best is selected out of these, and then the model performance is compared with the following advanced NLP algorithms.

3.4 BERT

As mentionend in subsubsection 3.2.2, BERT is a language represenation model making use of transformers. For the BERT framework, the authors included two different steps for using it on different NLP tasks. The first step in the framework is *pretraining* and the second step the authors took is *finetuning* to specialize the pretrained model to a particular NLP task. For every fine-tuned model there is an adaption to the input and output of the model. The architecture of BERT contains multiple layers of bidirectional transformer encoder with different model sizes. Devlin et al. (2019) tested the performance of the model on two different model sizes: $\text{BERT}_{\text{base}}$ with twelve transformer blocks and $\text{BERT}_{\text{large}}$ with 24 transformer blocks.

When the paper was published, the proposed architecture outperformed other models on the GLUE test (Devlin et al., 2019). As the BERT architecture provides a good performance on a variety of different datasets, we will evaluate the performance of the model on the lyrics dataset described in section 4.1.

As the goal of the report is to classify the sentiment of a song based on the lyrics, we will use a fine-tuned BERT model on classification. Due to the limited resources available for the project we will use the $\text{BERT}_{\text{base}}$ model which is not as computationally expensive as the $\text{BERT}_{\text{large}}$ model. A detailed run configuration of the model can be found in section 4.5.2.

3.5 XLNet

XLNet is a generalized autoregressive (AR) method, an enhanced BERT version, which avoids some limitations in BERT. For example, BERT use [MASK] token during pretraining, but the artificial symbol does not exist in real data during finetuning. This pretrain-finetune discrepancy problem may affect the model performance. The other limitation on BERT is the assumption of independence of the predicted tokens from each other. In many situations, however, most words are correlated to each other (Yang et al., 2019).

XLNet uses Permutation Language Modeling (PLM) to learn bidirectional contexts by all sequences of factorization order which removes the assumption of token independence (Yang et al., 2019). The model architecture builds up by two-stream self-attention which contains content stream attention, capturing the bidirectional context, and query stream attention, replacing the role of [MASK] token in BERT (Yang et al., 2019). For the long text understanding, XLNet use Transformer-XL (Dai et al., 2020) which makes XLNet has a long term dependency and becomes one of the language model without sequence limit. In many NLP application, XLNet outperforms BERT on a wide variety of problems including sentiment analysis, text classification, natural language inference (Yang et al., 2019). As XLNet has substantial improvement on various task, we will compare the performance of the model with other algorithms on the lyrics datasets.

4 Experimental Setting

4.1 Data

For Spotify song metadata, we use a dataset from Kaggle which contains more than 175,000 songs from 1921 to 2021 (Ay, 2021). It is generated from the Spotify Web API[2] and includes object data such as artist, release date and name of the song, as well as numerical data such as danceability, energy (ranging from 0.0 to 1.0), year (ranging from 1921 to 2020), popularity (ranging from 0 to 100) and valence. Valence is a measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry) (Spotify AB, 2021). This metric will be further used as the label in our sentimental task. To ease the computational pressure, we only consider songs from later than 2001, which gives us 41,205 songs. Figure 1 shows that the density distribution of valence is close to the normal distribution.

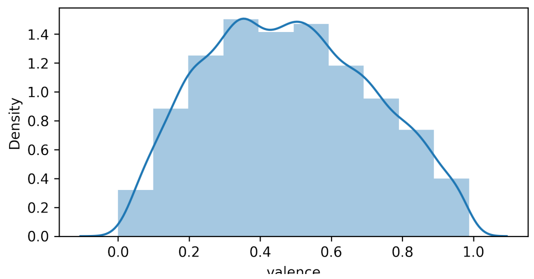

Figure 1: Density distribution of valence

For text mining, we need complete lyrics data for in-depth analysis. We generate lyrics data from Genius.com via the Genius API (Genius Media Group Inc., 2021) and use the corresponding python package Lyricsgenius[3] to download the lyrics. Then we match the lyrics with the Spotify metadata according to artist and name of the song. We fetch two versions of lyrics: one with space between each line (*all_songs_dataset*), and the other one with a comma separating each line (*all_songs_comma_dataset*). We generate the latter one in order to experiment the influence of punctuation on the advanced NLP models, as such raw data is usually crawled from the web.

