Information Retrieval and Text Mining Practical Assignment Report: Sentiment Analysis in Rock Music

Dogan Aykas

27 May 2021

1 Introduction

The chosen topic for the practical assignment is sentiment analysis in the history of rock music. It has several phases which will be discussed in greater detail as separate chapters: data collection, data preprocessing, classification and visualizations of different analysis. The code of the assignment can be found in the zip file. The code is implemented in Python and uses several libraries like PyTorch , NLTK, Pandas and lyricsgenius which will be explained in greater detail with the related sections.

2 Data Collection

Data collection is one of the most significant parts in sentiment analysis from text, since the classification process requires annotated data. Unfortunately, I was unable to find a list of rock lyrics where each element is labelled with respect to the sentiment of the song. Therefore, data collection will require a bit more work compared to using an annotated dataset. The related codes to this section can be found in "Data Preperation.ipynb" jupyter notebook file.

2.1 Spotify's History of Rock Dataset

The aim of this practical assignment is to analyse the sentiments in rock music with a few adaptations. Since I could not find a ready dataset for this goal, there is the need to gather one. There exists a dataset in kaggle (11) originally acquired from Spotify API, which include 5484 popular rock songs. In the dataset, there are several features originally retrieved from Spotify's API: name of the song, name of the artist, release date. In addition, there are several features that come from Spotify for assigning a score to each song in relation to its danceability, acousticness, energy, valence, etc. Although these features mainly incorporate each song's audial properties into account, I decided to use some of them as labels for the classification task. From these features, the scores for

energy and valence will be used as labels since I wanted to investigate if there is also a relationship between the lyrics themselves and some of these features.

There is a small preprocessing step of this raw data, since some of the songs are named like "Song - Remastered 1990". When using the lyricsgenius library to acquire the song lyrics, song names including additional information than the title will hinder the performance trying to acquire lyrics for that song. In other words, such names like "Song - Remastered 1990" are shortened as "Song", since lyricsgenius only (and strictly) needs a name of the artist and the song. Luckily, the additional information in song names always begin with the character "-", thus each song name is processed such that if it includes "-", the remaining characters in the strings are removed. There are several songs themselves that include "-", but lyricsgenius was able to find it even if some songs' names are shortened as well in this process.

2.2 Scraping Lyrics

From the previous section, data is stored as a Pandas dataframe object. Since the names of the song and the artist is available, a function iterating through this dataframe scrapes the lyrics with the help of lyrcisgenius library (5) after feeding the song and artist names as the only input.

During the lyrics scraping process, several songs were still unable to be found, therefore a total number of 112 songs without lyrics were extracted from the dataframe. In addition, lyrics are cleaned from unwanted strings like "[Verse]" denoted in squared brackets and next line indicators shown as "\n". Furthermore, the decade in which each song has been released in has been extracted using the year feature, which will be used frequently in the visualizations chapter.

2.3 Labelling Data

The reformed data does not have any labels to be classified initially, therefore it has to be done either manually or some other way. As mentioned before, the original dataset acquired via Spotify's API included several features like liveness, danceability, energy, valence. Out of those features, I thought I can use valence and energy scores as they may also be reflected from the lyrics themselves. Each of these scores are between 0 and 1, in which 1 means "happy" and "up", while 0 means "sad" and "down" for valence and energy scores, respectively (3). In order to categorize these scores, a threshold value of 0.5 is used, where scores above 0.5 were labelled as "happy" and "up" in valence and energy scores respectively.

Since these valence and energy scores are originally captured as audio features, it may not be a healthy way to classify them, despite the possible relationship between these and the lyrics. VADER (6), a rule based sentiment analyzer by NLTK (Natural Language Toolkit),

is a commonly used library for sentiment analysis. Using VADER is very simple since the only input it requires is the raw text itself, without even any tokenization. Choosing VADER as a classifier would be shallow for the assignment, thus I used it as a third label to be tested by the classifier, which will be covered later. As an output, VADER returns 4 scores: negative, neutral, positive and compound scores. Using a threshold commonly applied for categorizing VADER scores (10), lyrics which has a compound score above 0.05 were labelled as "positive", the ones below -0.05 were labelled as "negative" and the remaining ones in between -0.05 and 0.05 were labelled as "neutral". The 51 neutral ones will be extracted from the data since the three mentioned labels so far will be binarized as 0 and 1 during classification.

