MIT Academy of Engineering

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

TY MINI PROJECT REPORT

ON

Lyrics Analysis

BY

120160458 Upasana Mehta
120160457 Tripti Mehta
120160425 Prajakta Lasanpure

Project Advisor

Mrs.Vaishali Wangikar

SCHOOL OF COMPUTER ENGINEERING & TECHNOLOGY
MIT ACADEMY OF ENGINEERING
ALANDI (D), PUNE
2019 – 2020

CERTIFICATE

This is to certify that the Mini Project entitled “ **Lyrics Analysis**“ has been carried out by Upasana Mehta (SCETTY239), Tripti Mehta (SCETTY238), Prajakta Lasanpure (SCETTY 620), in partial fulfillment of Third Year Computer Engineering as well as in the record of Mini-project work done by him/her at SCET,MIT AOE- an Autonomous institute affiliated to Savitribai Phule University, Pune under the guidance of “ Mrs. Vaishali Wangikar ” during the academic year 2019-2020.

Sign:

Name: Mrs. Vaishali Wangekar

Date: Prof. Ranjana Badre

Project Guide /Advisor **Sign of School Dean (SCET)**

INDEX

TABLE OF CONTENT

CHAPTER 1
INTRODUCTION

1.1. PROBLEM DEFINITION/ OBJECTIVE:

Utilizing different machine learning algorithms we will perform Text mining and exploratory analysis using a dataset of hundreds of song lyrics by the legendary artist, Prince. We will focus on sentiment analysis and topic modeling with Natural Language Processing (NLP) and clustering techniques such as Latent Dirichlet Allocation (LDA) and K-Means to tease out motifs in the lyrics.

1.2. MARKET SURVEY:

Our project aims to do the sentiment analysis of singers work and life. To perform this analysis we determine what decade a song was released, or more interestingly, predict whether a song will hit the Billboard charts based on its lyrics , also Dive into the lyrics of Prince's music throughout the year. After exploring the dataset we need to know the total number of songs sung by him , how are the lyrics structured , How much cleaning and wrangling needs to be done, What are the facts, What are the word frequencies and why is that important.

1.3. SOFTWARE AND HARDWARE REQUIREMENT SPECIFICATION

Software - We will be using R programming language and its free open source software IDE, RStudio version 1.1.463. R is a tool for statistical computing and graphics. R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering,) and graphical techniques, and is highly extensible. RStudio includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS.

We will also be using MS excel where we will be storing the data on which we will be doing the analysis. Microsoft Excel is a spreadsheet developed by Microsoft for Windows, macOS, Android and iOS. It features calculation, graphing tools, pivot tables, and a macro programming language called Visual Basic for Applications.

1.4. TARGET SPECIFICATION

Comparing real life events, both personal and societal, can illuminate the mood of any lyric. This analysis can be performed on the lyrics of any singer based on his past records and present songs written. It depends on a wide range of criteria such as the amount of data preparation, the choice of lexicon, the method of analysis, the quality of the source data, and so on.

CHAPTER 2
LITERATURE REVIEW

Musical lyrics may represent an artist's perspective, but popular songs reveal what society wants to hear. Lyric analysis isn't an easy task. It is often structured in different patterns and formats. It requires caution with assumptions and a uniquely discriminant choice of analytic techniques. Musical lyrics permeate our lives and influence our thoughts with subtle ubiquity. The concept of Predictive Lyrics is an emerging and prevalent as a subject these days. Via this project we will cover a few topics of this emerging subject which will give us an insight as to what Predictive Lyrics is.

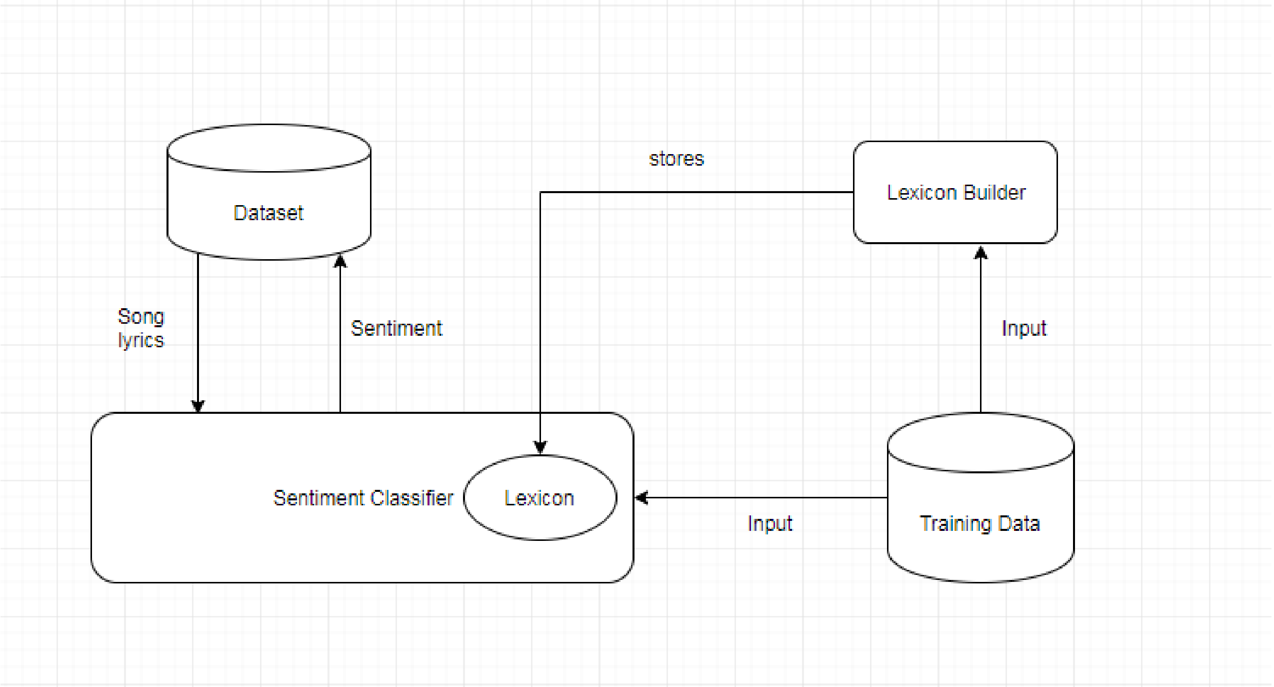
Here, we will be analyzing the song lyrics of an American singer, songwriter, musician, record producer, actor, and filmmaker, Prince Rogers Nelson. Prince's music had a great influence on the development of many genres globally.

In this project we will be utilizing text mining techniques on a set of lyrics using the tidy text framework. Tidy datasets have a specific structure in which each variable is a column, each observation is a row, and each type of observational unit is a table. After cleaning and conditioning the dataset, we will create descriptive statistics and exploratory visualizations while looking at different aspects of Prince's lyrics. We will explore and analyze Prince's career, his lyrical analysis and personality.