[2]https://developer.spotify.com/documentation/web-api/

[3]https://docs.genius.com/#/getting-started-h1

4.2 Evaluation

To evaluate the different models, it is necessary for the different approaches of the project to be evaluated on the same metrics. Another important point for the evaluation is that all the different models are trained on similar parameter settings to be able to compare the results properly. The evaluation of the final result will be conducted on a test dataset, which has a fraction of 30% of the total dataset for classic algorithm (see section 4.2). For the binary sentiment classification, the test dataset includes 3,436 rows, and for the three class sentiment task there are 3,440 rows. Both fractions of test dataset in BERT and XLNet use 10% of the total dataset in both two-class and three-class classification. The metrics for the evaluation of the sentiment classification used in the project are accuracy, precision, recall and F1 Score. Precision, recall and F1 score will be averaged on the macro average to also be able to compare the two- and three-class sentiment analyses. As the distribution of the different classes is balanced in both cases, the macro average is a valid option for a comparison between the different classes (Grandini et al., 2020).

4.3 Preprocessing

To prepare the data, we first need to clean it up. For both datasets, *all_songs_dataset* and *all_songs_comma_dataset*, we found that instead of lyrics, there are some prose descriptions returned by API. Therefore, we first filter out those wrong lyrics by keeping lyrics which contain [Intro], [Chorus] and [Verse] tags and which do not contain ‘Last updated’, ‘feat’ and ‘Version’. Also, we cleanse the song titles which sometimes contain words like ‘Unreleased’, ‘Remaster’, ‘Live’, ’Demo’ and ‘Mixed’, because they are always matched with wrong lyrics data and might occur due to an API fetching issue. After filtering out the wrong lyrics, we then remove duplicate songs based on song title, artist and year. Next, we filter out non-English lyrics since we only aim to analyze English lyrics. Finally, we normalize slang words based on a dictionary[4] which lists 175 slang words and their meaning.

We use two different versions of the dataset: *all_songs_dataset* and *all_songs_comma_dataset*.

[4]https://www.smart-words.org/abbreviations/text.html

In *all_songs_dataset*, we remove brackets such as [Intro], [Verse] and [Chorus] which repetitively appear in Lyrics. In *all_songs_comma_dataset*, we keep the words inside brackets but replace the brackets with dots to mark the end of a logical part in the lyrics. If there is a dot or comma at the beginning of the lyrics, we replace it with a space. If there is sequence of commas, which represents a line break, we replace a dot instead. Second, we convert all lyrics text to lower case and remove punctuation between lines in the former dataset, whereas keeping the raw lyrics in the latter one.

In order to apply the lyrics to classic classifiers, we need to further generate tokens from lyrics. In *all_songs_dataset*, we then remove stop words, which have little value, and lemmatize the text to avoid the words that are semantically similar but in different syntactical structure. By using the Spacy package[5], we are able to generate parts-of-speech tags, i.e. for each word if it is a verb, noun, adverb, adjective or corpus. We store this tagged dataset as all_songs_corpus.

After data cleaning and preprocessing, we are now ready to cut the dataset into different classes for our sentiment classification task. For binary classification, we remove the rows where valence is equal to 0.5 and label the data as negative if valence is below 0.5, and positive if it is above 0.5. This gives us a total of 11,455 songs, with 6,207 negative and 5,248 positive examples. For three-class classification, we keep the songs where valence is equal to 0.5. We separate the dataset into negative if valence is below 0.4, and positive if valence is above 0.6, and neutral if valence is between 0.4 and 0.6. This gives us a total of 11,469 songs, with 4,558 negative, 3,325 neutral and 3,586 positive examples.

For the keyword extraction the same text cleaning, normalisation and filtering steps are performed.

Class	Sentiment	Valence	#	Total
2	Negative	[0, 0.5)	6207	11455
	Positive	(0.5, 1]	5248	
3	Negative	[0, 0.4)	4558	11469
	Neutral	[0.4, 0.6]	3325	
	Positive	(0.6, 1]	3586	

Table 1: Overview of sentiment class definition based on valence score and numerical distribution (#)

[5]https://zenodo.org/record/4900666#.YMfAOy8RqJ8

4.4 Data Understanding

Before analyzing the data, we take a closer look at the lyrics data between 1960 and 2021.[6]

The density of words per song is displayed in figure 2. It very clearly shows that only very few of the songs have less than 256 words. The number of words does not exactly translate to the number of tokens in approaches like BERT but it becomes obvious that a solution might be needed to cover those longer texts. We therefore decided to use a sliding window approach which is explained in more detail in the subsection 4.5.