Finally, I wanted to investigate if there could be a relationship between the lyrics and their decades as labels. The rock songs have release dates between 1950s to 2020, which means we could set 8 labels as decades (50s, 60s, etc). Having 8 labels in contrast to 2 from the labels before may share some similarities between emotion recognition since emotion recognition also deals with multiclass examples. The 4 different created labels to be examined are shown below. As the last step of this data collection section, lyrics and the labels to be used are saved as a csv file since the scraping implementation takes quite some time to acquire all the lyrics in the song examples.

- VADER labels categorized as "positive" and "negative"
- Valence score categorized as "happy" and "sad"
- Energy score categorized as "up" and "down"
- Decades as "50s", "60s", "70s", ...

Artist	Title	Year	Decade	Lyrics	Compound Score	Valence Score	Energy Score	Compound Label	Valence Label	Energy Label
Nirvana	Smells Like Teen Spirit	1991	90s	load up on guns, bring your friends it's fun...	-0.7224	0.720	0.912	negative	happy	up
Led Zeppelin	Stairway to Heaven	1971	70s	there's a lady who's sure all that glitte...	0.9771	0.197	0.340	positive	sad	down
Queen	Bohemian Rhapsody	1975	70s	is this the real life? is this just fantasy?...	-0.9840	0.228	0.402	negative	sad	down
John Lennon	Imagine	1971	70s	imagine there's no heaven it's easy if you t...	0.6775	0.169	0.257	positive	sad	down
The Rolling Stones	(I Can't Get No) Satisfaction	1965	60s	i can't get no satisfaction i can't get n...	0.9677	0.931	0.863	positive	happy	up

Figure 1: Created dataset before preprocessing

2.4 Kappa Distance

Although the original dataset comes premade, the majority of the final dataset including the lyrics and the labels were in fact created. Therefore, there is the need to measure the quality of the dataset as indicated in the course slides. In order to do so, I thought of measuring the kappa distance between two models, rather than comparing two lists of

manual labels. From the labels created, "decade" label is not up to comparison. Valence and compound labels are a bit difficult to find pre-trained models to compare with, so I decided to observe a kappa distance by applying a sentiment score function by Spacy library (2). With this method, there will be the chance to have two labels from two natural language processing libraries, thus the kappa distance to be measured will give a clearer idea since both essentially had scores for sentiments. Similar to the threshold applied to VADER labels, a threshold was applied to Spacy polarity scores where lyrics with a positive score were labelled "positive" and the rest was labelled as negative (Neutral was not taken into account since "neutral" songs were mostly uninformative and thus were extracted in labelling section. The measured kappa distance between VADER and Spacy labels is 0.32, which can be categorized as a "fair" but not strong bound (4). The implementation of the kappa distance is available in "IRTM VADER.ipynb" file.

3 Classification

Data prepared from the previous section has 4 labels to be predicted. Before diving into the model and the results according to each label, the lyrics have to be preprocessed in order to be fed into the deep learning based model. The code to the classification section can be observed under "IRTM VADER Labels", "IRTM Valence Labels", "IRTM Energy Labels" and "IRTM Decades".

3.1 Lemmatization

In order to create a vocabulary from the lyrics, the words of the lyrics need to be lemmatized. Using NLTK's WordNetLemmatizer (7), this is not challenging, since the library has a pre-trained function to reduce different variants or inflectional forms of a word into its base one. Since the lyrics are stored as string, there are several other processing steps done to the string to feed it into the word lemmatizer such lowering all the capital letters, removing stopwords and punctuation (9) again with the help of NLTK library.

3.2 Negation Handling

Negations are one of the most significant topics in sentiment analysis. Since the NLTK library cleans English stopwords during the lemmatization, an example string like "not good" is likely to be tokenized as "good", due to the fact that negations like "not" will be removed for being a part of the English stopwords set. This false tokenization is inevitable if not handled. Thus, during the lemmatization in the previous section, the stopword "not" is specifically extracted from the set of exclusion words. After that, if any lyric includes the characters "not ", it will be replaced with "_". By applying this method, during the vocabulary dictionary creation in "Classification" section will be able recognize negations like "not good" and tokenize it as "_good". Eventually, when the word tokens

will be indexed to be represented by numbers, examples like "good" and "_good" will be represented by different indices.