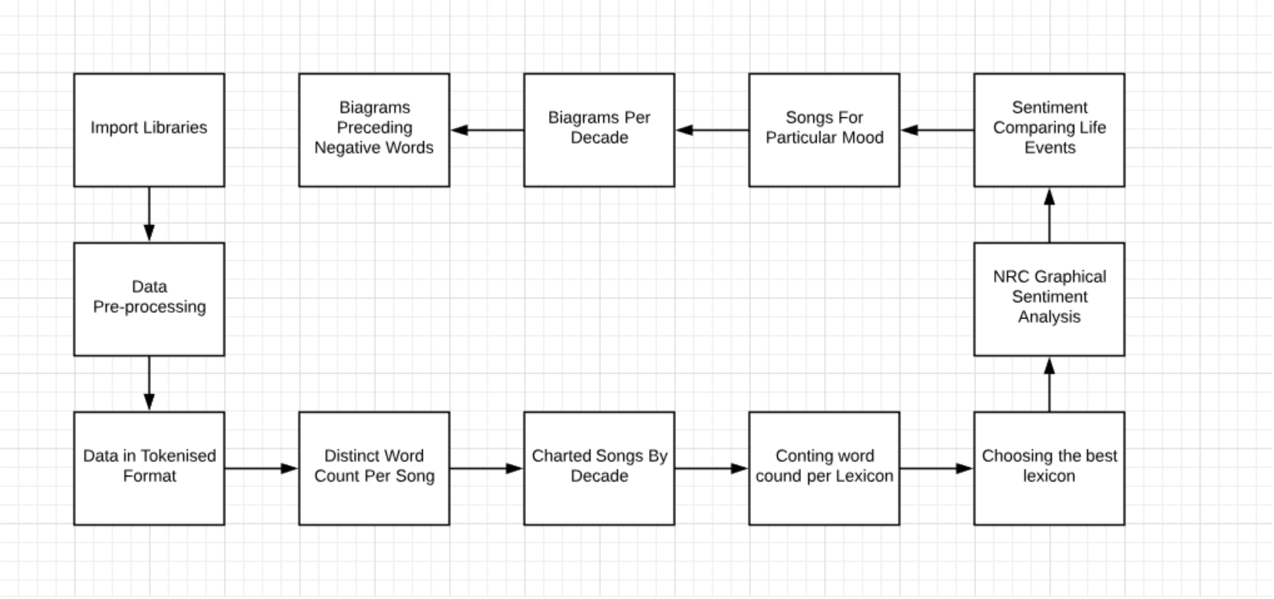
We will use a variety of packages provided by R to clean the data and perform operations to extract meaningful insights from the data. We will plot graphs for visual analysis. These graphs will help us conclude a lot of results.

CHAPTER 3
SYSTEM ARCHITECTURE

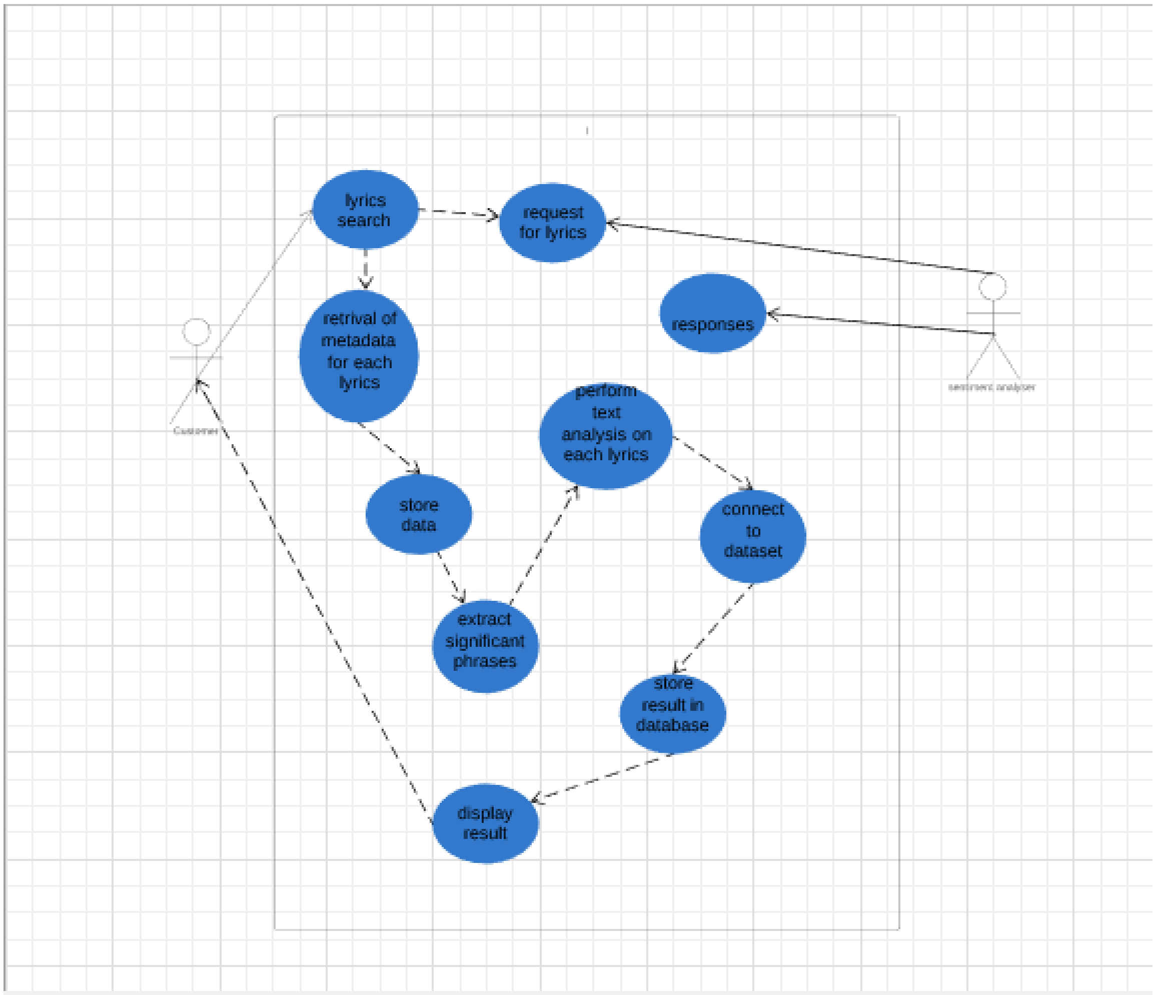
3.1. SYSTEM ARCHITECTURE



3.2. DATA FLOW DIAGRAMS


Import Libraries
Biagrams Preceding Negative Words
Biagrams Per Decade
Songs For Particular Mood
Sentiment Comparing Life Events
Data Pre-processing
NRC Graphical Sentiment Analysis
Data in Tokenised Format
Distinct Word Count Per Song
Charted Songs By Decade
Conting word cound per Lexicon
Choosing the best lexicon

3.3. UML DIAGRAM



CHAPTER 4
SIMULATION/SOFTWARE DEVELOPMENT FABRICATION OF PROJECT

4.1 SIMULATIONS:

- **DATA PRE-PROCESSING**

  Data pre-processing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data pre-processing is a proven method of resolving such issues. Tasks in data pre-processing :

  1. Data cleaning: fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies.
  2. Data integration: using multiple databases, data cubes, or files.
  3. Data transformation: normalization and aggregation.
  4. Data reduction: reducing the volume but producing the same or similar analytical results.
  5. Data discretization: part of data reduction, replacing numerical attributes with nominal ones.

- **LEXICAL DIVERSITY PER DECADE**

  Here is an interesting view of the clean and tidy data showing the lexical *diversity*, or, in other words, vocabulary, of the lyrics over time. An advanced method of plotting a continuous dependent variable, such as the word count, as a function of a categorical independent variable, like decade. This combines raw data points, descriptive and inferential statistics into a single effective plot.

  Every colored circle in this pirate plot represents a song. The dense red area with the "NONE" value shows that a large number of songs in the dataset do not have a release date. There is a slight upward trend in the unique number of words per song in the early decades: the solid horizontal line shows the mean word count for that decade. This is important to know when you begin to analyze the sentiment over time. The words become more illuminating throughout Prince's career.

- **SONG COUNT PER YEAR AND RELATIONSHIP BETWEEN CHART AND DECADE:**

The graph below is simply a circular bar chart using coord_polar() from ggplot2 that shows the relative number of songs per year. The gap at the very top is an indicator of the hundreds of songs without release. With such a large number of unreleased songs, the chart would not be useful if they were included. The missing years indicate those where no songs were released. The most prolific years were 1996 and 1998.