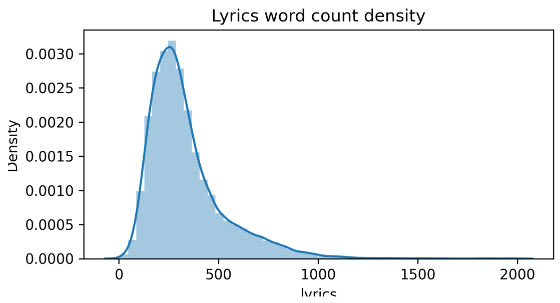

Figure 2: Lyrics word count density

As mentioned in subsection 4.3, the valence score is used as a measure to classify songs, describing the positivity of a song. When plotting the mean valence between 1960 and 2021, an interesting downward trend is noticeable (see figure 3). It seems that artists over time have become more negative in what they are saying.

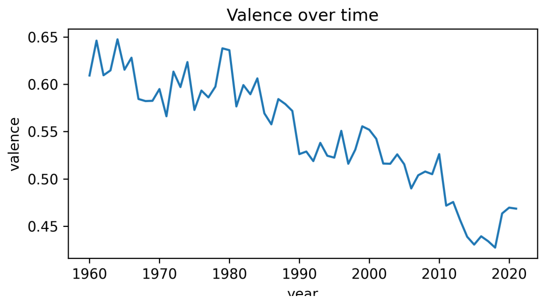

Figure 3: Mean valence per year

This trend is also visible in the keywords. Due to its simplicity and comparable results, we picked TF-IDF as a first approach. But as lyrics contain a lot of word repetitions in only a handful of documents, TF-IDF performs poorly on our

[6]Due to the computational effort of collecting the lyrics data and the limited scope of this report we have only taken a look at this period.

use case. The keyword "kanye" for example received a smoothed TF-IDF weight of 0.982. We traced this back to the song "I love Kanye" by Kanye West containing 24 appearances of the word "kanye": "I miss the old *Kanye*, straight from the Go *Kanye* Chop up the soul *Kanye*, set on his goals *Kanye* I hate the new *Kanye*, the bad mood *Kanye* The always rude *Kanye*, spaz in the news *Kanye* I miss the sweet *Kanye*, chop up the beats *Kanye*..." [7]

We therefore only rely on the results extracted by KeyBERT, as it allowed a more meaningful extraction of keywords and gave a better understanding of the data. The top seven stemmed keywords (incl. count) per decade are visible in figures 4 - 10 in appendix A. They also highlight that swearing and slang words are becoming more popular while words like "love" and "kiss" are declining in importance. Further analysis on this topic is out of scope for this project, but it might phrase some interesting future research topics.

4.5 Model Configuration

4.5.1 Classic Algorithms

On our first model approach, SVM, we implement TF-IDF vectorizer and SVM classifier using Pipeline package. Next, we conduct grid search cross validation on SVM parameters, tuning C, gamma and kernel setting.

With the performance of SVM in mind, we try to select the best model among classic algorithms, including Multinomial Naive Bayes, Stochastic Gradient Descent, Random Forest and Xgboost. With the help of PipelineHelper[8] and RandomizedSearchCV packages[9], we are able to search parameters for different models, and it returns the optimized classifier with the corresponding parameter setting.

For classic algorithms, we use corpus dataset as our training data, since it has been further removed stop words and lemmatized text. Our configuration settings implement 70% of the dataset as training data and 30% as testing data, and classify songs into 2 or 3 classes on corpus dataset.

[7]https://genius.com/Kanye-west-i-love-kanye-lyrics

[8]https://github.com/bmurauer/pipelinehelper

[9]Due to the computational stress, we conduct RandomizedSearchCV with 150 parameter candidates

4.5.2 BERT and XLNet

In XLNet, we use Python Huggingface library with *xlnet-based-cased* model and use *XLNetForSequenceClassification* to add a single classification layer on top. The output layer will be two or three classes. Due to resource limitation, we train on one GPU which gives us a maximum sequence length of 256, an Adam optimizer, and a batch size of 16, maximum available number (Wolf et al., 2020).

