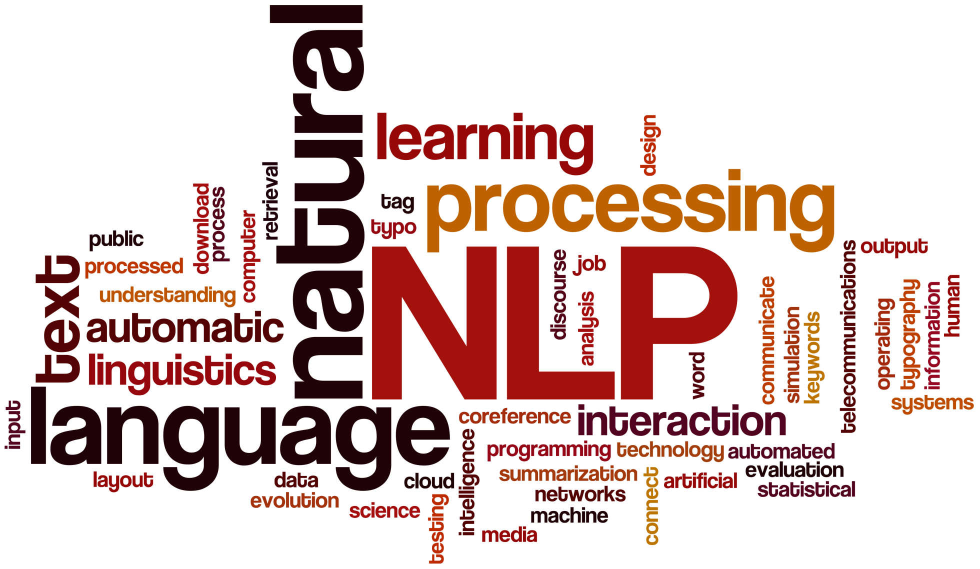
PROJECT REPORT

# Natural Language Processing (NLP) Phase 1 Final Project

| | |
|---|---|
| **Supervised By** | Dr Sauleh Etemadi |
| **Name** | Zahra Hosseini |
| **Email** | hosseini99.zahra@gmail.com |
| **Date** | May 11, 2021 |



# Contents

# List of Figures

# List of Tables

# 1. Data Source

Genius is an online community where users browse, rate, and create annotations for lyrics to help explain their meaning and context.
I used johnwmillr's fantastic LyricsGenius (version 3.0.1) package to access the Genius Lyrics API. At the first I use the version 3.0.0 which one of features that I need has deleted. so I move on version 3.0.1 to use new features.

Figure 1.1: Genius logo

## 1.1 Data gathering

First of all, I've signed up to the Genius website to generate a personal token then I collected a list of artists to find the artists' IDs from API. After this, I randomly collect some of the lyrics, the album's release date, and the artist's name, finally save them to the dataset.csv file.

## 1.2 Data Lableing

In the previous step, I saved the release date of songs. As I describe in the phase 0 project report, I want to label the music based on it before the 2000s or after the 2000s. my Unit on labeling is lyrics of songs So with a simple comparison, I've tagged before 2000's songs with 0 and after 2000's songs with 1.

Table 1.1: Example of data raw

| Name | Label | year | lyrics |
|---|---|---|---|
| Pink Floyd | 0 | 1979 | I am just a new boy A stranger ... |

# 2. File Structures

The structure of the files is as below. We have three folders, and the codes are available in the **src** folder in three parts. First of all, you need to go to the **data_collection.ipynb** to prepare the dataset file then we are ready for the next steps. Data collection may take some minutes, or you got **timeout error**, I turn on VPN to solve this problem, so I think you need to this too.

```
cs224n-final-project-data
├─ data
│  ├─ clean_words
│  │  ├─ clean_words_0.txt
│  │  └─ clean_words_1.txt
│  ├─ sentences
│  │  ├─ sents_0.txt
│  │  └─ sents_1.txt
│  ├─ words
│  │  ├─ words_0.txt
│  │  └─ words_1.txt
│  └─ dataset.csv
├─ src
│  ├─ statics.ipynb
│  ├─ data_collection.ipynb
│  └─ data_cleaning.ipynb
└─ Statics
   └─ Images
      ├─ Clean_words_garph.png
      ├─ Clean_words_table.png
      └─ ...
```

# 3. Preprocess

## 3.1 Sentence Splitting

For split and tokenization the senteces i used the **Natural Language Toolkit (NLTK)** python package. NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces.
A good useful first step is to split the data into sentences. Some modeling tasks prefer input to be in the form of paragraphs or sentences, such as word2vec. we could first split our data into sentences, split each sentence into words, then save each sentence to file, one per line.
NLTK provides the **sent_tokenize()** function to split text into sentences.

## 3.2 Word Splitting

NLTK provides a function called **word_tokenize()** for splitting strings into tokens (nominally words).
It splits tokens based on white space and punctuation. For example, commas and periods are taken as separate tokens. Contractions are split apart (e.g. “What’s” becomes “What” “‘s“). Quotes are kept, and so on.

