EPiC Series in Computing
Volume 58, 2019, Pages 475–483
Proceedings of 34th International Conference on Computers and Their Applications

L,M&A: An Algorithm for Music Lyrics Mining and Sentiment Analysis

Vasu Saluja, Minni Jain and Prakarsh Yadav
Delhi Technological University, New Delhi, India
vasusaluja@outlook.com, minnijain91@gmail.com, prakarshy@gmail.com

Abstract

Here we propose an open source algorithm, L,M&A(Lyrics, Mine and Analyse) to create a dataset of lyrics of the works of various artists. The aim of this approach is to facilitate the generation of a large data set that can be used for improving accuracy of song recommendation algorithms. The limited availability of such datasets has excluded the sentiment analysis of lyrics from music recommendation systems. By using the L,M&A algorithm, it is possible to generate a large dataset which can function as training dataset for future classifier systems. We have used iterative API requests from musixmatch and Genius servers to text mine lyrics data of songs by multiple artists. The data is processed and then analysed for sentiment using lexicons provided in the Tidytext package (BING, AFINN, NRC) and the overall sentiment of artist was determined through modal counts. The occurrence of each sentiments was evaluated and visualized using ggplot2. This representation exhibits the merit of our approach and the applicability of our data. The key feature of our approach is the open source platforms utilized and simplicity of input required from user.

Keywords: Data Mining, Natural Language Processing, Sentiment Analysis, Lyrics Database, Music Recommendation, musicology, Rstudio, musixmatch, Genius

1 Introduction

Music analysis is central to all music recommendation algorithms. Efficient music recommendation algorithms make use of a song's popularity, user's past preference of music and clustering of different types of songs. From the above-mentioned methods, algorithms based on clustering of songs are the most complex. It involves analysis of the elements of song such as lyrics, beats, tempo and a host of other elements. A challenging task for the development of such an algorithm is the presence of accurate classifiers of music sentiment. Considering and analyzing the sentiment behind a music can greatly improve the accuracy of song recommendation algorithms.

The developing music recommendation systems based on lyrics is greatly stymied by the limited availability of open source lyrics datasets. A thorough and comprehensive dataset is necessary for training classifiers that can accurately ascribe sentiment to a song on the basis of its lyrics. Here we propose an algorithm to mine lyric data for various artists which can be further utilized to generate a sentiment for individual artists.

Classification of artists based on the analysis of lyrical content of their work offers a unique dimension to improve the accuracy of existing song recommendation algorithms. Many music enthusiasts are interested in the lyrics of the songs. At the moment, widely popular genres of music, such as rap, are heavily based on lyrics. Thus, recommendation algorithms that over-look this aspect are often missing out on the accuracy of recommendation such an approach can offer. The employed model is an independent function that requires an artist's name as primary input and through open-source APIs[*][†] (Application Programming Interface) gathers the required data regarding that artist's work and then employs our algorithm, L,M&A(Lyrics, Mine and Analyse) to generate a sentiment (based on the different categories of sentiment present in the lexicons used) for that particular artist and is also able to quantitate that sentiment. A wholistic dataset would form the premise of such recommendation systems.

Furthermore, such a dataset can be used to improve the existing sentiment-based lexicons. The sentiment lexicons' scoring is not specific towards lyrics and developing such specific lexicons requires an expansive dataset. This further underscores the need for the availability of music lyrics specific dataset. Herein, our L,M&A algorithm would be the catalyst for generating such datasets. Any user can simply supply a list of artists and the algorithm is able to create a dataset with all the albums of that artist, and subsequently fetch lyrics of all the songs in these albums.

This paper is organized as follows: Section 2 provides the work that has been done to address the lack of musical dataset and its subsequent sentiment analysis. Section 3 provides our proposed work, i.e. how we developed the dataset of lyrics of songs from various artists. Section 4 discusses the salient features of the output of our L,M&A algorithm and a sample of the sentiment analysis done on this dataset. Section 5 provides a glimpse of the future applications of such a dataset, such as music recommendation systems, improving sentiment lexicon accuracy and musicology.