Similar to XLNet we use for BERT also the Huggingface library with the pretrained *bert-based-uncased* model. As the base model is not specialized on a specific downstream task we make use of the *AutoForSequenceClassification* model class which adds a sequence classification head to a preloaded model, in our case the BERT. Depending on the scenario looking the calculation would be two or three classes. For the optimization during training we use the Adam optimizer with a learning rate of 2e-5 and an epsilon value of 1e-8. Due to the ressource limits the model settings were set to a maximum sequence length of 256 and a batch size of 32 (Wolf et al., 2020).

With the limitation of maximum text length, at 256, most of the words are truncated which strongly influences the model accuracy. Thus, we use a sliding window as discussed in Wang et al. (2020) and Ding et al. (2020) with a window size of 200 and concatenate 50 words from previous window which avoids data loss.

Our experiment implements a variety of configuration settings which are with or without sliding window, and classifies dataset into two or three classes on both datasets. For each dataset, we use 80 % of it as training data which takes about 13 minutes to almost an hour per training epoch on different cases. Then we split the remaining 20% into a 10 % validation dataset and 10 % testing dataset.

5 Evaluation and Discussion

As outlined in section 4.1, the research was taken under different underlying datasets. The main differentiator of the two datasets is the different number of classes on which the models were finetuned. In the following two sections the different results will be discussed and evaluated.

5.1 Two classes

For the two-class sentiment, the results displayed in table 2 were achieved with the approaches discussed in chapter 4. Every used model outperformed the base model so had better results in the accuracy then 0.5419. The model with the highest accuracy value was for the two sentiment classes in both datasets the finetuned BERT model. The highest accuracy score was achieved on the *all_songs_dataset* including the sliding window with an accuracy of 0.7419. Furthermore, the BERT model finetuned on the preprocessed *all_songs_comma_dataset* using the sliding window approach achieved the second highest accuracy score of 0.7307.

The XLNet model has with 0.6027 the lowest accuracy score for the overall performance across the different models on the two classes sentiments. The accuracy score was achieved with the *all_songs_comma_dataset* without the splitting of the lyrics with a sliding window. Similar to the BERT model the XLNet model has the highest accuracy score on the *all_songs_dataset* with the use of a sliding window and has achieved an accuracy score of 0.7283. Furthermore the result for XLNet also improved on the result of the *all_songs_comma_dataset* with the use of a sliding window from an accuracy of 0.6027 to 0.6997.

The SVM was able to reach for the *all_songs_dataset* an accuracy above 0.6. By not adjusting the default parameter given by the scikit-learn package for SVM (Pedregosa et al., 2011). An improvement of the accuracy can be achieved by using the Grid Search optimizer to get the optimal parameter for the vectorizer and get the optimal model as discussed in 3.3. The accuracy for the optimized vectorizer and model improved from 0.61 to 0.63. For the vectorizer the optimal vectorizer determined is the Count Vectorizer. For the max_df the optimized parameter is increased from one to ten. Also the min_df is increased from one to 6. For the model selection the RandomForest Classifier is selected. The parameter bootstrap is set from the default setting of True to False. The max_features parameter is changed from auto to sqrt and the min_sample_leaf from one to two. The parameter min_samples_split changed the run value from two to five and the n_estimators for the trained number of trees increased from 100 to 800.

Precision and recall do not differ much from the accuracy scores of the models. A detailed overview of the different metrics can be found in table 2. In the section 5.2 the results for the three class classification will be presented.

5.2 Three classes

The results of all different models for the three classes data show a higher accuracy score than the accuracy measures from our defined baseline approach with 0.3974. The approach discussed in section 3.3 achieves on the *all_songs_dataset* an accuracy of 0.44 and a precision and recall of both 0.41. Using the optimizer, an accuracy of 0.5 and a precision of 0.51 and recall of 0.45 was achieved. The model with the best accuracy was achieved on the CountVectorizer model and the RandomForest, the same vectorizer and model selected in 5.1. The max_df parameter for the maximum threshold for the maximum document frequency was optimized from one to 30. The minimum document frequency is optimized to four from one. Furthermore, the parameters of the Random Forest model were optimized using the Grid Search optimizer. The maximum number of features which are considered to calculate the best split is determined by the square root of the total number of features. The minimum number of samples that are required for an internal split is optimized from two to ten. The number of trees constructed in the Random Forest is increased from 100, which is the default value in the scikit-learn package, to 1400 (Pedregosa et al., 2011).