The relationship between the decade a song was released and whether or not it hit the Billboard charts was done using a chordDiagram().
The graph is split into two categories: charted (top), and decade (bottom). The two categories are separated by wide gaps, with smaller gaps between the values.
It nicely illustrates the counts of songs per decade, per chart level. You can see that Prince began his career in the 1970s with only a few releases, some of which charted. If you compare the 1980s to the 1990s, you'll find that more songs were released in the 1990s, but more songs charted in the 1980s. There were only a few commercially successful songs in the 2000s and in the 2010s there were no hit songs.

- **AFINN NRC AND BING**

The tidytext package includes a dataset called sentiments which provides several distinct lexicons. These lexicons are dictionaries of words with an assigned sentiment category or value. tidytext provides three general purpose lexicons:

AFINN: assigns words with a score that runs between -5 and 5, with negative scores indicating negative sentiment and positive scores indicating positive sentiment

Bing: assigns words into positive and negative categories

NRC: assigns words into one or more of the following ten categories: positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust

- **POLAR MELTING**

Polarity refers to creating an positive and negative field.For creating a polarity graph we add a yintercept with geom_hline().
Since you're looking at sentiment from a polar perspective, you might want to see weather or not it changes over time (geek humor).
ThUS we use geom_smooth() with the loess method for a smoother curve and another geom_smooth() with method = lm for a linear smooth curve.
we have done analysis for polarity over time and percent positive over time. From both these graphs we have observed polarity trend over time is negative in both cases.

- **SENTIMENT BY EVENTS**

we first create create a list of Prince's life events collected from various sources. We create a princeEvents.csv for all the list of his events.
To seperate the words into categories of positive and negative words we create a prince_bing.To create a polarity score per year of his events we use spread().
using the same dataset we count event years with no releases and keep an account of the bad data.

- **BIGRAMS**

Examining pairs of two consecutive words is called as bigram. Bigram on the year 1998, in 1998.Prince married, lost two children, and was said to predict 9/11 on stage.Given the personal loss and the looming threat of terrorism, words like "suicide", "waste", "mad", "hurt", and many more not captured in this graph are very indicative of the actual events.
These words are powerful and provide real insight into the sentiment of that time period.
call unnest_tokens() passing the token argument ngrams. Since you're just looking at bigrams (two consecutive words), pass n = 2. Use prince_bigrams to store the results.
In order to remove the stop words and undesirable words, you'll want to break the bigrams apart and filter out what you don't want, then use unite() to put the word pairs back together.
This makes it easy to visualize the most common bigrams per decade.
In the year 1998 from the bigram we can predict that there is a shift of phase from romance to religion and children.

- **POLAR SENTIMENT OF WORDS PRECEDED BY NOT**

Polarity in sentiment analysis refers to identifying sentiment orientation (positive, neutral, and negative) in written or spoken language. The AFINN lexicon is used to perform sentiment analysis on word pairs, looking at how often sentiment-associated words are preceded by "not" or other negating words. Care is given a *false positive* sentiment because the "not" is ignored with single-word analysis. So if your analysis is based on unigrams and "alone" comes back as negative, the bigram "not alone" as you see above will have a reverse effect.