2 Related Work

This sentiment analysis algorithm draws on multiple sources of knowledge [1]–[3]. We greatly rely on the basics of text classification, in form of open-source lexicons. Since the sentiment classifier is based on the text classification method it was necessary to incorporate concepts of negation and advanced pre-processing to ensure that only relevant components of lyrics were extracted and further analysed. It was found that all the works that were referred to had only attempted to ascribe sentiment to individual songs.[4] No attempts were made to correlate the sentiments to artist in regards to the entire discography, i.e. assign a generalized sentiment to an artist rather that individual songs.[5]

The 55,000 songs dataset and million song dataset [6]are open source but they carry certain limitations. Such as, in the million song dataset lyrics are present as a bag of words and lack the structure found in a song. A song is not just collection of words there is an order and structure to how each word is used. Similarly, the 55,000 songs dataset[‡] does not contains the whole work of an artist and is susceptible to noise in the lyrics, in terms of unnecessary words, single characters etc. Apart from these, it needs several pre-processing steps before any analysis can be performed such as removal of contractions, single character & unnecessary words. We also apply these pre-processing steps but in our case the exceptions are quite few and follow a definite pattern, unlike in 55k where web scraping techniques gather several different texts. On the other hand, our method can be used to retrieve lyrics of works of single or multiple artist and at the same time can be scaled to mine data for hundreds of thousands of songs.

[*] https://developer.musixmatch.com
[†] https://docs.genius.com
[‡] https://www.kaggle.com/mousehead/songlyrics#songdata.csv

A thorough literature review had made it apparent that the subject of sentiment analysis through lyrics has previously been done, but in some sense was found to be insufficient to address this complex issue. There are studies where the subject is sentiment determination through analysis of lyrics, but they were found to be limited in the dataset used [7], degree to which the sentiments ascribed were different, for example happy, good, sad, etc.

In the work by Gomez et. al., [1] the methodology for acquisition of data, through data mining, were adequately highlighted. However, after referring it was apparent that the methodology proposed had been applied on a prohibitively small data set of 593 songs. Further works also had a similar short coming in the scope of the data set used to test the model or the algorithm proposed in the paper. Elaborate mathematical models had been proposed, [8] but were not adequately tested to a wholistic degree.

This led to the motivation of our work wherein we employ a direct method of text classification and subsequent sentiment assignment. We attempted to use all the major available open-source lexicons and, compare and contrasted all such lexicons in their ability to accurately determine the sentiment associated to a particular artist. After testing on various data sets of songs it was made apparent that this methodology is highly accurate in ascribing a generalized sentiment to a particular artist. The sentiment ascribed can also be quantitated and incorporated in other functions. This work can also be conveniently applied to other languages simply by changing the lexicon being employed.

3 Proposed Work

Before starting with the actual L,M&A algorithm, we first had to register for the musixmatch & Genius Developer account, from where using the instruction, we acquired the root URL of musixmatch API and Genius API {base_key & genius_baseurl} (API- Used to interact with server database). Along with these, authentication key, for verification was generated on the developer page of both {api_key, access_token}. After all the architecture setup, the necessary packages (jsonlite[§], plyr[**], GeniusR[††], Tidyverse[‡‡], stringdist[§§]) along with the artist dataset is loaded into the R workspace11. Afterwards artist's name was iteratively passed, wherein in each iteration a series of tasks were performed. Represented in Algorithm 1.

At First, we created a request URL to search for xi in the musixmatch database by joining the root URL with the required ARTIST.SEARCH API method of musixmatch (xi as argument) and the api_key {artist_search}. Using jsonlite methods (artist_search as argument) to convert the json response received (json is a data readable format, which is used by several API to parse a request) after connection to the API into a R's data object. Once the response was verified ([art$status_code == 200] – HTTP Code indicating that request was successful) and checked ([art&body !empty] – that the received json response body is not empty), Dataframe was extracted from it wherein the artist name was then tested based on string similarity{stringism()}, if the similarity was above 85% then the ID of the artist was extracted from the dataframe.