## 3.3 data cleaning

The first problem of the dataset was the instrumental [1] songs. The API doesn’t provide a feature to check that a song is instrumental or not.I realized that the lyrics of instrumental songs save as a null string (""). To solve this problem, I check the lyrics’ size before adding a new row to the dataset. This solution was the first step of data cleaning and decreased the size of the dataset by about 20-25 units.

Table 3.1: Size of dataset before/after cleaning

| Before cleaning | After cleaning |
|---|---|
| 546 | 521 |

To more cleaning data, I’ve done below steps:

[1] An instrumental is a recording normally without any vocals, although it might include some inarticulate vocals, such as shouted backup vocals in a big band setting.

- **Filter Out Punctuation**
  We can filter out all tokens that we are not interested in, such as all standalone punctuation.

  This can be done by iterating over all tokens and only keeping those tokens that are all alphabetic. Python has the function **isalpha()** that can be used.

- **Filter out Stop Words (and Pipeline)**
  Stop words are those words that do not contribute to the deeper meaning of the phrase. They are the most common words such as: “the“, “a“, and “is“. NLTK provides a list of commonly agreed upon stop words for English.

- **Stem Words**
  Stemming refers to the process of reducing each word to its root or base. For example “fishing,” “fished,” “fisher” all reduce to the stem “fish.”a popular and long-standing method is the Porter Stemming algorithm. This method is available in NLTK via the **PorterStemmer** class.
  After seeing the result of this step, I decide to skip this step because the algorithm changes some words negatively. For example, It turns the word: "unnecessary" to "necessari" or removes the last character of the word, for example, "divide" to " divid".

Table 3.2: words count before/after cleaning

| before 2000’s words | after 2000’s words | before 2000’s words(clean) | after 2000’s words(clean) |
|---|---|---|---|
| 85326 | 102860 | 35704 | 40275 |

## 3.4 Unit

Unit of the dataset is music lyrics. each song has a lyrics and it is a unit for my dataset.