4.2 CODE:

```
install.packages('dplyr')  #data manipulation operations-filter,sort,aggregate
install.packages('ggplot2')#data visualization package
install.packages('gridExtra') #provides low-level functions to create graphical objects
install.packages('tidytext')  #Text mining package
install.packages('wordcloud2')#To make wordcloud

prince_orig <- read.csv("C:/Users/Hp/Desktop/Minor Project/prince_raw_data.csv",
stringsAsFactors = FALSE)

names(prince_orig)

prince <- prince_orig %>%
  select(lyrics = text, song, year, album, peak,
         us_pop = US.Pop, us_rnb = US.R.B)


fix.contractions <- function(doc) {
  # "won't" is a special case as it does not expand to "wo not"
  doc <- gsub("won't", "will not", doc)
  doc <- gsub("can't", "can not", doc)
  doc <- gsub("n't", " not", doc)
  doc <- gsub("'ll", " will", doc)
  doc <- gsub("'re", " are", doc)
  doc <- gsub("'ve", " have", doc)
  doc <- gsub("'m", " am", doc)
  doc <- gsub("'d", " would", doc)
  # 's could be 'is' or could be possessive: it has no expansion
  doc <- gsub("'s", "", doc)
  return(doc)
}
prince$lyrics <- sapply(prince$lyrics, fix.contractions)
removeSpecialChars <- function(x) gsub("[^a-zA-Z0-9 ]", " ", x)
prince$lyrics <- sapply(prince$lyrics, removeSpecialChars)
#convert to lower case
prince$lyrics <-sapply(prince$lyrics, tolower)
str(prince[139, ]$lyrics, nchar.max = 300)
summary(prince)

prince <- prince %>%
  mutate(decade =
           ifelse(prince$year %in% 1978:1979, "1970s",
                  ifelse(prince$year %in% 1980:1989, "1980s",
                         ifelse(prince$year %in% 1990:1999, "1990s",
                                ifelse(prince$year %in% 2000:2009, "2000s",
                                       ifelse(prince$year %in% 2010:2015, "2010s",
                                        "NA"))))))
prince <- prince %>%
  mutate(chart_level =
          ifelse(prince$peak %in% 1:10, "Top 10",
                ifelse(prince$peak %in% 11:100, "Top 100", "Uncharted")))

prince <- prince %>%
  mutate(charted =
          ifelse(prince$peak %in% 1:100, "Charted", "Uncharted"))



prince_data <- read.csv('prince_new.csv', stringsAsFactors = FALSE, row.names = 1)
glimpse(prince_data)

undesirable_words <- c("prince", "chorus", "repeat", "lyrics",
                     "theres", "bridge", "fe0f", "yeah", "baby",
                     "alright", "wanna", "gonna", "chorus", "verse",
                     "whoa", "gotta", "make", "miscellaneous", "2",
                     "4", "ooh", "uurh", "pheromone", "poompoom", "3121",
                     "matic", " ai ", " ca ", " la ", "hey", " na ",
                     " da ", " uh ", " tin ", "  ll", "transcription",
                     "repeats", "la", "da", "uh", "ah")
prince_tidy <- prince_data %>%
  unnest_tokens(word, lyrics) %>% #Break the lyrics into individual words
  filter(!word %in% undesirable_words) %>% #Remove undesirables
  filter(!nchar(word) < 3) %>% #Words like "ah" or "oo" used in music
  anti_join(stop_words) #Data provided by the tidytext package
glimpse(prince_tidy)


#****************Sentiment Analysis*******************
word_summary <- prince_tidy %>%
  mutate(decade = ifelse(is.na(decade),"NONE", decade)) %>%
  group_by(decade, song) %>%
  mutate(word_count = n_distinct(word)) %>%
  select(song, Released = decade, Charted = charted, word_count) %>%
  distinct() %>% #To obtain one record per song
  ungroup()
pirateplot(formula =  word_count ~ Released + Charted, #Formula
          data = word_summary, #Data frame
          xlab = NULL, ylab = "Song Distinct Word Count", #Axis labels
          main = "Lexical Diversity Per Decade", #Plot title
          pal = "google", #Color scheme
          point.o = .2, #Points
          avg.line.o = 1, #Turn on the Average/Mean line
          theme = 0, #Theme
          point.pch = 16, #Point `pch` type
```

```
           point.cex = 1.5, #Point size
           jitter.val = .1, #Turn on jitter to see the songs better
           cex.lab = .9, cex.names = .7) #Axis label size

songs_year <- prince_data %>%
  select(song, year) %>%
  group_by(year) %>%
  summarise(song_count = n())

id <- seq_len(nrow(songs_year))
songs_year <- cbind(songs_year, id)
label_data = songs_year
number_of_bar = nrow(label_data) #Calculate the ANGLE of the labels
angle = 90 - 360 * (label_data$id - 0.5) / number_of_bar #Center things
label_data$hjust <- ifelse(angle < -90, 1, 0) #Align label
label_data$angle <- ifelse(angle < -90, angle + 180, angle) #Flip angle
ggplot(songs_year, aes(x = as.factor(id), y = song_count)) +
  geom_bar(stat = "identity", fill = alpha("purple", 0.7)) +
  geom_text(data = label_data, aes(x = id, y = song_count + 10, label = year, hjust = hjust), color =
"black", alpha = 0.6, size = 3, angle =  label_data$angle, inherit.aes = FALSE ) +
  coord_polar(start = 0) +
  ylim(-20, 150) + #Size of the circle
  theme_minimal() +
  theme(axis.text = element_blank(),
        axis.title = element_blank(),
        panel.grid = element_blank(),
        plot.margin = unit(rep(-4,4), "in"),
        plot.title = element_text(margin = margin(t = 10, b = -10)))



#***************Sentiment Lexicons and Lyrics**************
#Calculating word count per lexicon
new_sentiments <- sentiments %>% #From the tidytext package
  filter(lexicon != "loughran") %>% #Remove the finance lexicon
  mutate( sentiment = ifelse(lexicon == "AFINN" & score >= 0, "positive",
                             ifelse(lexicon == "AFINN" & score < 0,
                                    "negative", sentiment))) %>%
  group_by(lexicon) %>%
  mutate(words_in_lexicon = n_distinct(word)) %>%
  ungroup()

new_sentiments %>%
  group_by(lexicon, sentiment, words_in_lexicon) %>%
  summarise(distinct_words = n_distinct(word)) %>%
  ungroup() %>%
  spread(sentiment, distinct_words) %>%
  mutate(lexicon = color_tile("lightblue", "lightblue")(lexicon),
         words_in_lexicon = color_bar("lightpink")(words_in_lexicon)) %>%
  my_kable_styling(caption = "Word Counts Per Lexicon")

#NRC graphical sentiment analysis
nrc_plot <- prince_nrc %>%
  group_by(sentiment) %>%
  summarise(word_count = n()) %>%
  ungroup() %>%
  mutate(sentiment = reorder(sentiment, word_count)) %>%
  #Use `fill = -word_count` to make the larger bars darker
  ggplot(aes(sentiment, word_count, fill = -word_count)) +
  geom_col() +
  guides(fill = FALSE) + #Turn off the legend
  theme_lyrics() +
  labs(x = NULL, y = "Word Count") +
  scale_y_continuous(limits = c(0, 15000)) + #Hard code the axis limit
  ggtitle("Prince NRC Sentiment") +
  coord_flip()

#plot the sentiments for the songs
plot(nrc_plot)

circos.clear()
#Set the gap size
circos.par(gap.after = c(rep(5, length(unique(decade_mood[[1]])) - 1), 15,
                         rep(5, length(unique(decade_mood[[2]])) - 1), 15))
chordDiagram(decade_mood, grid.col = grid.col, transparency = .2)
title("Relationship Between Mood and Decade")



#Sign Of The Times - Looking for the modd of a particular song
prince_nrc %>%
  filter(song %in% "sign o the times") %>%
  group_by(sentiment) %>%
  summarise(word_count = n()) %>%
  ungroup() %>%
  mutate(sentiment = reorder(sentiment, word_count)) %>%
  ggplot(aes(sentiment, word_count, fill = -word_count)) +
  geom_col() +
  guides(fill = FALSE) +
  theme_minimal() + theme_lyrics() +
  labs(x = NULL, y = "Word Count") +
  ggtitle("Sign O' The Times NRC Sentiment") +
  coord_flip()


#Bigrams Per Decade
```

```
prince_bigrams <- prince_data %>%
  unnest_tokens(bigram, lyrics, token = "ngrams", n = 2)

bigrams_separated <- prince_bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")

bigrams_filtered <- bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  filter(!word1 %in% undesirable_words) %>%
  filter(!word2 %in% undesirable_words)

#Because there is so much repetition in music, also filter out
#the cases where the two words are the same
bigram_decade <- bigrams_filtered %>%
  filter(word1 != word2) %>%
  filter(decade != "NA") %>%
  unite(bigram, word1, word2, sep = " ") %>%
  inner_join(prince_data) %>%
  count(bigram, decade, sort = TRUE) %>%
  group_by(decade) %>%
  slice(seq_len(7)) %>%
  ungroup() %>%
  arrange(decade, n) %>%
  mutate(row = row_number())

bigram_decade %>%
  ggplot(aes(row, n, fill = decade)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~decade, scales = "free_y") +
  xlab(NULL) + ylab(NULL) +
  scale_x_continuous(  # This handles replacement of row
    breaks = bigram_decade$row, # Notice need to reuse data frame
    labels = bigram_decade$bigram) +
  theme_lyrics() +
  theme(panel.grid.major.x = element_blank()) +
  ggtitle("Bigrams Per Decade") +
  coord_flip()
```