Then using the ID another request was created to get the discography of the artist, ARTIST.ALBUMS method {artist_album}. The album dataset was extracted from the JSON response of the request, parsed, checked and verified in the same manner as mentioned above{artist_album}.

[§] https://arxiv.org/abs/1403.2805
[**] http://www.jstatsoft.org/v40/i01
[††] https://github.com/JosiahParry/geniusR
[‡‡] https://www.tidyverse.org/packages/
[§§] https://cran.r-project.org/web/packages/stringdist/index.html

```
if X is a set of all artist name
x_i is the i^th artist name
base_key, api_key, genius_baseurl, access_token
for x_i in X:
        artist_search <- base_key + ARTIST.SEARCH(x_i)+ api_key
        art <- fromJSON(artist_search)
        if (art$status_code == 200 & art$body !empty)
                if (Stringism(x_i ,art$body$artistdata) > 0.85)
                        id <- art$body$artistdata$id
                artist_album <- base_key + ARTIST.ALBUMS(id) + apikey
                  albums <- fromJSON(artist_album)
                if (albums$status_code ==200 & albums$body !empty)
                  album<- select (albums$id, albums$name, albums$trackcount, albums$type)
                  album <- filter (album$type in (Album, EP), album&trackcount >5)
                  data <- dataframe(track_title, lyrics, artist_name)
                  genius_artist <- genius_baseurl + GET_SEARCH (x_i )+ access_token
                  name <- fromJSON(genius_artist)
                  if (name$status_code == 200 & name$body !empty)
                     if (stringsism(name$primary_name, x_i ))
                        name <- filter(name$primary_name_url)
                  for i in album:
                          lyrics <- genius_album(name , i)
                          data <- bind (data, lyrics)
                  data <- summarise(data)
data <- bind_row(data, data)
write_csv(data)

remove contractions & singlechar & lower_casedata
for i in unique(data$artist_name):
        filter (data for same artist name)
        tokens <- unnest_token(data$lyrics)              #tokenization
        antijoin(tokens, stop_words, unnecessary_word)
        bing <- innerjoin(tokens, bing_sentiment)
        nrc <- innerjoin(tokens, nrc_sentiment)
        afinn <- innerjoin(tokens, afinn_sentiment)
        bing_count <- top_n_words(bing)
        nrc_count <- top_n_words(nrc)
        afinn_count <- top_n_words(afinn)

        unnest_token(data$lyric, ngram = 2)     #bigram Tokenization
        words_not <- filter (word1 == "not")
        pair_count <- pairwise_count(tokens)
        pair_corr <- pairwise_correlation(tokens)
        Plot (words_not, pair_count, pair_corr, bing_count, afinn_count, nrc_count, bing, nrc, afinn)
```

Algorithm 1: Lyrics, Mine and Analyse Algorithm (L,M&A)

The data was filtered for album name, ID, track count & release type having type Album or EP and count greater than 5{album}. Reason being an album contains more than 4-5 songs in it. Also, many were listed as albums but were actually compilation of some songs of album. And in case of EP only a few songs are released. Live and singles were neglected, as repetition.

Finally, a query was formed for the genius API to extract and match the name of the artist before providing the information to succeeding function to minimize error. Using genius_baseurl along with GET_SEARCH method of Genius API and access_token, a query was formed{genius_artist}. Data was retrieved (see above) from response and relevant data (artist name present in the variable containing primary URL of an artist, defined on genius server was extracted) based on string similarity between xi and data was filtered accordingly{name}.

The artist name{name} along with all the album names of that artist {album} were used to extract the lyrics of all the songs in a particular album. This was done using GeniusR package genius_album method, where artist name, and album names were passed iteratively to capture the lyrics of all songs of the artist{lyrics}. A dataframe with all the songs in an album were received. Wherein each line of text of a song was as a separate observation in it. All the lines of a song were concatenate into one observation{data}. This was done for all the albums by an artist sequentially, i.e. all the songs.