# 4. Analysis

## 4.1 units

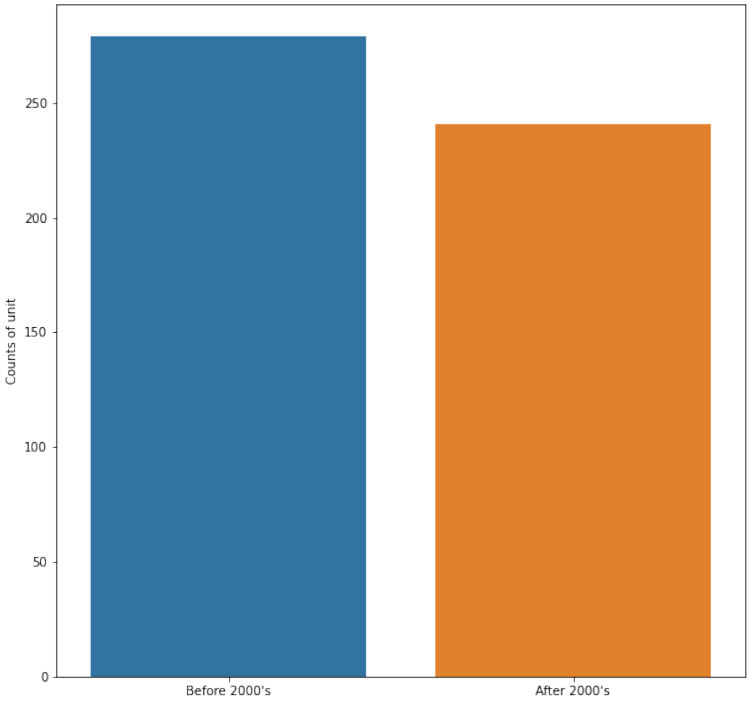


Figure 4.1: Units per Label

Table 4.1: words count before/after cleaning

| before 2000’s units | after 2000’s units |
|---|---|
| 279 | 241 |

## 4.2 Sentences

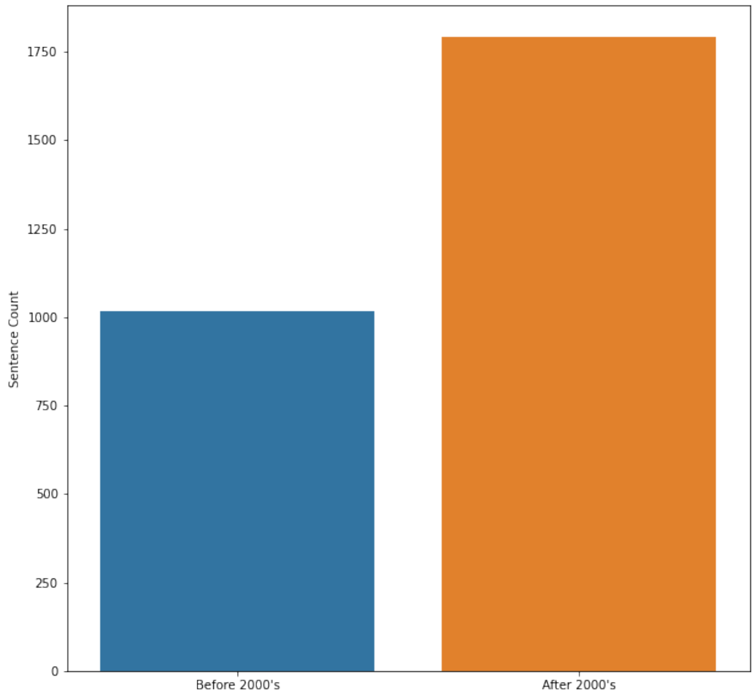


Figure 4.2: Sentences per Label

Table 4.2: words count before/after cleaning

| before 2000’s sentences | after 2000’s sentences |
|---|---|
| 1017 | 1792 |

## 4.3 Words

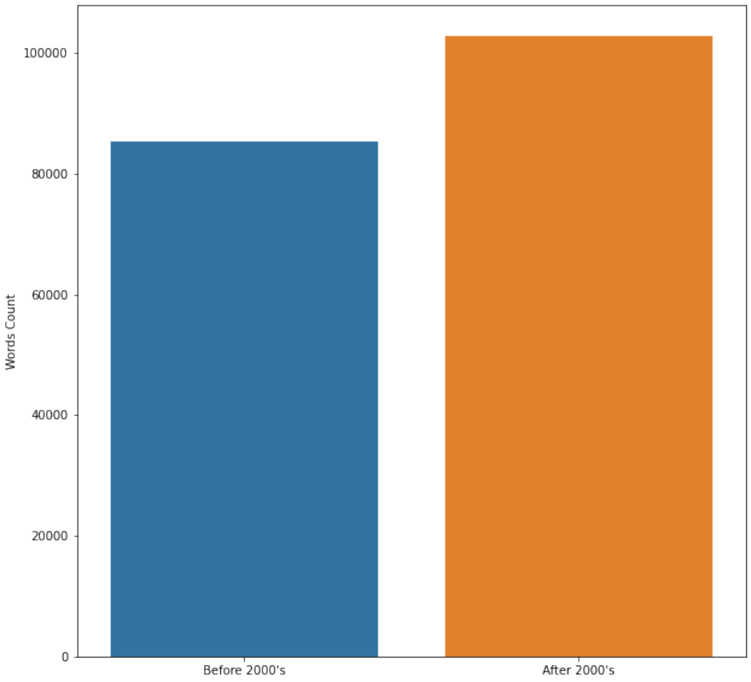


Figure 4.3: Sentences per Label

Table 4.3: words count

| before 2000’s words | after 2000’s words |
|---|---|
| 85326 | 102860 |

## 4.4 Distinct Words

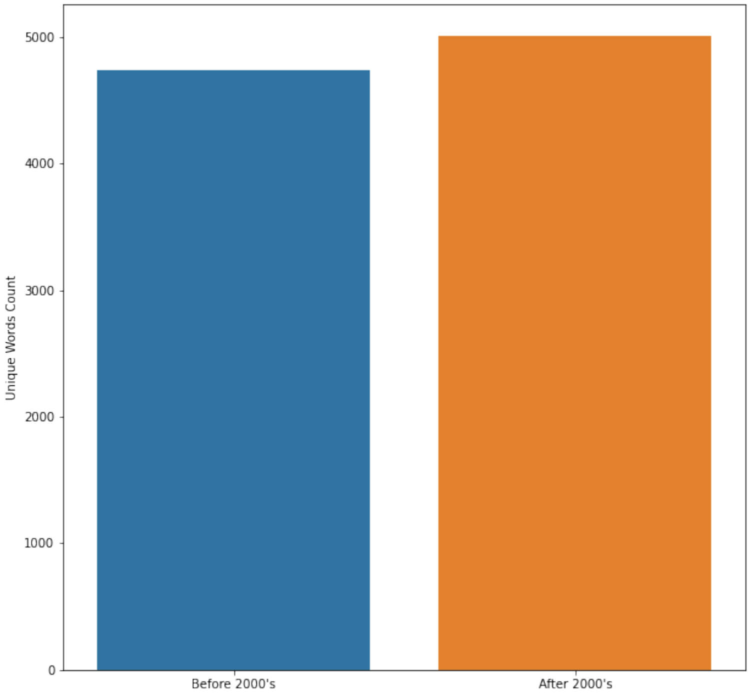


Figure 4.4: Distinct words per Label