The BERT model is tested on different datasets as discussed in the subsection 3.4. On the *all_songs_dataset* the BERT model achieved an accuracy of 0.4717 and 0.48 on the precision and recall. With the truncation of the lyrics by using a sliding window BERT achieved an accuracy of 0.5908 with a precision and recall value of 0.58. The BERT model on the *all_songs_comma_dataset* has an accuracy value of 0.4665 and a precision and recall value of 0.45. With the use of a sliding window the BERT model had an accuracy value of 0.5667 and a recall value of 0.56 and a precision value 0.55.

XLNet finetuned on the *all_songs_dataset* achieved an accuracy score 0.4339 and a precision and recall value of each 0.43. When the lyrics are truncated with a sliding window the accuracy achieved in 0.593 and a precision and recall of 0.59. With the *all_songs_comma_dataset* the XLNet model achieved an accuracy score of 0.4435 and an precision and recall value of 0.44. By splitting the dataset with the help of a sliding window XLNet achieved an accuracy of 0.5339. The recall and precision value on the *all_songs_comma_dataset* was 0.55. In the table 3 the results of the different models are displayed according to the base of the dataset.

5.3 Discussion of results

The results presented in section 5.1 and 5.2 are now discussed critically and compared to the results of published papers. The different settings for the model part show different effect on the model performance and the related metrics.

Among the different model settings, the models finetuned on the *all_songs_dataset* achieved higher accuracy, precision and recall value then on the *all_songs_comma_dataset*. The main difference between the different versions includes the different preprocessing approaches discussed in section 4.3. The major difference between both datasets is that the punctuation and commas are added in the *all_songs_comma_dataset*. In our experimental setting we examined that the *all_songs_dataset* achieved a slightly higher accuracy, precision and recall as the respective model setting on the *all_songs_comma_dataset*. One exception is the BERT model for the two classes sentiment prediction without the sliding window where the accuracy is slightly higher on the *all_songs_comma_dataset* compared to the the *all_songs_dataset*. The observations made in this report regarding the presence or absence of punctuation and commas match with the results of a related work where a BERT model, finetuned on models without punctuation, also had a slight improvement in accuracy (Ek et al., 2020).

It is also notable that the sentiment of a song is not only determined by the lyrics, but also the melody plays an important role for the perceived sentiment. Our hypothesis that the lyrics of a song play an important role in the sentiment of a song was proven correct by the performance of the created models.

If we compare the different model results we see that the BERT model mostly scored the highest accuracy except for one case in which XLNet achieved a slightly higher accuracy score. This was achieved for the three class sentiment on the *all_songs_dataset* with the sliding window truncation. This result is in contrast to other works as mostly XLNet had a slightly better performance than BERT (Yang et al., 2019; Liu et al., 2019). Furthermore the finetuned XLNet and BERT models achieved a higher accuracy than the SVM or the optimized Random Forest. This observation does not stand in contrast to the current evaluation of the different models.

As we deal with long texts in the lyrics, there are different strategies to handle long text. As BERT and XLNet have a limitation for the maximum tokens which can be used in the finetuning of the models, there is a need to reduce the tokens size. As discussed in section 4.3 we used the truncation of tokens and sliding windows to split the lyrics data. For the truncation, the first N tokens of the generated tokens are considered during finetuning of the model (Sun et al., 2020). For the sliding window approach we defined a window which allows overlaps between the different splits (Wang et al., 2020). In all of the evaluated models the results improved significantly by splitting the lyrics based on the sliding window. Our results match with the research regarding handling long text where the performance of the models increased with the introduction of sliding windows (Wang et al., 2020; Ding et al., 2020). Using the sliding window approach, the information which would have been thrown away is then included in the finetuning of the model.

