Which Linguistics Features Affect the Popularity of Songs from Then till Now

Mei-Shin Wu, Roshanak Hamidi

September 2017

1 Abstract

In this paper we compared two sets of songs from three different years. The goal was to observe the language variation over time. We observed noticeable changes from 1975 to 2015. In order to track these changes, linguistic features from python's Natural Language Toolkit (NLTK) along with statistical analysis were applied.

2 Introduction

In this paper, we applied python's NLTK in order to find out the lexical commonalities and variations of song texts which belong to three different generations: *baby boomers*, generation X and generation Y[1]. It should also be mentioned that the estimated population for baby boomers, generation X and generation Y in the US is 80, 51 and 75 Million respectively.
We are interested in finding out:

- If the content of the lyrics has changed and if there has been a change in song tastes throughout these years.
- What are the so-called differences?

With this project, we hope that the linguistics differences between generations will be more comprehensible.

[1] According to Harvard Center, babies who are born from 1945 to 1965 are called baby boomers, those who are born from 1965 to 1984 are called generation X and babies who are born from 1982 to 2004 are considered as generation Y

3 Method

Our hypothesis has been formed based on the so-called generation gap. We predict that changes can be spotted in the content of the lyrics. In order to do that, several Python packages were involved; however, NLTK played an essential role in this project. NLTK has been used for frequency distribution, tokenization, part of speech tagging and extracting function words. In the following sections we explain the code and the analysis in detail[2].

3.1 Corpus

The Billboard Hot 100 is the standard American music industry record chart for music from all genres. We selected our corpus from the favourite and least favourite 10 English songs of 1975, 1995 and 2015. These years were selected taking into account the young generation of their time. (e.g. a baby who was born in 1945 will be 30 years old and supposedly music from 1975 should be a sample of their taste).
Since The Billboard Hot 100 also contains non-English songs, we also excluded those songs which were in any language other than English. Non-English songs were replaced by the next-ranked song.

Rank	1975		Rank of 1995		Rank of 2015	
	Artist	Song	Artist	Song	Rank	Artist
1	Captain	Love Will Keep Us Together	Coolio	Gangsta's Paradise	Mark Ronson	Uptown Funk
2	Glen Campbell	Rhinestone Cowboy	TLC	Waterfalls	Ed Sheeran	Thinking Out Loud
3	Elton John	Philadelphia Freedom	TLC	Creep	Wiz Khalifa	See You Again
4	Frankie Valli	My Eyes Adored You	Seal	Kiss from a Rose	Fetty Wap	Trap Queen
5	Earth, Wind & Fire	Shining Star	Boyz II Men	On Bended Knee	Maroon 5	Sugar
6	David Bowie	Fame	Real McCoy	Another Night	Walk The Moon	Shut Up and Dance
7	Neil Sedaka	Laughter in the Rain	Mariah Carey	Fantasy	Taylor Swift	Blank Space
8	Eagles	One of These Nights	Madonna	Take a Bow	Silento	Watch Me
9	John Denver	Thank God I'm a Country Boy	Monica	Don't Take It Personal	Weeknd	Earned It
10	Bee Gees	Jive Talkin'	Montell Jordan	This Is How We Do It	George Ezra	Budapest
90	Leo Sayer	Long Tall Glasses	Melissa Etheridge	Like the Way I Do	Weeknd	The Hills
91	Ray Stevens	Misty	FireHouse	I Live My Life for You	Sia	Chandelier
92	Elton John	Someone Saved My Life Tonight	Stevie B	Dream About You	Kelly Clarkson	Heartbeat Song
93	Neil Sedaka	Bad Blood	Rednex	Cotton Eye Joe	Ed Sheeran	Don't
94	The Carpenters	Only Yesterday	Boyz II Men	Thank You	Ella Henderson	Ghost
95	Dwight Twilley Band	I'm on Fire	The Pretenders	I'll Stand by You	Alessia Cara	Here
96	Ringo Starr	Only You (And You Alone)	N II U	I Miss You	Mr. Probz	Waves
97	Amazing Rhythm Aces	Third Rate Romance	Da Brat	Give it 2 You	Ne-Yo	She Knows
98	Bachman–Turner Overdrive	You Ain't Seen Nothing Yet"	Brandy	Best Friend	One Direction	Night Changes
99	Frankie Valli	Swearin' to God	Soul Asylum	Misery	Drake	Back To Back
100	Disco-Tex and the Sex-O-Lettes	Get Dancin	Van Halen	Can't Stop Lovin' You	Calvin Harris	How Deep Is Your Love

Table 1: Song ranks for years 1975, 1995 and 2015

3.2 Crawler

A Python function was created to browse and obtain information from AZLyrics website [3]. Our tool integrated two Python packages, BeautifulSoup along with

[2]The full Python code will be accessible on Github.
[3]`https://www.azlyrics.com/`