Table 4.4: words count

| before 2000’s words | after 2000’s words |
|---|---|
| 4742 | 5010 |

## 4.5 Common-Uncommon Distinct Words

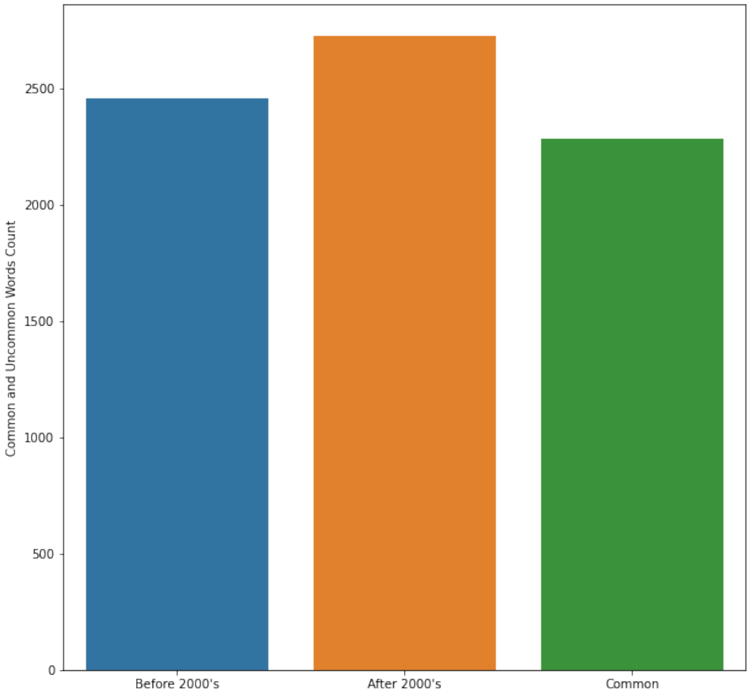


Figure 4.5: Distinct words per Label

Table 4.5: Common-Uncommon Distinct Words

| Uncommon before 2000’s words | Uncommon after 2000’s words | common words |
|---|---|---|
| 2457 | 2725 | 2285 |

## 4.6 Top-10 Uncommon Words

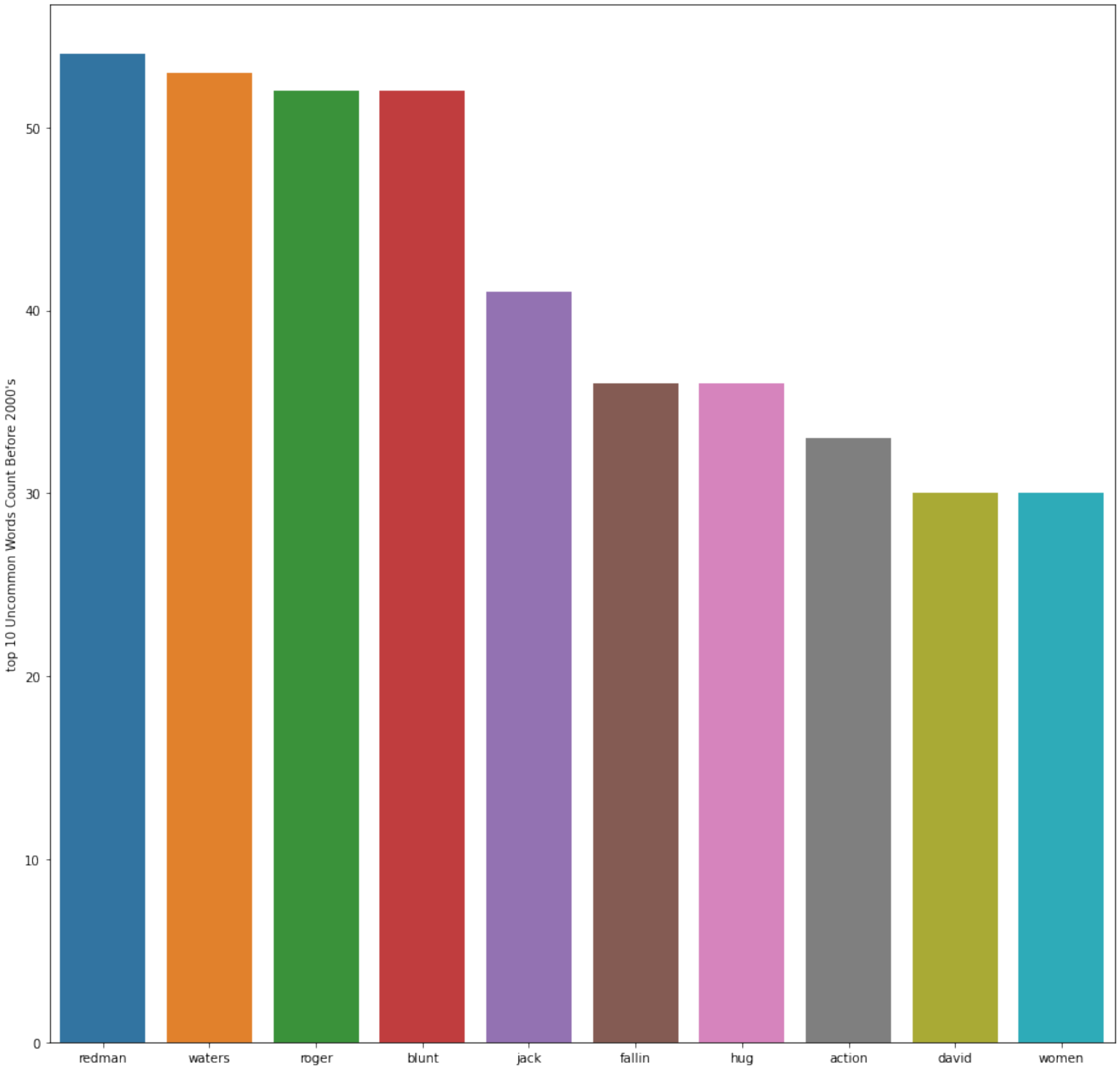


Figure 4.6: Top-10 Uncommon Words before 2000’s

Table 4.6: Top-10 Uncommon Words before 2000’s

| redman | waters | roger | bluent | jack | fallin | hug | action | david | women |
|---|---|---|---|---|---|---|---|---|---|
| 54 | 53 | 52 | 52 | 41 | 36 | 36 | 33 | 30 | 30 |