When comparing the model-wise performance between the two-class and the three-class problems, we see that the two classes sentiment achieved a higher accuracy score compared to the three classes sentiment prediction. This observation is similar to the results achieved in a study by Gamez and Mercer (2019). For the multiclass sentiment prediction the authors achieved a lower performance on the different metrics similar to the results observed in our report (Gamez and Mercer, 2019). Furthermore, it must be said that the accuracy from their BERT model was with 0.7825 higher than the accuracy of 0.7419 achieved in our work. The possible origin of this discrepancy and the better results in Gamez and Mercer (2019) could be the manually labeled valence score and different datasets that were used.

6 Conclusion and Outlook

Our report shows that different approaches in handling long texts lead to similar results as expected from research in other text analytics subdomains (Ding et al., 2020). Yet, some tweaks that worked for others do not lead to an improvement in our work, for example applying the structure given in the *all_songs_comma_dataset*, which even decreases the measurement scores for the evaluation of our models. The idea of incorporating the structure of a song to the lyrics we analyze does not yield the result we expected with regard the work by Gamez and Mercer (2019).

For future research, implementing CogLTX (Cognize Long TeXts) to apply BERT to large texts could be tested on the used lyrics data to evaluate its impact. In the published paper the results show an improvement in model performance compared to the sliding window approach and tackles the challenge around the lack of long-distance attention. (Ding et al., 2020)

A proper next step could be extending the classes and predicting more precise sentiments, starting with the four main categories in Russell's model (see section 2), and eventually extending it to more descriptors similar to the EMO model (Yang and Lee, 2009). This would enable more targeted song recommendations.

To improve the overall performance, feature extraction could be aligned more strictly with the sentiment vector space model (Xia et al., 2008) (see again 2).

The decline of positivity in songs (see figure 3) and the increase in violent and slang keywords (see 4 - 10 in appendix A) highlight the importance of text-based sentiment analysis models and might be interesting research topics to apply those enhanced models.

But since music consists not only of lyrics, but also acoustics features including melody, it is a high dimensional problem. After all, acoustical data may be taken into account to provide a holistic sentiment analysis, although we learned that lyrics data alone gives more insights regarding sentiment than audio data alone. Learning a model with both aspects, similar to what was done by Yu et al. (see 2, could therefore be a logical next step.

References

Mahesh Ahuja and A. L. Sangal. 2018. Opinion Mining and Classification of Music Lyrics Using Supervised Learning Algorithms. *ICSCCC 2018 - 1st International Conference on Secure Cyber Computing and Communications*, pages 223–227.

Yamac Eren Ay. 2021. Spotify dataset 1921-2020, 160k+ tracks.

Brazdil P. Colas F. 2006. Comparison of svm and some older classification algorithms in text classification tasks. In *Artificial Intelligence in Theory and Practice pp 16*, pages 169–178. Springer, Boston, MA.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2020. Transformer-XL: Attentive language models beyond a fixed-length context. *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 2978–2988.

Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1(Mlm):4171–4186.

Ming Ding, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. CogLTX: Applying BERT to Long Texts. *Advances in Neural Information Processing Systems*, 33(NeurIPS):12792–12804.

Adam Ek, Jean-Philippe Bernardy, and Stergios Chatzikyriakidis. 2020. How does Punctuation Affect Neural Models in Natural Language Inference. *Proceedings of the Probability and Meaning Conference (PaM 2020)*, (PaM):109–116.

Nazanin Firoozeh, Adeline Nazarenko, Fabrice Alizon, and Béatrice Daille. 2020. Keyword extraction: Issues and methods. *Natural Language Engineering*, 26(3):259–291.

David Gamez and Joseph Mercer. 2019. Sentiment Analysis of Songs Lyrics An NLP approach using deep learning. pages 1–6.

Genius Media Group Inc. 2021. Genius api documentation.

Margherita Grandini, Enrico Bagli, and Giorgio Visani. 2020. Metrics for Multi-Class Classification: an Overview. pages 1–17.

Maarten Grootendorst. 2020. Keybert: Minimal keyword extraction with bert.

Anja Hagen. 2016. *Music Streaming the Everyday Life*, pages 227–245.