The whole process was repeated for all the artists in the artist.csv. Finally, all the songs of all the artist were merged to create the final dataset. Represented diagrammatically in Figure 1 A.

After the creation of the data, it was exported as a comma separated file, containing 23,138 observations and 3 variables (track_title, lyric, artist_name). This dataset was then used for sentiment analysis. This was done by first rectifying abbreviated words then removing single character/ digits with no apparent meaning {remove (contraction & singlechar)}, finally converting the lyrics to lower case {lower_casedata}.

The data was iterated artist wise after initial pre-processing. Tokenization [Breaking up a set of texts into individual words] (unigram) of lyrics{unnest_token}, followed by removal of undesirable words (such as "ah", "uh", "yo" & many other), stop words and words less than 3 characters{antijoin}.

Once the tokenized dataset was available, Word-clouds are generated to visualize and infer the most used words by an artist (wordcloud2). Sentiment datasets were created for the lexicons (a form of vocabulary, in this case word as key and sentiment associated with that word as value) used, namely AFINN, BING and NRC, provided in Tidytext[***] package, via an inner join (separating out words that are present in both the sentiment dataset and the tokenized, filtered dataset) and then visualized (sentiment vs count). Frequency of words corresponding to different sentiment categories was then evaluated. {bing_count, nrc_count, afinn_count}.

After all the visualization was done for the case of unigrams, tokenization was done for the case of bigrams (set of two consecutive words in test) {ngram = 2}. Polarity of words which were preceded by 'not', was checked and plotted {words_not} Afterwards Pair-wise count and Pair-wise correlation (using Widyr[†††] package) of most frequent words with other (pairwise_corr(), pairwise_count()) words were carried out and visualized{ggplot()[‡‡‡]for visualization, Rstudio12 [§§§]as IDE}. Represented diagrammatically in Figure 1 B.

4 Results & Discussion

Through the implementation of the above proposed L,M&A algorithm a dataset of 23138 songs from the set of 265 artists. API requests for various songs of each of the 265 artists were sent to the musixmatch server. A sample of the dataset is shown in Table1. The response was verified and further details of artists' album name was extracted. The names were then selected, filtered and fed into a dataframe that functions as the basic input of GeniusR package. This 'Album Dataframe' is called by the GeniusR package to extract lyrics of the songs in that album.

[***] https://cran.r-project.org/web/packages/tidytext/index.html
[†††] https://cran.r-project.org/web/packages/widyr/index.html
[‡‡‡] http://ggplot2.org
[§§§] http://www.rstudio.com/

Since GeniusR package is highly susceptible for error in input, the Dataframe of artists' name and album names had to be curated appropriately. This was achieved by implementation of multiple "exception handling" conditions.

The main error surrounding this was of the way an artist name was supplied to the function, if there was any variability between the name that we supplied and the artist name stored on Genius API, we got an error. The same was true for the musixmatch API. So in order to resolve this we took string similarity instead of equality between the strings to retrieve artist's name from respective APIs and then use those for further processing, we took a threshold of above 75-85% similarity to proceed further. Another main error that we came across was the case of the albums list of an artist, since we had to supply that as an argument to retrieve lyrics of all the songs in the album, we had to build exception handling (using try-catch blocks) for the cases the album name supplied was not present in the Genius server. We also had to take exception for cases that an artist with already built in platform, but possessed no data on the server, had to be removed accordingly.

This generates a dummy dataset that is used as a feedback into the Album List. Finally, once the algorithm has been completely executed, the dummy dataset is then processed to generate a single array with artists' name, album name and songs' lyrics.

The advantage of our proposed L,M&A algorithm for generation of a dataset is that it requires minimal input form user, in form of artist name, and can provide all the lyrical data that is in that artist's discography. Additionally, we have used a method of string similarity in our API requests, with cut-off of 85% in the name of an artist, this ensures that even if there is variance in spelling or special characters are used the code is still able to accurately determine the artist. This functionality is used twice, once for API requests through the musixmatch server and second time for the GeniusR package. This functionality was necessitated by the difference in musixmatch and GeniusR's method of indexing artist name. The string similarity method ensures that these variances do not lead to a mismatch or return of null value at any point in the algorithm.