3.3 Tokenization

The lemmatized version of the lyrics are stored as a seperate column in the main dataframe. In order to feed these lyrics into a deep learning model, the strings need to be transformed into fixed arrays. The first step to do so is the tokenization of these lyrics, where each word is separated by whitespaces and stored as elements of a list. After this, a dictionary is created using the tokenized lyrics, where the keys represent the words and the values represent their frequencies according to all of the lyrics. This will also help with the later processes since the keys of the dictionary will be unique.

Using the vocabulary dictionary, the keys are enumerated and each word is given an index. Using these indices, the tokenized lyrics are processed where each song lyric consists of a list of word indices. However, each lyric as a learning example has a different length which will create a problem when feeding these into the model. Thus, a fixed size of 200 is chosen and lyrics with less words are padded with index 0 which was reserved for the padding instances. Similarly, lyrics with more than 200 words after tokenization are shortened by taking the first 200 words. In the meanwhile, the labels are binarized, except for the "decades" label which will be represented as a one hot vector (1). After all the preprocessing, data is finally suitable for feeding into a neural network model, since each lyric is represented by a 200 length array and each label is also categorized.

```
(5321, 200)

array([[    0,     0,     0, ...,  4082,  4082,  4082],
       [    0,     0,     0, ...,  3618, 12645,   356],
       [    0,     0,     0, ...,    26,   514,   780],
       ...,
       [    0,     0,     0, ...,   227,   227,   227],
       [    0,     0,     0, ...,   209,   473,   209],
       [    0,     0,     0, ...,  1359, 10765,  1147]])
```

Figure 2: The input to the model: 5321 lyrics are represented as arrays with a length of 200

3.4 Training, Validation and Test Separation

In order to implement a deep learning model, there is the need to separate data for training, validation and testing. Thus, the 80% of the vectorized lyrics will be used for training, while 10% will be used for validation and the remaining will be used for testing. Since Pytorch library will be used, the training, validation and test examples are transformed into Tensors and dataloaders for each one is created.

3.5 Model

For the type of model, Long short-term memory were used due to its suitability to process data with temporal dependency. They are commonly used for text classification, lyrics sentiment analysis can be considered as a branch of this. There are several layers of the model, where each will be explained in a separate chapter.

3.5.1 Embedding Layer

The embedding is the first layer of the model. After data preprocessing, lyrics were vectorized and are ready to be fed into the model. Following this, the embedding layer converts the input into embedding of a specific size, which will be 100 for our case.

3.5.2 LSTM Layer

LSTMs are popularly used in deep learning as recurrent models due to their success in handling vanishing gradient problem frequently observed in vanilla recurrent neural networks which hinders the model's ability to capture long term dependencies. Many examples of sentiment analysis with LSTMs have been implemented, especially using IMDB's annotated review dataset (8). The LSTM layer will take a batch of inputs of 100 dimensions, the output of the embedding layer. Since PyTorch's LSTM function is used for the example, the only parameters fed into the layer is the shape of the embedding output, and the number of neurons in the hidden layer which is taken as 256. It is also possible to create a bidirectional LSTM, which is a slight modification to vanilla LSTMs, in which hidden states are updated with regards to both forward and backward (inverted sequence) LSTMs.

3.5.3 Fully Connected Layer

After the output of the embedding layer is processed through the LSTM layer, the last hidden neuron is fed into a fully connected layer which outputs a scaler for the labels except the decades. For the case of "VADER Sentiments", "Energy", and "Valence" the output dimension will be 1 due to binarized labels. In contrast, for the "Decades" label, the output dimension will be 8 due to the labels being one-hot-vectors. At the end, a dropout with p=0.5 is applied after the fully connected layer, which will slightly improve the model's performance.