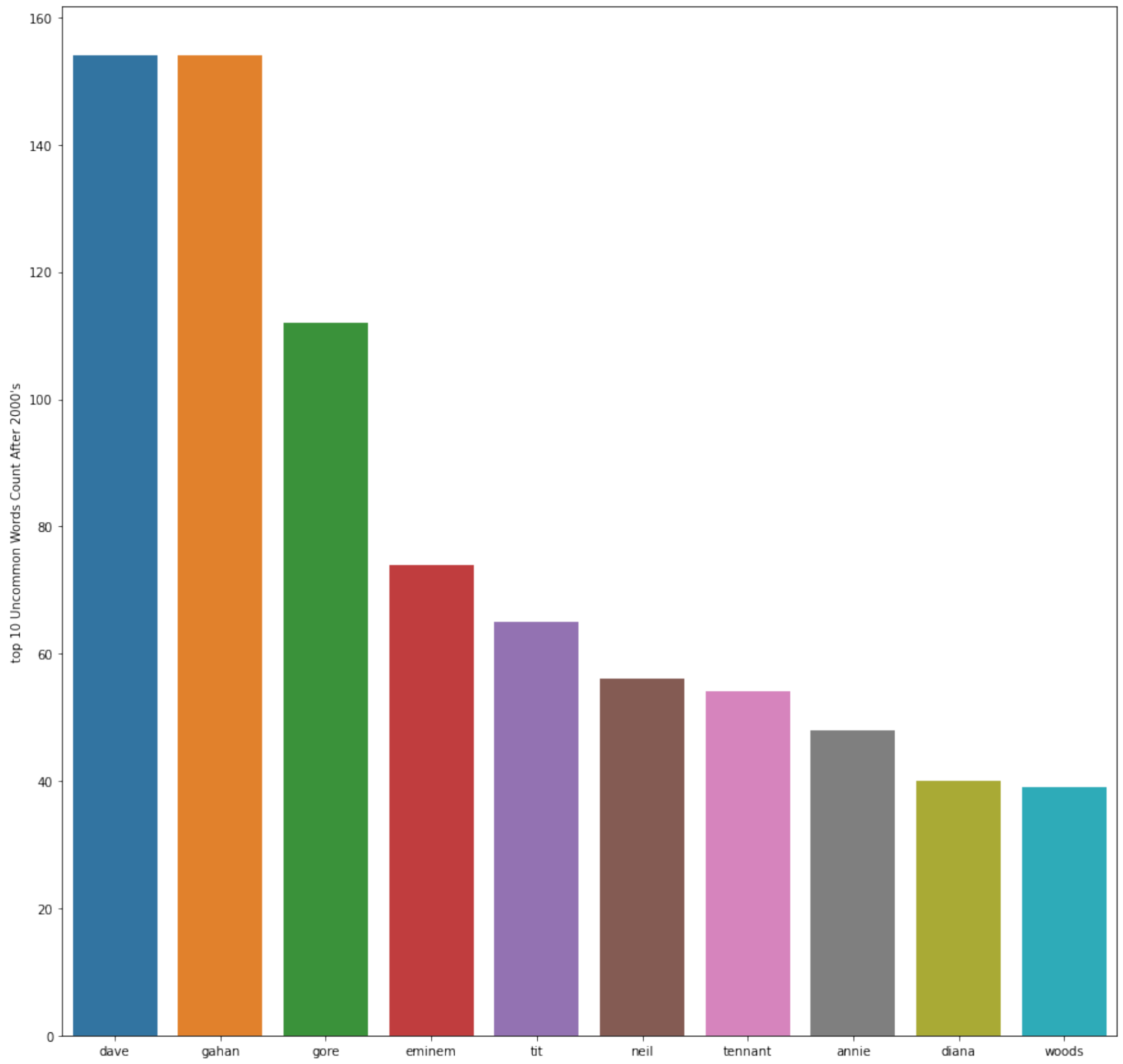


Figure 4.7: Top-10 Uncommon Words after 2000’s

Table 4.7: Top-10 Uncommon Words after 2000’s

| dave | gahan | gore | eminem | tit | neil | tennat | annie | diana | woods |
|---|---|---|---|---|---|---|---|---|---|
| 154 | 154 | 112 | 74 | 66 | 56 | 54 | 45 | 40 | 30 |