4.3 RESULTS:

FIG.4.3.1 LEXICAL DIVERSITY PER DECADE

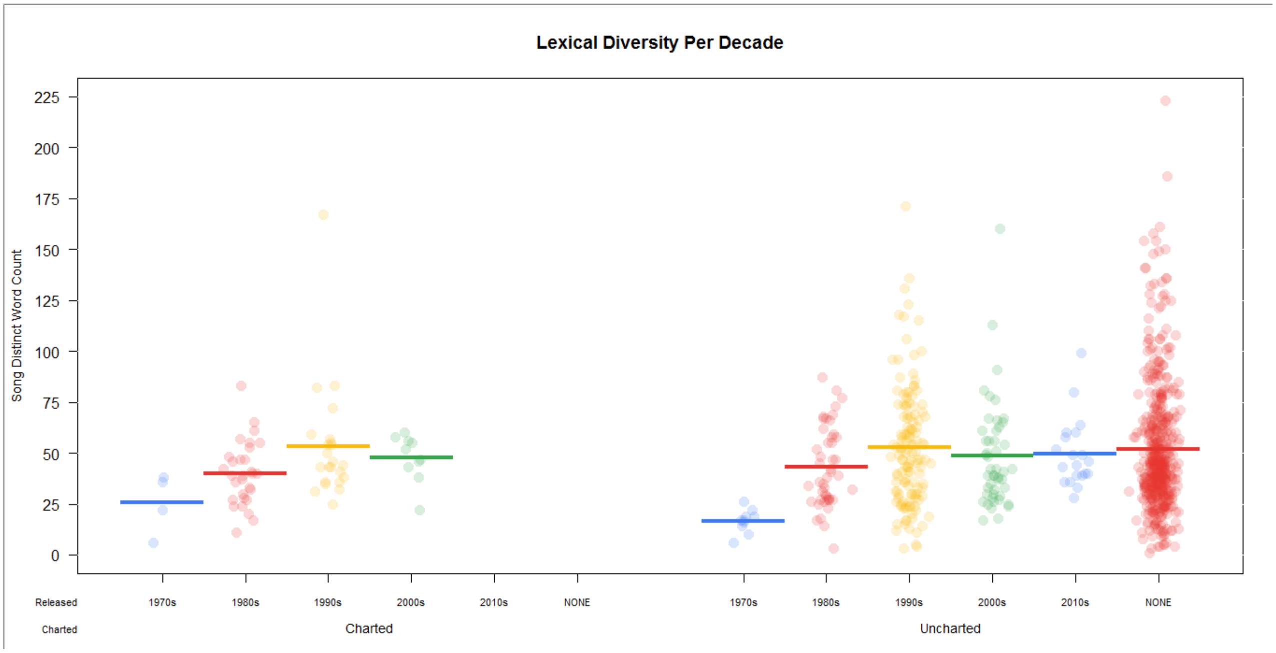


FIG.4.3.2 SONG COUNT PER YEAR AND RELATIONSHIP BETWEEN CHART AND DECADE

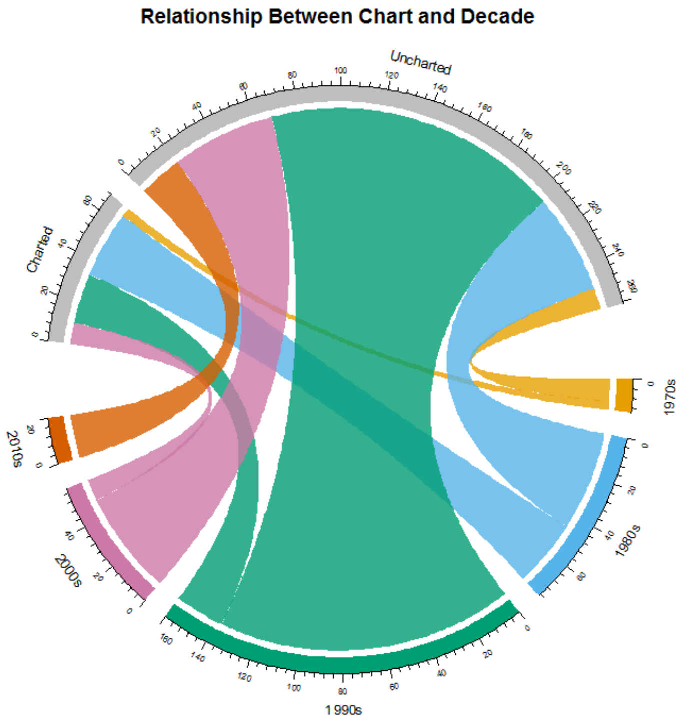


FIG.4.3.3 SONG COUNT PER YEAR

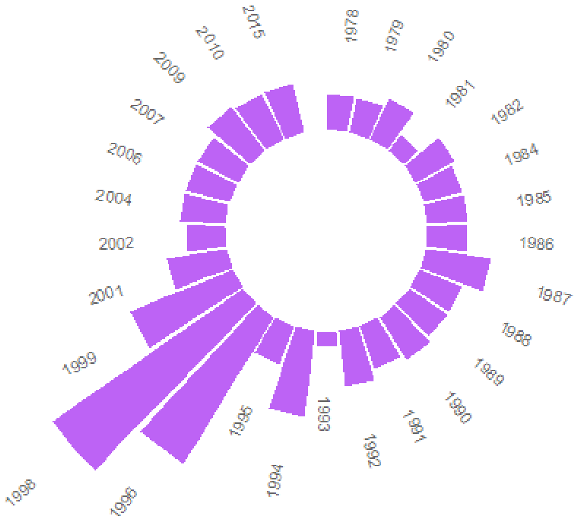
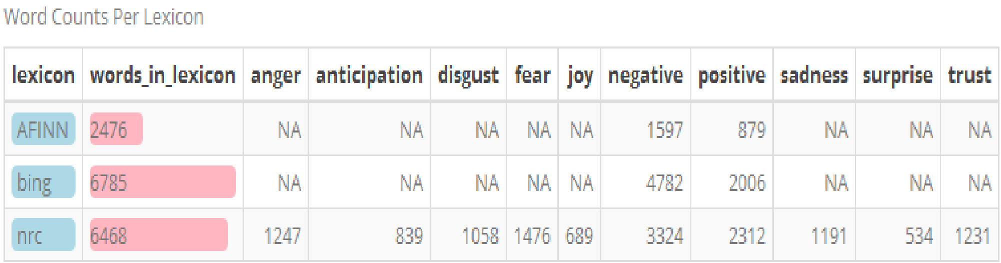


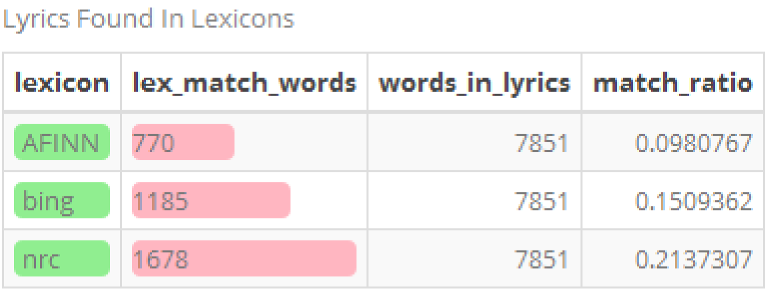
FIG.4.3.4 WORD COUNT PER LEXICON



Word Counts Per Lexicon

lexicon	words_in_lexicon	anger	anticipation	disgust	fear	joy	negative	positive	sadness	surprise	trust
AFINN	2476	NA	NA	NA	NA	NA	1597	879	NA	NA	NA
bing	6785	NA	NA	NA	NA	NA	4782	2006	NA	NA	NA
nrc	6468	1247	839	1058	1476	689	3324	2312	1191	534	1231

FIG.4.3.5 WORD MATCH RATIO FOR LEXICON



Lyrics Found In Lexicons

lexicon	lex_match_words	words_in_lyrics	match_ratio
AFINN	770	7851	0.0980767
bing	1185	7851	0.1509362
nrc	1678	7851	0.2137307