S. Leijonhufvud. 2018. *Liquid Streaming: The Spotify Way To Music (PhD dissertation)*. Luleå University of Technology.

Yaoyong Li, Kalina Bontcheva, and Hamish Cunningham. Adapting svm for natural language learning: A case study involving information extraction.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. (1).

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Scoring, term weighting, and the vector space model*, page 100–123. Cambridge University Press.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Peter Rentfrow. 2012. The role of music in everyday life: Current directions in the social psychology of music. *Social and Personality Psychology Compass*, 6.

James Russell. 1980. A circumplex model of affect. *Journal of Personality and Social Psychology*, 39:1161–1178.

Spotify AB. 2021. Spotify for developers web api reference.

Valerie N. Stratton and Annette H. Zalanowski. 1994. Affective impact of music vs. lyrics. *Empirical Studies of the Arts*, 12(2):173–184.

Chengyu Sun, Liang Hu, Shuai Li, Tuohang Li, Hongtu Li, and Ling Chi. 2020. A review of unsupervised keyphrase extraction methods using within-collection resources. *Symmetry*, 12.

Swathi Swaminathan and E. Schellenberg. 2015. Current emotion research in music psychology. *Emotion Review*, 7:189–197.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017-December(Nips):5999–6009.

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 2020. Multi-passage BERT: A globally normalized BERT model for open-domain question answering. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, (1):5878–5882.

Kento Watanabe, Yuichiroh Matsubayashi, Satoru Fukayama, Masataka Goto, Kentaro Inui, and Tomoyasu Nakano. 2018. A melody-conditioned lyrics language model. *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1:163–172.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yunqing Xia, Linlin Wang, Kam-Fai Wong, and Mingxing Xu. 2008. Sentiment vector space model for lyric-based song sentiment classification. In *HLT-Short '08: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies*, page 133–136. Association for Computational Linguistics.

Dan Yang and Won Sook Lee. 2009. Music emotion identification from lyrics. *ISM 2009 - 11th IEEE International Symposium on Multimedia*, pages 624–629.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, 32(NeurIPS):1–18.

Yi Yu, Suhua Tang, Francisco Raposo, and Lei Chen. 2019. Deep cross-modal correlation learning for audio and lyrics in music retrieval. *ACM Transactions on Multimedia Computing, Communications and Applications*, 15(1):1–16.