The L,M&A algorithm is highly robust and requires an input of .CSV format containing the names of artists. Using such minimal input, it is able to generate highly relevant and accurate results. As an application of the dataset, we further developed an algorithm to perform sentiment analysis on it. We employed various sentiment-based lexicons to quantitate the sentiment for each of the artist. The output data was provided as graphs. The plots shown in this paper, Figure 2, are sufficient for representing the intent of our work. However, the magnitude of data processed is beyond the scope of graphical representation.

The supplied data clearly exhibits the merit in our approach for lyrical analysis with far reaching applications. The most obvious application would be in song recommendation algorithms. The execution of algorithm provided AFFIN count, NRC count and BING count for each of the artist from the collated csv. These plots and generated values formed the basis for further clustering of artist to allow generation of a primitive recommendation system. Considering the case of Beatles, we observed that NRC sentiment of Beatles scored high on positive aspects such as joy and trust. Similarly, in BING count there was a markedly higher frequency of positive words such as love, sweet, free and loving, in comparison to lonely, shame and cry. Finally, in case of AFFIN count the songs scored very high in positive scoring elements like fun, love, happy, sweet, care, etc. Figure 2 On the contrary, Black Sabbath, a band greatly influenced by the era, scored high on elements like fear, sadness and anticipation in NRC count. AFFIN and BING count showed similar results as the band scored high on negative sentiment words like death, cry, pain, etc.

Through our code it is possible to generate a similar data set for any artist. The dataset of lyrics is unique as unlike other data sets that are publicly available, our data set requires very minimal, if any, preprocessing. The structure of lyrics of each song is conserved and can be easily utilized for various forms of analysis. For instance, the sociological analysis or improve accuracy of lexicon or as a basis of music recommendation systems.

Lyrics convey the essence of emotion in a song and can greatly influence the listener's reception of the song. Thus, development of such a classification system is one of the first steps towards improving the existing song recommendation systems. Further pursuit of this classification system is required to make it highly robust and exploit the intricacies of the vast information provided by lyrics in a song.

5 Conclusion & Future Work

Through the analysis of results presented above it can be concluded that lyrics play a crucial role in the general sentiment of musical works. The quantitated analysis of lyrics in such an algorithmic way allows for betterment of existing music recommendation systems. Many musical connoisseurs greatly appreciate the intricacies of lyrics and such an algorithm directly caters to such needs. The quantitated score obtained from the proposed L,M&A algorithm can be coupled with a weighted function to be seamlessly incorporated into other music comparators, such as tone/tempo analysis.

The limitation to our L,M&A algorithm is centered around the fact that we had to employ quantitated sentiment lexicons. The accuracy of our interpretations and results are directly correlated to the accuracy of used lexicons. Hence, we propose to employ machine learning and fuzzy logic approach to further improve the quantitative accuracy of these lexicons in regards of lyrical words. The training of this algorithm would be carried out on the mined lyrics data. This would curate the lexicons to preform markedly better on atypical data sets, such as music lyrics.

Another application of our data is in the field of music sociological analysis or musicology. Music is greatly representative of the society's mood in that time period. Drastic events like war and communal violence inspire artists to create music that reflects the sentiment of society at that point of time. The quantitated approach offered by our L,M&A algorithm offers a promising new dimension of sociological analysis. The representation of artists' discography in form of graphs and charts allows convenient and accurate assessment of sociological implications of their work.

References

[1] L. M. Gómez and M. N. Cáceres, "Applying Data Mining for Sentiment Analysis in Music," in *Trends in Cyber-Physical Multi-Agent Systems. The PAAMS Collection - 15th International Conference, PAAMS,2017*

[2] S. Howard, C. N. Silla, and C. G. Johnson, "Automatic Lyrics-based Music Genre Classification in a Multilingual Setting," In: *Thirteenth Brazilian Symposium on Computer Music,* 2011.