3.6 Results

During the training, as an optimizer, Stochastic Gradient Descent was selected. For the loss function, BCEWithLogitsLoss was used. After all the training, and the testing, below are the results for each label created. As a metric, besides the accuracy which is not generally a good indicator due to possible imbalances in the dataset, precision, recall and F1 values are shown.

```
print(f'The model has {count_parameters(model):,} trainable parameters')

model
```

```
The model has 10,352,041 trainable parameters

RNN(
  (embedding): Embedding(96186, 100)
  (rnn): LSTM(100, 256, batch_first=True, dropout=0.5, bidirectional=True)
  (fc): Linear(in_features=256, out_features=1, bias=True)
  (dropout): Dropout(p=0.5, inplace=False)
)
```

Figure 3: Number of trainable parameters and the overview of the model.

Model	Loss	Accuracy(%)	Precision(%)	Recall(%)	F1(%)
VADER Labels	0.682	59.14	85.17	59.15	69.61
Energy Labels	0.402	87.83	88.18	88.87	88.45
Valence Labels	0.689	55.51	0.00	0.00	0.00
Decades	0.473	25.59	77.21	25.59	37.27

Table 1: Performance of the 4 different labels under different metrics

The greatest performance were observed for the "Energy" labels, which were categorized versions of energy scores coming from the original dataset. Classification task for predicting the "VADER Sentiments" was the second in terms of the performance. For the remaining labels which are "valence" categories from the original dataset and the "decades", the classification task was not successful as it can be observed from the low metrics measured for the test data.

Achieving highest performance for energy labels can be an indicator that the audio features coming from the original dataset can also be linked to the lyrics themselves. For the cases of decades and valence, the poor performance is probably coming from two aspects: the fact that these labels were not informative with respect to the lyrics and the fact that the dataset created approximately 5000 examples which is a quite low number for general deep learning frameworks.

4 Visualizations

One of the aims of this assignment was to observe a potential relationship between the rock lyrics and the decade that they were written in. From the results of the classification task, this was unsuccessful in terms of the metrics observed during the testing phase. While this may also be a result of a wrong model selection, it seems that there is not a strong relationship between decades and lyrics, which means there have not been much change of the vocabulary observed in rock music.

The wordclouds for every decade labelled in figure 4. The implementation of this includes some groupby operations (9) of decades with respect to words, thus only the lyrics of 500 songs were taken for the visualizations. In addition, 50s and 20s decade labels have been removed since there were few songs fell into those categories.

Figure 4: Word Clouds according to each decade taken into consideration

Figure 5 shows the most frequent words of the "00s", and how frequently these words were used in other decades as well (9).

Further visualizations about the word choices and the decades are can also be observed. All the code related to this section is available in "IRTM Visualization.ipynb". Each decade's score assigned by VADER is indicated ion Figure 6. Since the label "20s" has very few examples, if it is disregarded, it seems that throughout the years, the song lyrics have lost some of their positive valence and gained more negative valence according to the VADER scores. Except the most negative decade "00s", there seems a linear relationship starting from the "50s", rock lyrics has became more negative.

Since the binarized "energy" labels gave the best performance with the classification task, I also wanted to depict how the songs changed in terms of this energy scored provided in the original dataset 7. During the classification task, the labels for this task was "up" and "down", but in the graph the original scores coming from the dataset was used to have a more expressive depiction.