## 4.7 RNF Top-10 common Words

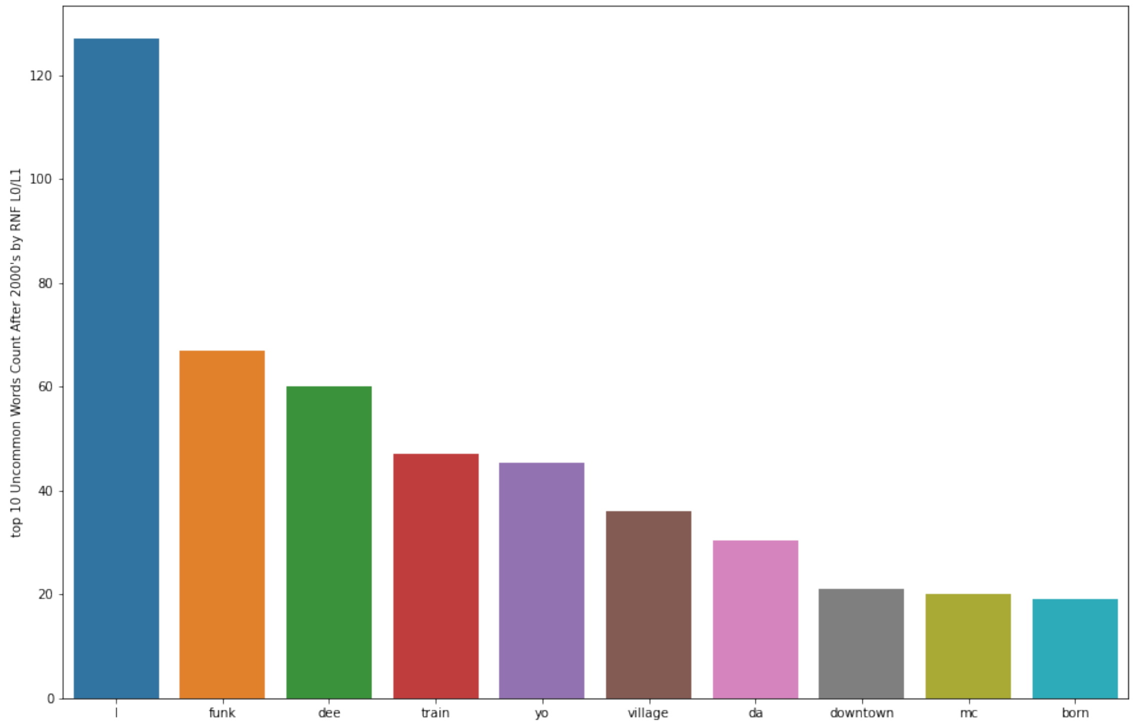


Figure 4.8: RNF Top-10 common Words before 2000’s

Table 4.8: RNF Top-10 common Words before 2000’s

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| I | funk | dee | train | you | village | da | downtown | mc | born |
| 127 | 67 | 60 | 47 | 45 | 36 | 30 | 21 | 20 | 19 |