FIG.4.3.6 NRC SENTIMENT ANALYSIS

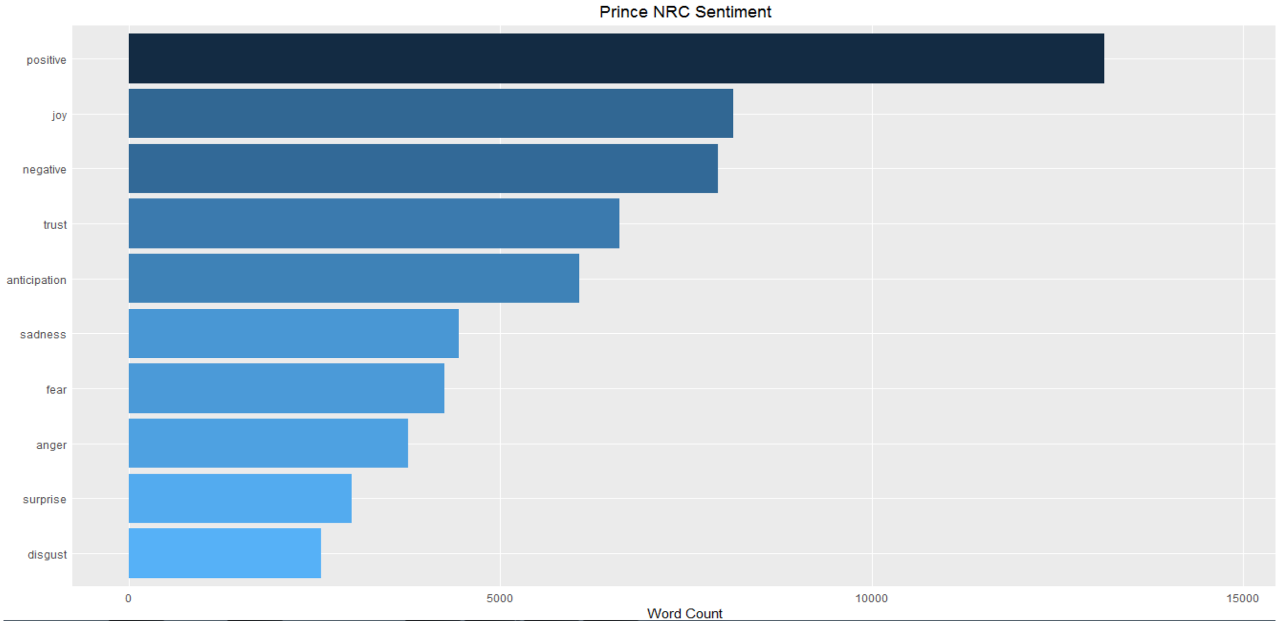


FIG.4.3.7 RELATIONSHIP BETWEEN MOOD AND DECADE

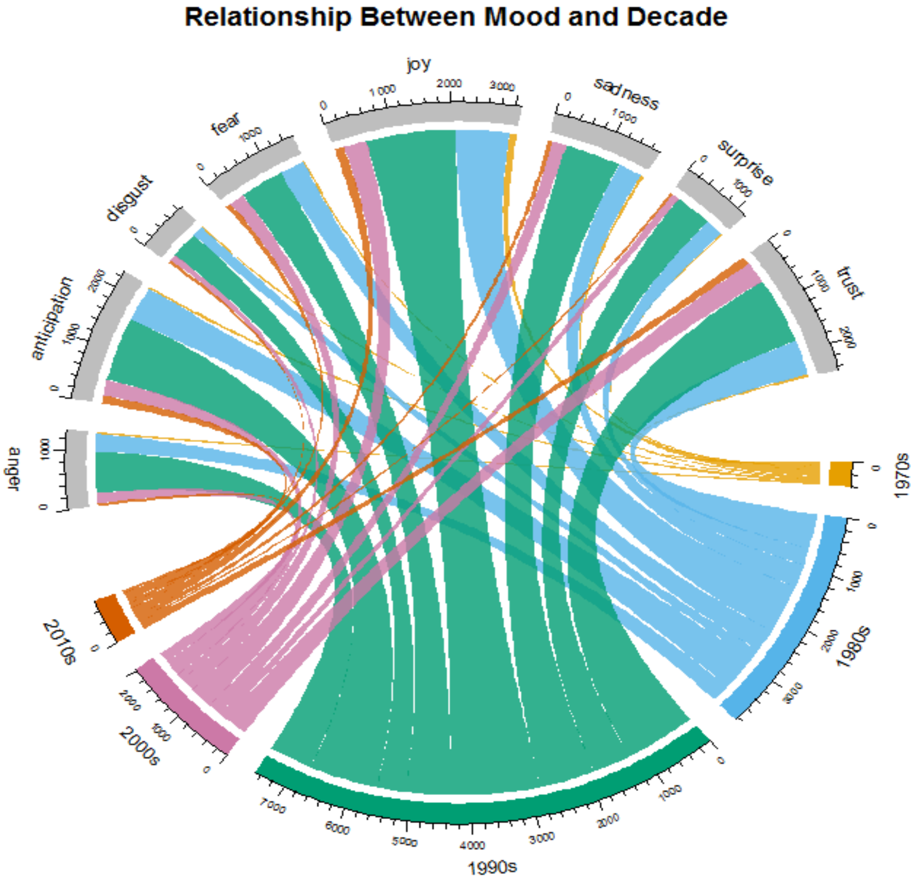


FIG.4.3.8 SENTIMENT BY LIFE EVENTS

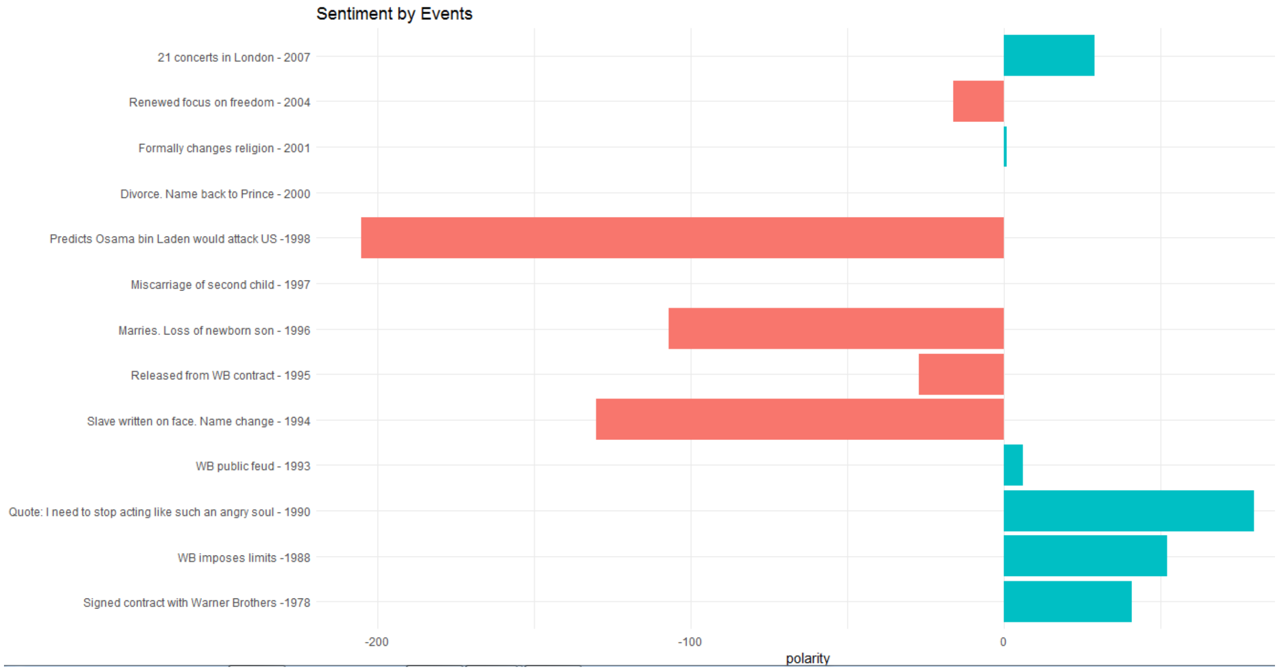


FIG.4.3.9 1998 NRC SENTIMENTS

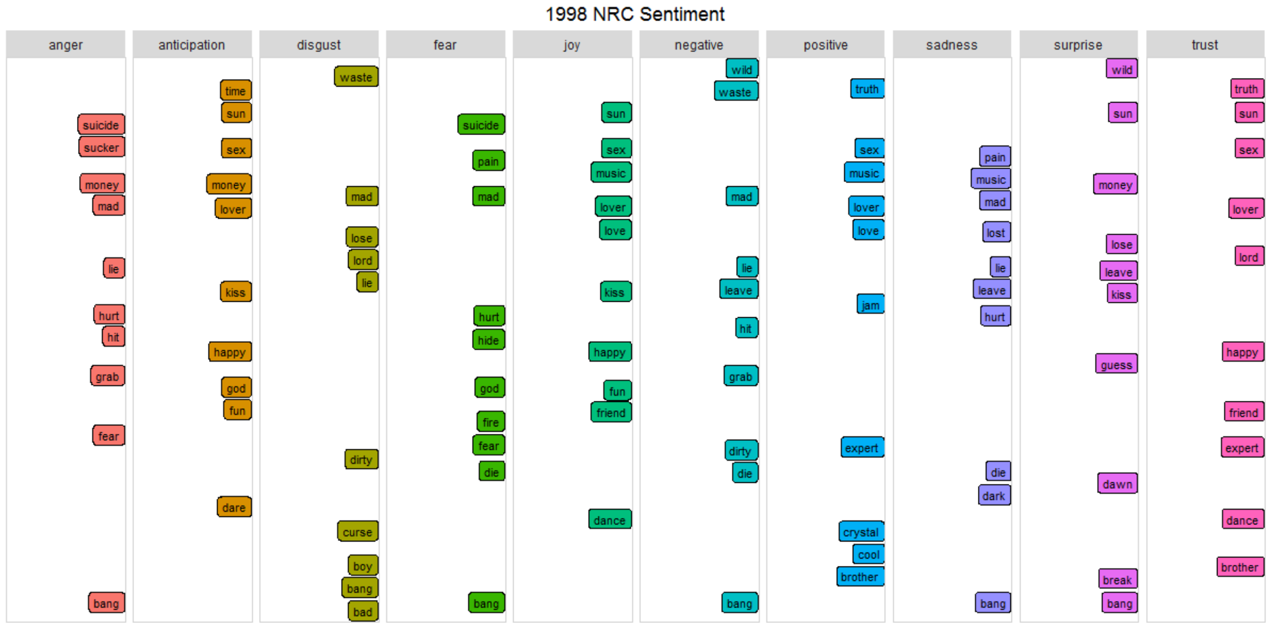


FIG.4.3.10 SIGN 0 TIMES NRC SENTIMENT

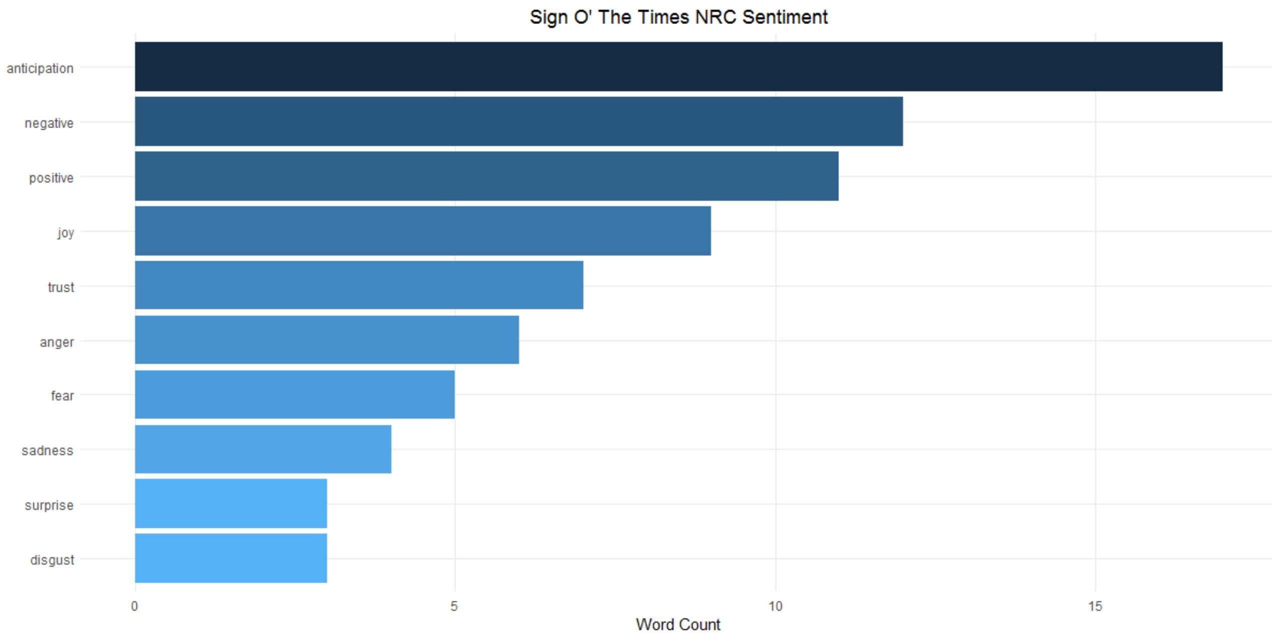


FIG.4.3.11 SONGS AND THEIR SENTIMENTS

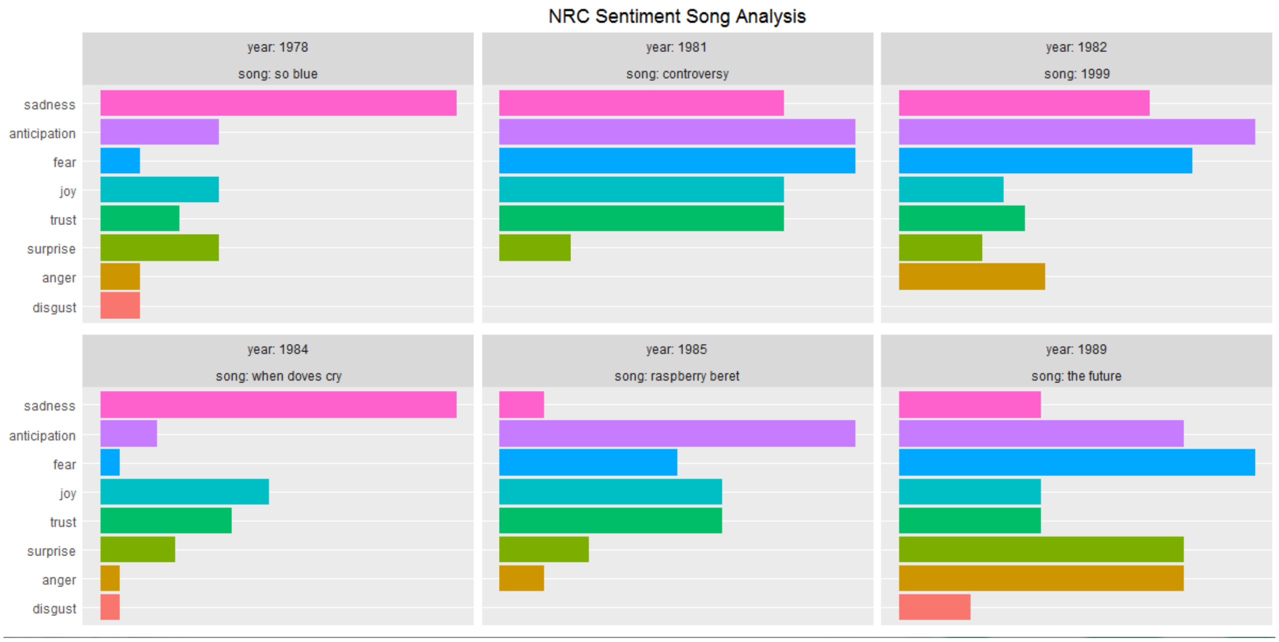


FIG.4.3.12 BIGRAMS PER DECADE

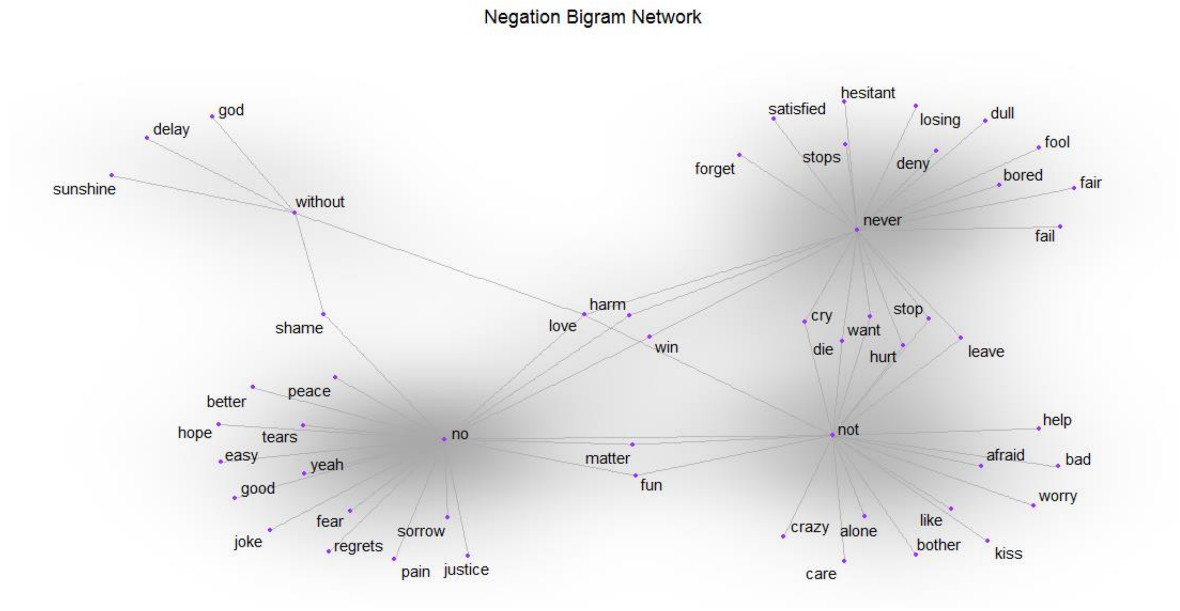


FIG.4.3.13 POLAR SENTIMENTS OF WORDS PRECEDED BY NOT

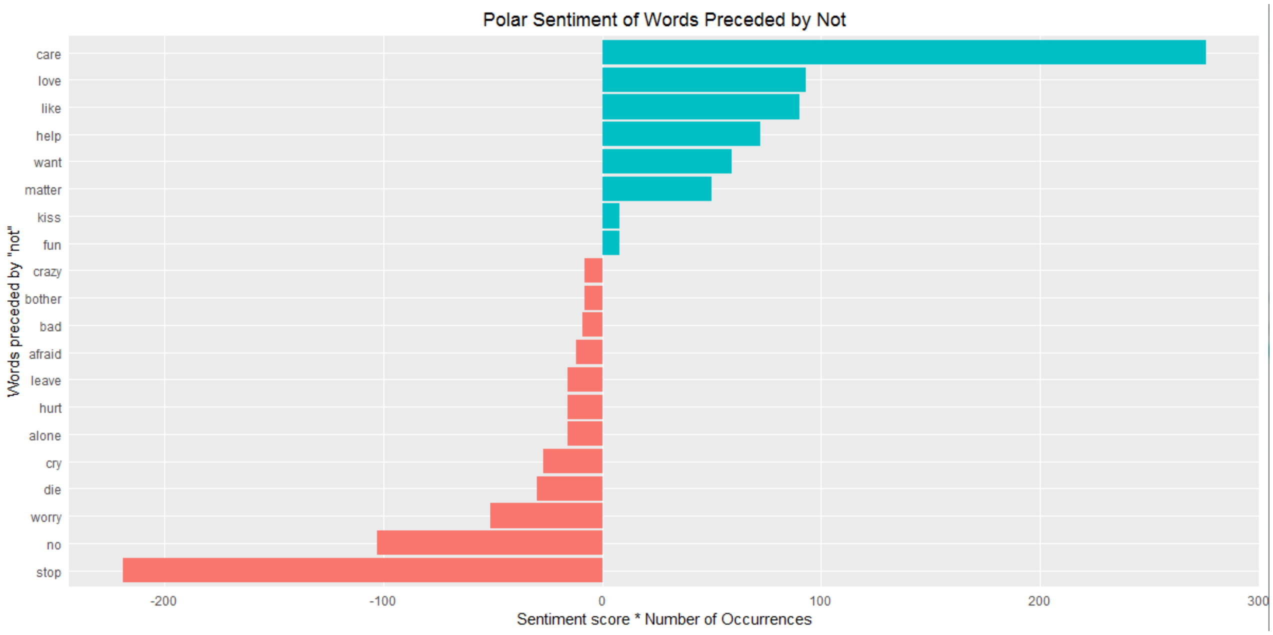
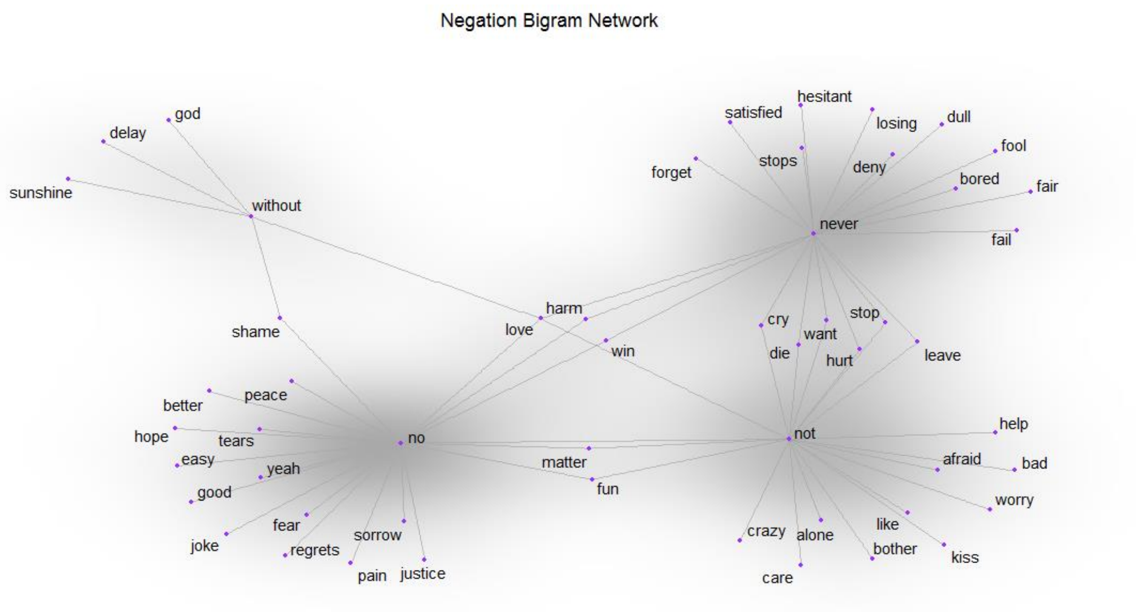


FIG.4.3.14 NEGATION BIGRAM NETWORK



CHAPTER 5
OTHER SPECIFICATIONS

5.1. ADVANTAGES:

- Sentiment analysis can pick up negative discussions, and give you real-time alerts.
- Identifying emotional polarity of songs.
- Track trends over time.
- Sentiment analysis lets you easily filter mentions by positivity and negativity, showing you which blazing fires to put on the "extinguish immediately" list and which slow smolders can wait a bit.

5.2. LIMITATIONS:

- Irony, humor or sarcasm are barely understood using sentiment analysis. They are usually misclassified.
- Sentiment analysis needs greater context(large data sets and lots of content).
- Cannot identify root cause of a review.

5.3. SPECIFICATIONS:

- Also we can use this to categorize songs and give recommendations to users on the basis of their past history.
- We can also do the sentiment analysis of the songs of different singers and categorize them according to different genre and mood.
- We can know the reviews of people for a particular movie or a book and do a sentiment analysis on that and can know their opinion about it.
- Also twitter sentiment analysis for tweets already prevails in the market for any trending topic.
- Also sentiment analysis can be done on surveys to know the actual result of a survey.

CHAPTER 6
CONCLUSION AND FUTURE WORK

We created a tidy dataset and analyzed basic information such as the lexical diversity and song counts per release year, and examined the relationship between release decade and whether a song hit the charts. We then explored some sentiment lexicons and how well they matched the lyrics. Next, we performed sentiment analysis on all songs in the dataset, sentiment over time, song level sentiment, and the impact of bigrams.

Is it possible to write a program to determine mood in lyrics.

How reliable are the results depends on a wide range of criteria such as the amount of data preparation, the choice of lexicon, the method of analysis, the quality of the source data, and so on. Comparing real life events, both personal and societal, can illuminate the mood of any lyric. Prince's polar sentiment seemed to slightly decline over time, yet overall, *joy* does seem to stand out. Charted songs seem to be more positive than uncharted songs. The claims of predicting 9/11 and his own death seem to eerily match his words.

CHAPTER 7
REFERENCES

1. **DATA CAMP:**
   https://www.datacamp.com/community/tutorials/sentiment-analysis-R#prepwork
   https://www.datacamp.com/community/tutorials/R-nlp-machine-learning

2. TWITTER :
   https:// tweet-sentiment-analysis-

3. https://image.slidesharecdn.com/finalyear-170810032309/95/tweet-sentiment-analysis-20-638.jpg?cb=1524725811

4. https://image.slidesharecdn.com/finalyear-170810032309/95/tweet-sentiment-analysis-20-638.jpg?cb=1524725811