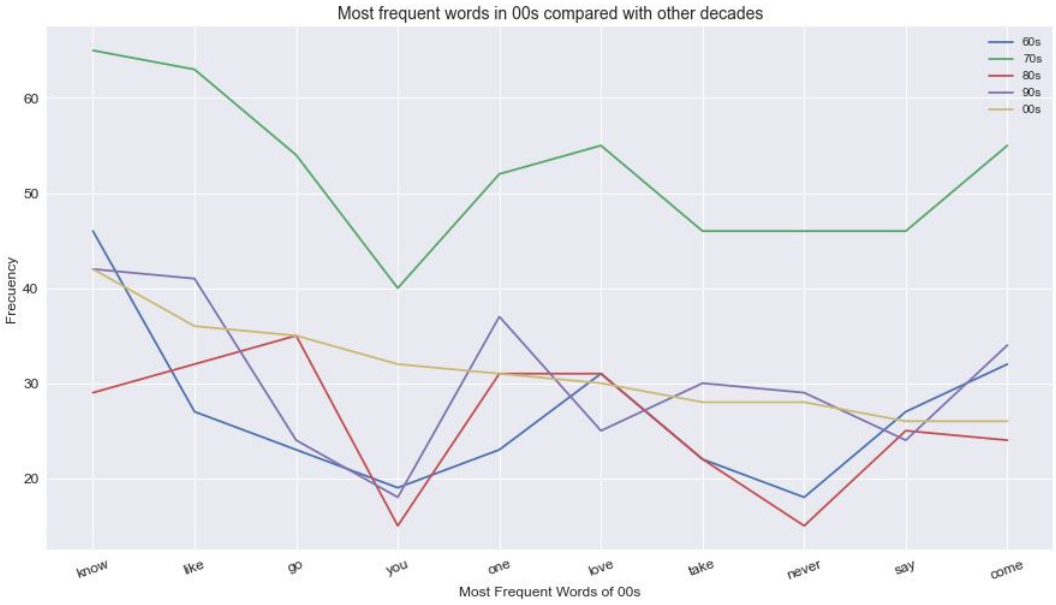

Figure 5: Most frequent words of "00s", compared to other decades

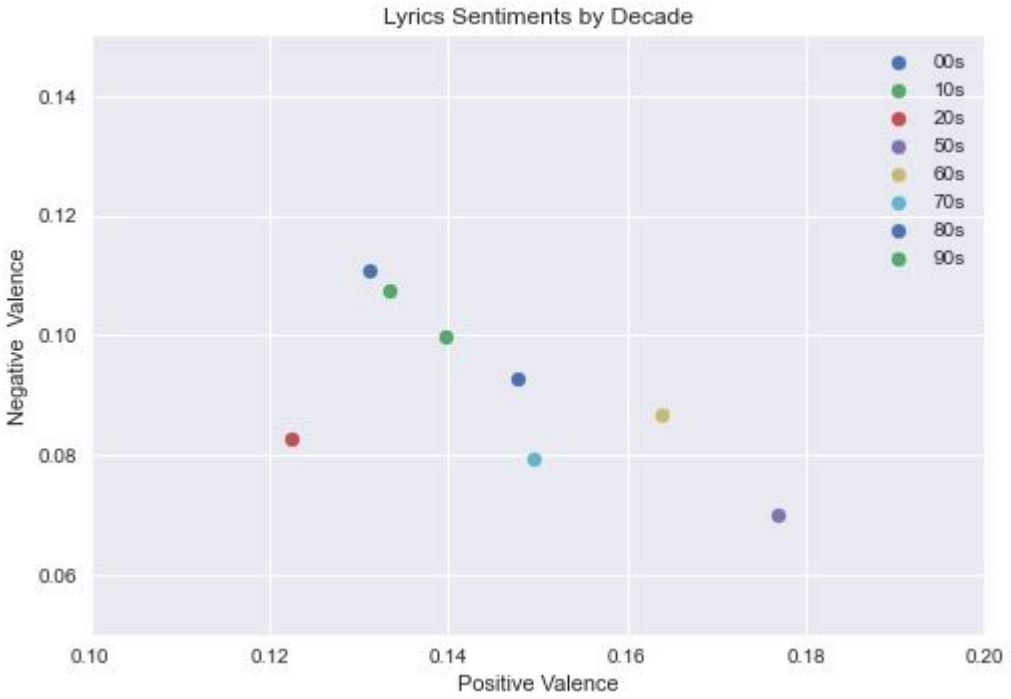

Figure 6: VADER Sentiment Scores according to their decades

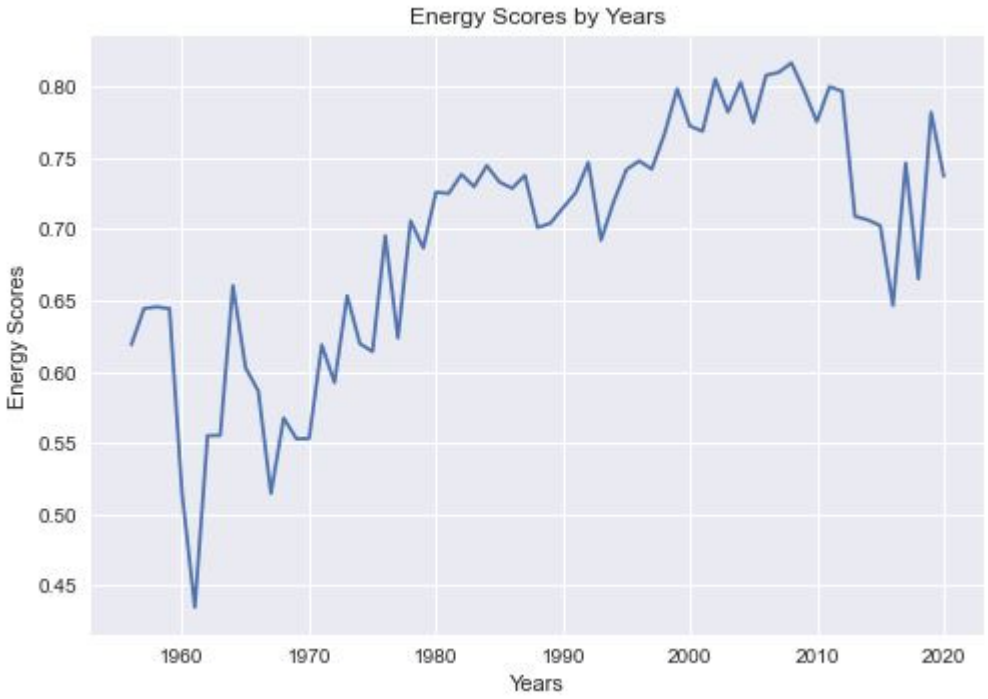

Figure 7: How the lyrics changed in terms of the spotify feature: "Energy Score"

5 Conclusion

This practical assignment deals with a sentiment classification task as its main objective, while incorporating other areas of information retrieval and text mining. Obtaining data of popular rock songs with several Spotify API's several features, the lyrics were scraped. In order to perform the classification, several different labels for different classification tasks were given. Necessary preprocessing steps like tokenization, lemmatization, negation handling, embedding and indexing the lyrics were applied. A deep learning model based on LSTMs were implemented in order to predict the formed labels. Out of the labels formed, I observed that the "energy" feature provided as a Spotify feature can also be derived from textual information coming from due to the high performance observed under different metrics. In the visualization section; wordclouds by decades and most frequent words of "00s" were compared with other ones. In addition several of the features coming from the dataset were displayed with respect to years in their raw continuous form.

References