[3] T. Mullen and N. Collier, "Sentiment analysis using support vector machines with diverse information sources," In *Proceedings of Conference on Empirical Methods in Natural Language Processing,* 2004.

[4] A. Jamdar, J. Abraham, K. Khanna, and R. Dubey, "Emotion Analysis of Songs Based on Lyrical and Audio Features," in *International journal of Artificial Intelligence & Applications,* 2015.

[5] S. Shukla, P. Khanna, and K. K. Agrawal, "Review on sentiment analysis on music," in *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*, 2017.

[6] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The Million Song Dataset," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[7] V. Sharma, A. Agarwal, R Dhir, G. Sikka, "Sentiments mining and classification of music lyrics using SentiWordNet" in *IEEE Xplore*, 2018.

[8] Y. Hu, X. Chen, and D. Yang, "Lyric-based Song Emotion Detection with Affective Lexicon and Fuzzy Clustering Method," in *Proceedings of ISMIR 2009*

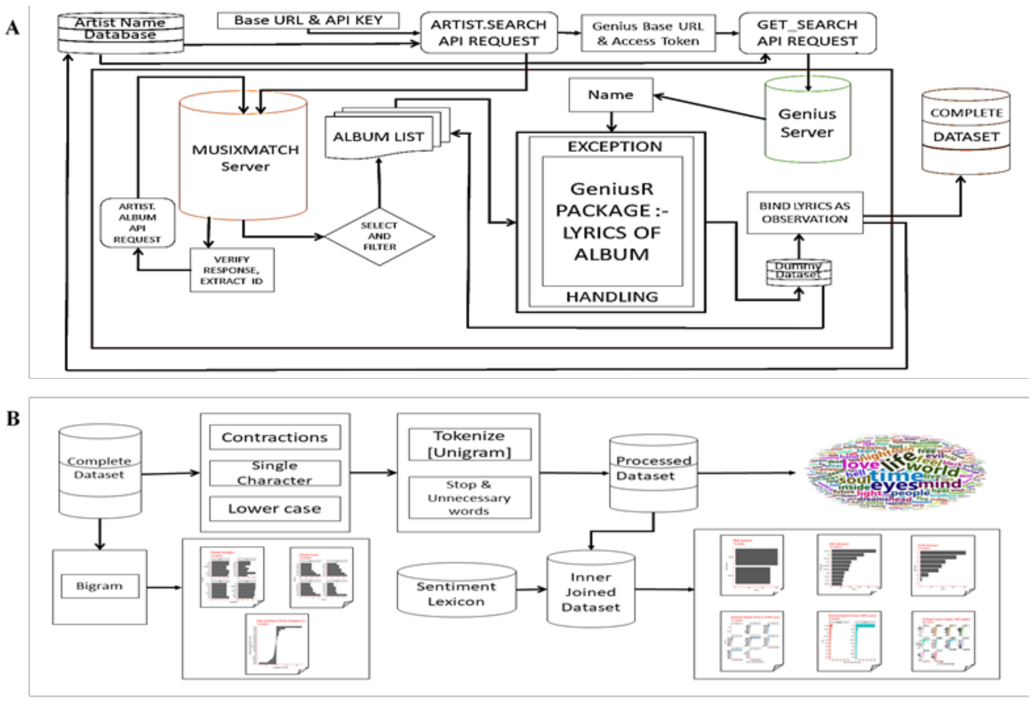

Figure 1: Flow diagram representing the dataset. 1 A represents the data mining aspect of the data (Deep red box represents global loop and deep blue representing local loop), 1 B represents the resultant sentiment analysis