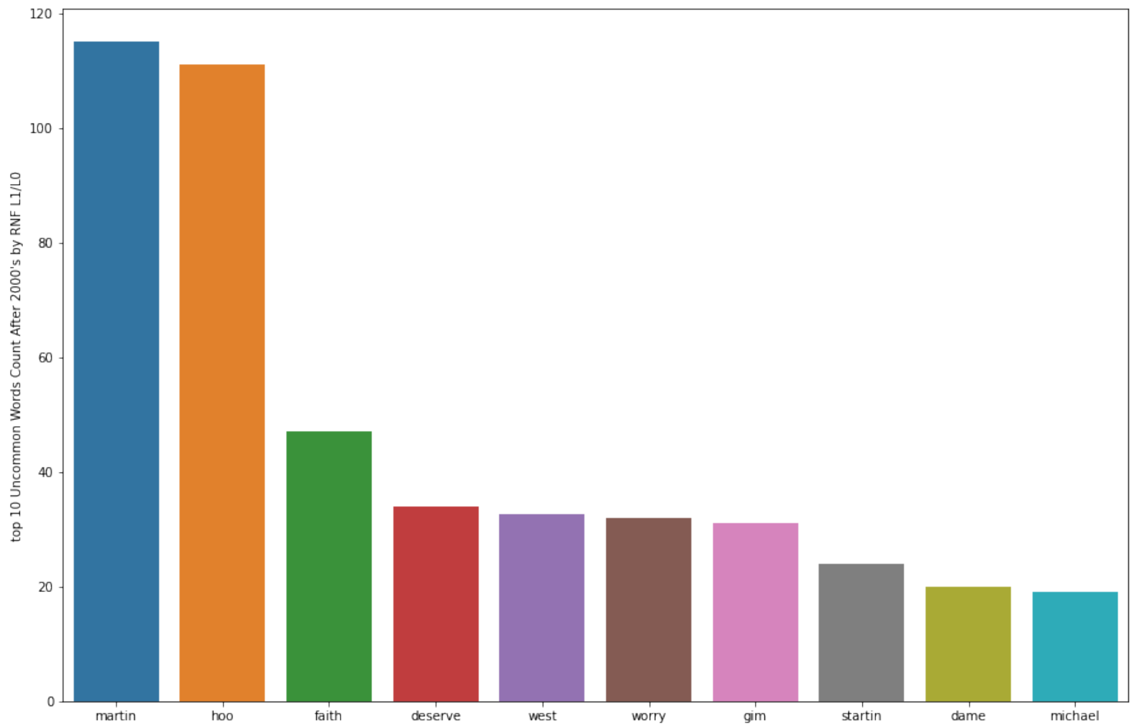


Figure 4.9: RNF Top-10 common Words after 2000's

Table 4.9: RNF Top-10 common Words after 2000's

| dave | gahan | gore | eminem | tit | neil | tennat | annie | diana | woods |
|---|---|---|---|---|---|---|---|---|---|
| 115 | 111 | 47 | 34 | 32 | 32 | 31 | 24 | 20 | 19 |

## 4.8 TF-IDF Top-10 common Words

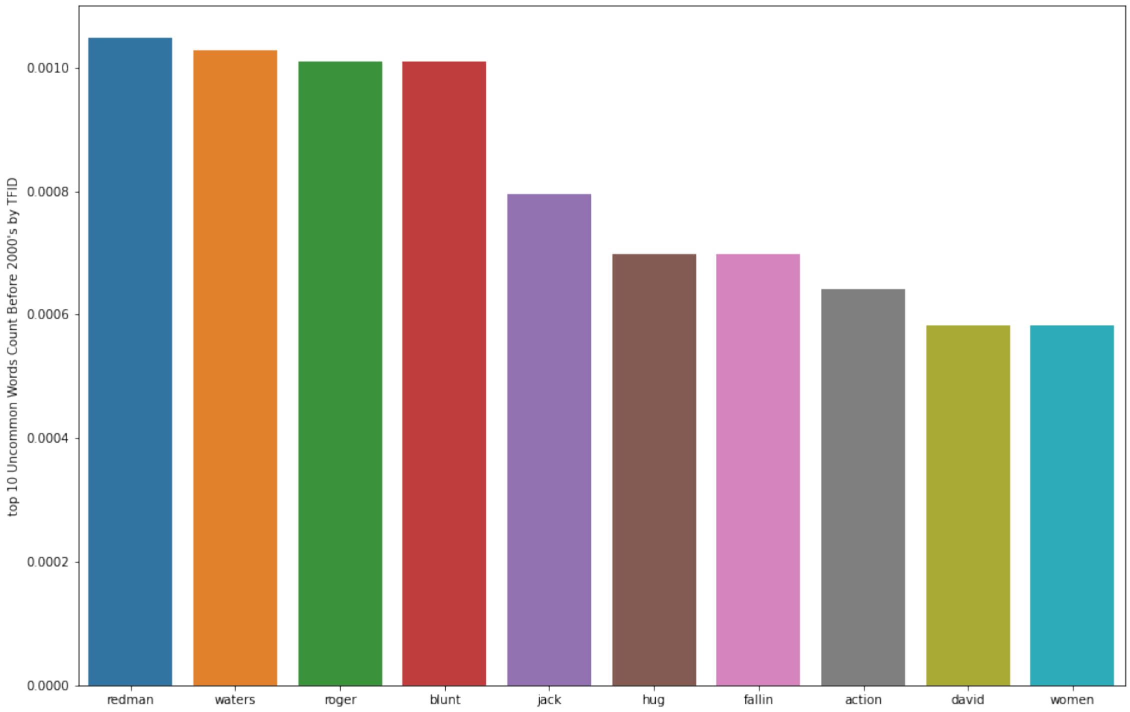


Figure 4.10: TF-IDF Top-10 common Words before 2000’s

Table 4.10: TF-IDF Top-10 common Words before 2000’s

| martin | hoo | faith | deserve | west | worry | gim | startin | dame | micheal |
|---|---|---|---|---|---|---|---|---|---|
| 0.00104 | 0.00102 | 0.0010 | 0.0009 | 0.0007 | 0.00069 | 0.0005 | 0.00063 | 0.0005 | 0.0005 |