[1] Samarth Agrawal. Sentiment analysis using lstm (step-by-step tutorial), 2019. URL: https://towardsdatascience.com/sentiment-analysis-using-lstm-step-by-step-50d074f09948.

[2] Sam Edwardes. spacytextblob, 2021. URL: https://spacy.io/universe/project/spacy-textblob.

[3] Spotify for developers. Web api reference, 2021. URL: https://developer.spotify.com/documentation/web-api/reference/#endpoint-get-audio-features.

[4] Mary L. McHugh. Interrater reliability: the kappa statistic, 2012. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3900052/.

[5] John W. Miller. Lyricsgenius, 2021. URL: https://pypi.org/project/lyricsgenius.

[6] NLTK Project. Source code for nltk.sentiment.vader, 2014. URL: https://www.nltk.org/_modules/nltk/sentiment/vader.html.

[7] NLTK Project. Source code for nltk.stem.wordnet, 2021. URL: https://www.nltk.org/_modules/nltk/stem/wordnet.html.

[8] Ben Trevett. Pytorch sentiment analysis, 2021. URL: https://github.com/bentrevett/pytorch-sentiment-analysis.

[9] Cristóbal Veas. How to analyze emotions and words of the lyrics from your favorite music artist, 2020. URL: https://towardsdatascience.com/how-to-analyze-emotions-and-words-of-the-lyrics-from-your-favorite-music-artist-

[10] Analytics Vidhya. Simplifying sentiment analysis using vader in python (on social media text), 2018. URL: https://www.nltk.org/_modules/nltk/sentiment/vader.html.

[11] Lukas Zamora. History of rock - spotify dataset, 2020. URL: https://www.kaggle.com/lukaszamora/history-of-rock-19502020.