Table 1 : Sample representation of dataset generated

track_title	lyric	artist_name
Baby's in Black	Oh dear, what can I do? Baby's in black and I'm feeling blue	The Beatles
Hey Jude	Hey Jude, don't make it bad Take a sad song and make it	The Beatles
God Is Dead?	Lost in the darkness I fade from the light Faith of my father,	Black Sabbath
Paranoid	Finished with my woman, 'cause she couldn't help me With	Black Sabbath

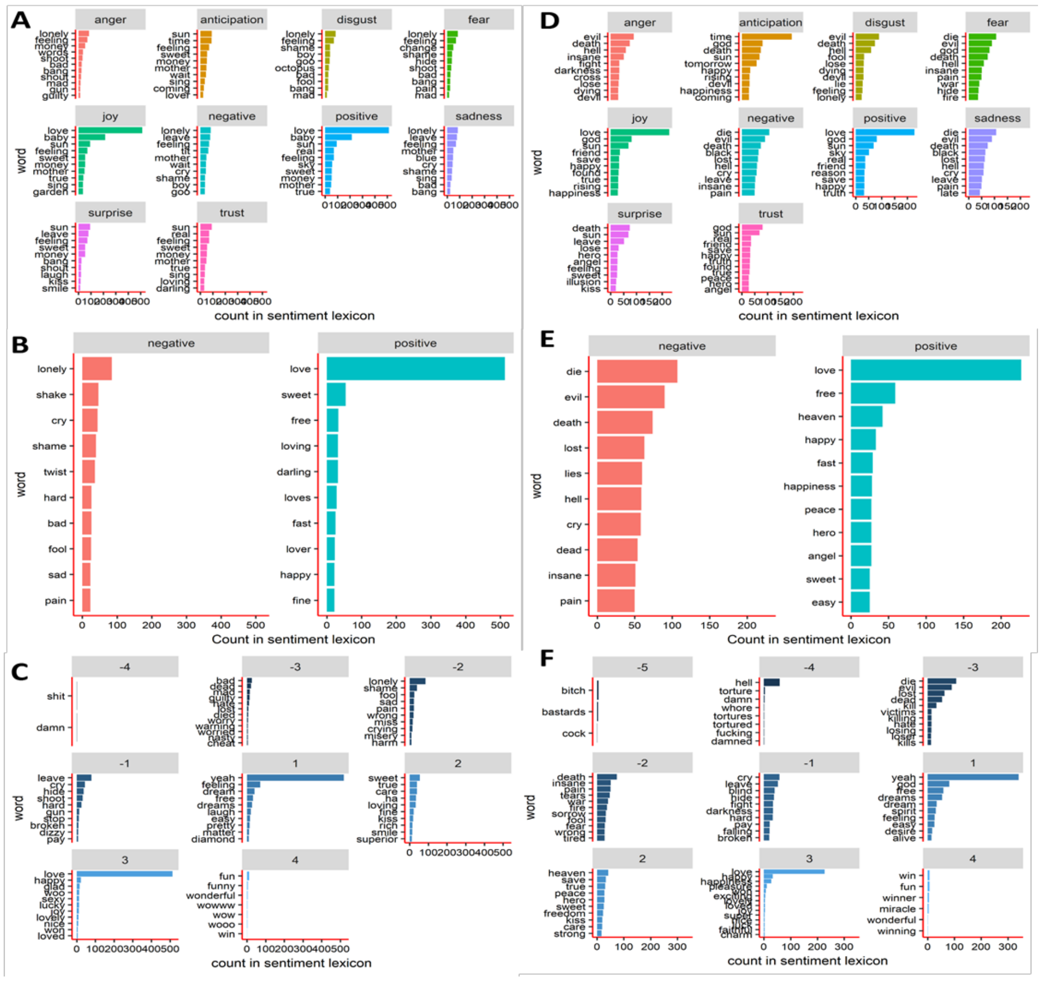

Figure 2: Panel A-C represent the exploratory plots of band 'The Beatles' and panel D-F represent same plots for the band 'Black Sabbath'. Plot A and D represent most used words in different sentiment categories within NRC lexicon, similarly plot B, E, C & F represent for BING and AFINN lexicons.