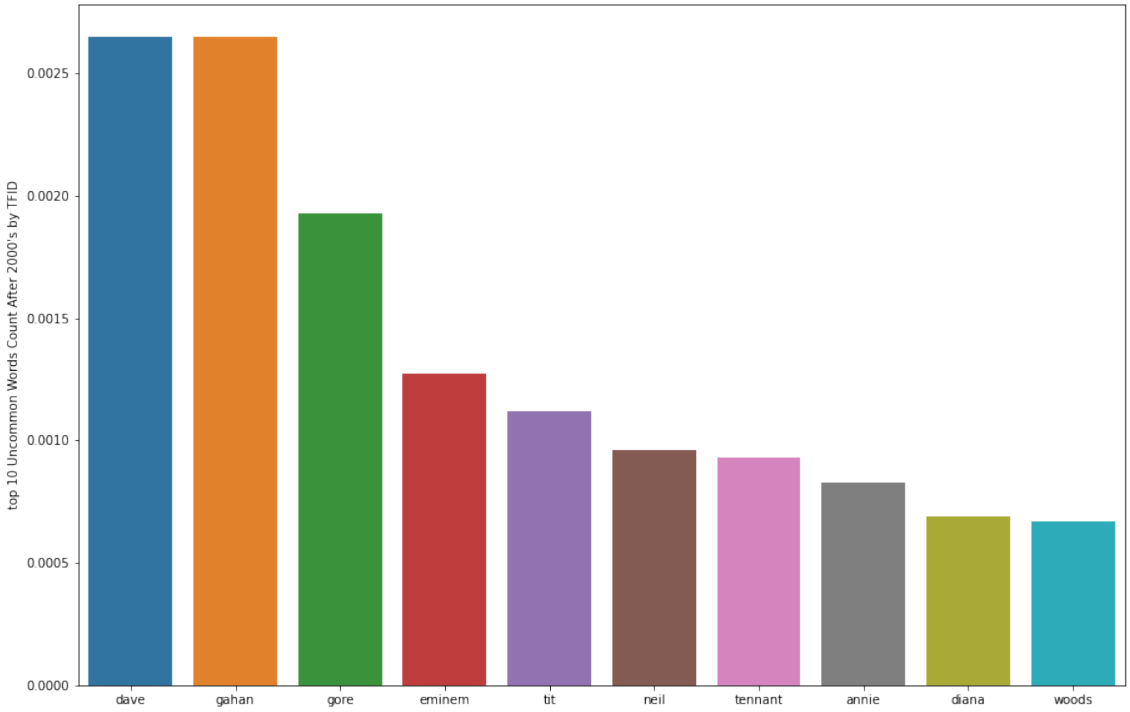


Figure 4.11: TF-IDF Top-10 common Words after 2000’s

Table 4.11: TF-IDF Top-10 common Words after 2000’s

| dave | gahan | gore | eminem | tit | neil | tennat | annie | diana | woods |
|---|---|---|---|---|---|---|---|---|---|
| 0.0026 | 0.0024 | 0.0019 | 0.00128 | 0.00118 | 0.0009 | 0.0008 | 0.0007 | 0.0006 | 0005 |

## 4.9 Histogram

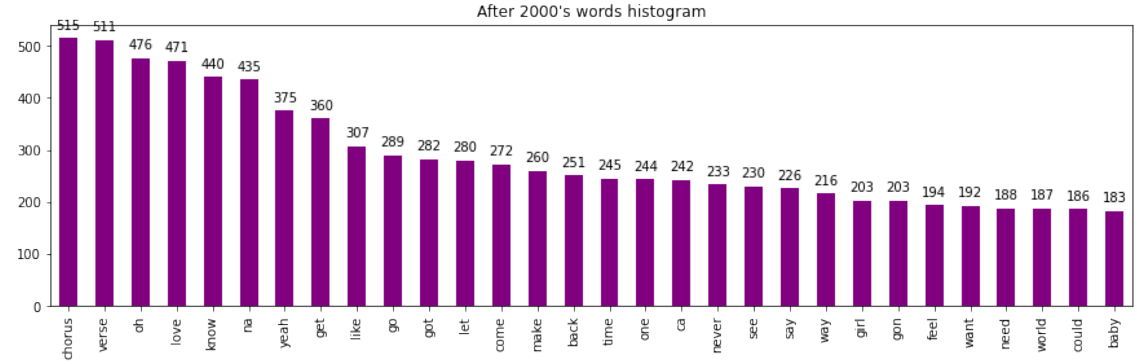


Figure 4.12: Top-10 Histogram after 2000's

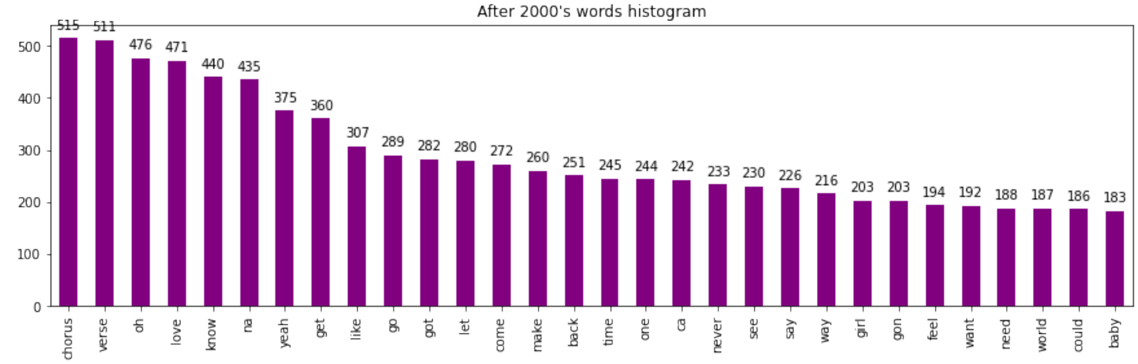


Figure 4.13: Top-10 Histogram before 2000's