A Appendices

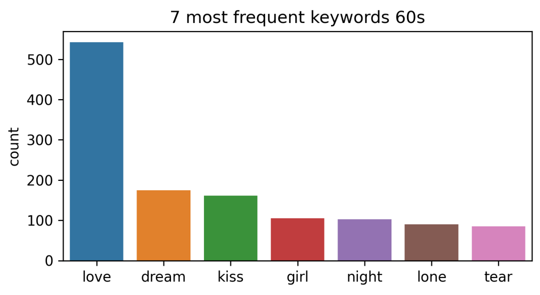

Figure 4: Seven most frequent keywords in songs from the 1960s

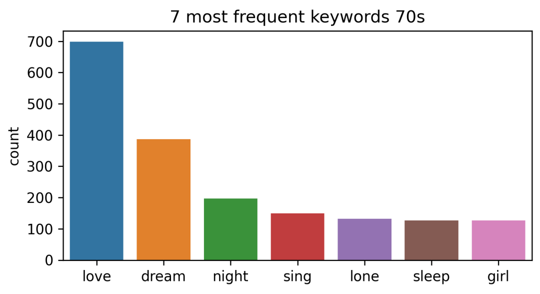

Figure 5: Seven most frequent keywords in songs from the 1970s

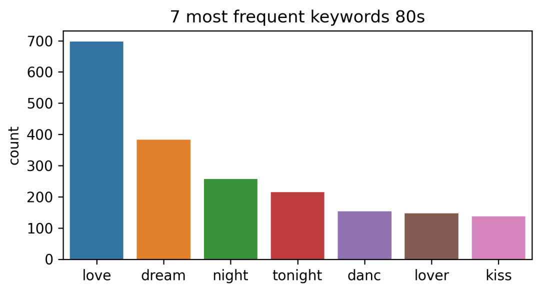

Figure 6: Seven most frequent keywords in songs from the 1980s

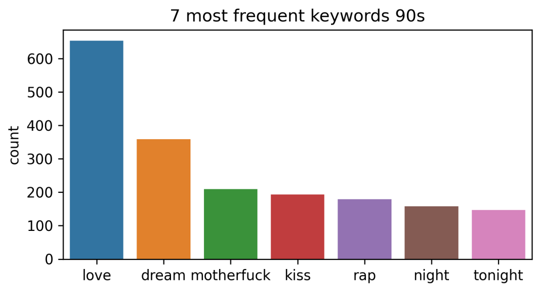

Figure 7: Seven most frequent keywords in songs from the 1990s

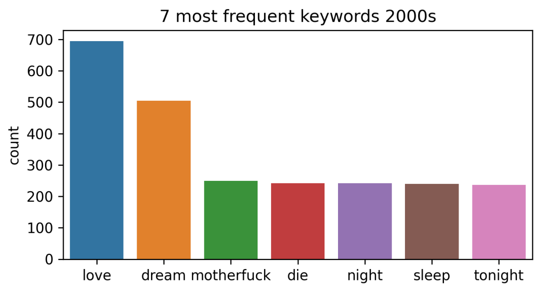

Figure 8: Seven most frequent keywords in songs from the 2000

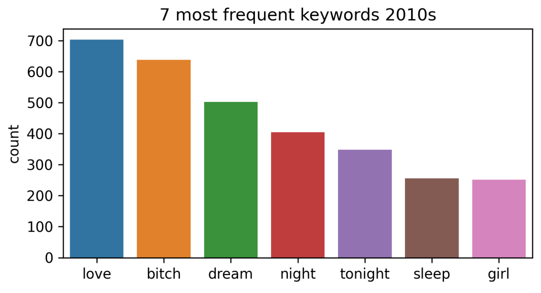

Figure 9: Seven most frequent keywords in songs from the 2010

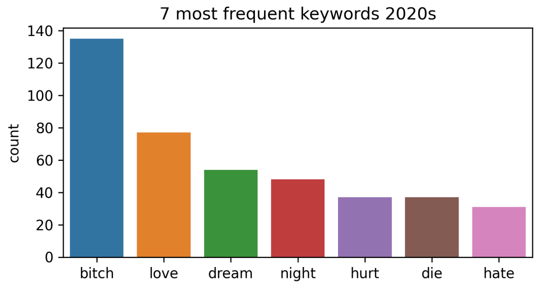

Figure 10: Seven most frequent keywords in songs from the 2020s

B Supplemental Material

You may find our project code and used data mentioned in the paper on GitHub: https://github.com/MartinBoeckling/WebMiningProject-2021-Group08

Model	Dataset	Sliding Window	Metrics			
			accuracy	precision	recall	F1
Baseline	all songs dataset	False	0.5419	0.5419	0.5419	0.5419
SVM	all songs corpus	False	0.61	0.6	0.6	0.6
Random Forest	all songs corpus	False	0.63	0.63	0.62	0.63
BERT	all songs dataset	False	0.6308	0.63	0.63	0.63
		True	0.7419	0.74	0.74	0.74
	all songs comma dataset	False	0.6393	0.63	0.63	0.63
		True	0.7307	0.71	0.7	0.705
XLNet	all songs dataset	False	0.625	0.63	0.62	0.625
		True	0.7283	0.73	0.73	0.73
	all songs comma dataset	False	0.6027	0.6	0.6	0.6
		True	0.6997	0.7	0.7	0.7

Table 2: Model results for two classes sentiment

Model	Dataset	Sliding Window	Metrics			
			accuracy	precision	recall	F1
Baseline	all songs dataset	False	0.3974	0.3974	0.3974	0.3974
SVM	all songs corpus	False	0.44	0.41	0.41	0.41
Random Forest	all songs corpus	False	0.5	0.51	0.45	0.4781
BERT	all songs dataset	False	0.4717	0.48	0.48	0.48
		True	0.5908	0.58	0.58	0.58
	all songs comma dataset	False	0.4665	0.45	0.45	0.45
		True	0.5567	0.56	0.55	0.555
XLNet	all songs dataset	False	0.4339	0.43	0.43	0.43
		True	0.593	0.59	0.59	0.59
	all songs comma dataset	False	0.4435	0.44	0.44	0.44
		True	0.5539	0.55	0.55	0.55

Table 3: Model results for three classes sentiment