NLTK. The Beautifulsoup package parsed the source code from a target webpage, extracted the lyrics and stored the plain text into one variable. NLTK was then used to tokenize the text. Our tokenized text later returns a list of tokens without punctuation by using regular expressions. When executing this function, it takes arguments from command line.

```
# python script.py --singer=<singer's name> --song=<song name>
```

```
from bs4 import BeautifulSoup
from nltk import *, RegexpTokenizer
from urllib import request
from collections import Counter
import re,sys, getopt

def crawler(argv):
        singer=''
        song=''
        try:
                opts, args=getopt.getopt(argv,'x',['singer=','song='])
        except getopt.GetoptError:
                print('python script.py --singer=<singer name> --song=<song
                 ↪  name>')
                sys.exit(2)
        for opt, arg in opts:
                if opt in ("--singer"):
                        singer=arg
                elif opt in ("--song"):
                        song=arg
        singer=singer.lower().replace(" ", "")
        song=song.lower().replace(" ","")
        url='https://www.azlyrics.com/lyrics/{0}/{1}.html'.format(singer,song)
        response=request.urlopen(url)
        soup=BeautifulSoup(response, 'html.parser')
        raw=soup.find_all('div',attrs={'class':None})[1]
        raw=raw.get_text()
        raw=re.sub("[\(\[].*?[\)\]]", "", raw)
        raw=re.sub(r'[!?.]','',raw)
        tokenizer = RegexpTokenizer('\s+', gaps=True)
        tokens=tokenizer.tokenize(raw)
        words=[w.lower() for w in tokens]
        return(words)
```

3.3 Features

Previous studies have proposed several measurement for authorship identification, sentiment analysis and other types of analysis. Kešelj et al. [2003] applied byte-based profiling for automated authorship attribution and achieved the same or higher accuracy with token-based analysis. Our goal for this project is to compare the differences for years and popularity. Thus we employed the byte-based and token-based measurements.

3.3.1 Length

We measured the length of a song text in two levels, character based and token based. Since the database we use is open for public submission, the usage of punctuation is inconsistent between participants who uploaded the lyrics. In case the punctuation affects the calculation, we provided two byte-based measurements to include or exclude punctuation at the end of sentences.

```
from nltk import *
from nltk.tokenize import RegexpTokenizer,sent_tokenize
import re

def DigitsAnalysis(raw,function):
        if function=='Original':
                rawLen=len(raw) # original length.
                return(rawLen)
        elif function=='Includespace':
                rawPR=re.sub(r'[!?.,]','',raw) # remove punctuation.
                rawPRLen=len(rawPR) # length without punctuation, but include
                 ↪  white space.
                return(rawPRLen)
        elif function=='Tokenize':
                tokenizer = RegexpTokenizer('\s+', gaps=True)
                rawPR=re.sub(r'[!?.,]','',raw)
                tokens=tokenizer.tokenize(rawPR)
                words=[w.lower() for w in tokens]
                words=list(filter(None, words))
                # count the average word length.
                Average=sum([len(w) for w in words])/len(words) # understand
                 ↪  the average length of words
                Max=max([len(w) for w in words])
                Min=min([len(w) for w in words])
                return((Max, Average, Min))
```

3.3.2 Lexical Richness

The simple type-token ratio as well as noun and verb variations were applied to represent the lexical variation. There were different functions for calculating the type-token ratio. The TTR function was used to calculate the type-token ratio from monograms, bigrams and trigrams. The POSTTR function calculates the noun and verb variations. The following example shows how we can use these two functions.

```
from nltk import *

def TTR(tokenlist):
        Type=len([t for t,v in Counter(tokenlist).items()])
        TTRs=Type/len(tokenlist)
        return(TTRs)

def POSTTR(tokenlist,taglist):
        posTTR=len(taglist)/len(tokenlist)
        return(posTTR)

if __name__ == "__main__":
        data=crawler(sys.argv[1:])
        MonogramTTR=TTR(data)
        BigramTTR=TTR(Ngrams(data,2))
        TrigramTTR=TTR(Ngrams(data,3))
        print(MonogramTTR)
        print(BigramTTR)
        print(TrigramTTR)

        for c in ['NN','VBP','VBD']:
                ctags=[items[0] for items in tags if items[1]==c]
                uctags=[t for t,v in Counter(ctags).items()]
                cTTR=POSTTR(data,uctags)
                print('{0} TTR:{1}'.format(c,cTTR))
```

3.3.3 Token categorization

In the token-based features, words were categorized into two categories which are content words and function words. To obtain all the function words from the lyrics, we applied NLTK stop words corpus (Bird, 2006) and CoreNLP stop words corpus (Manning et al., 2014) to detect and extract the information. The

following functions calculated the proportion and variation of function words in a song text as well as the list of words.

```
import pickle
from nltk import *
import sys, re
from nltk.corpus import stopwords

def NLTKfunctionwords(tokenlist,value):
        stop=stopwords.words('english')
        # we need to create a new tokenlist.
        Newlist=[w.split("'") for w in tokenlist]
        Newlist=[item for sublist in Newlist for item in sublist]
        getfunct=[w for w in Newlist if w in stop]
        if value=='list':
                return(getfunct)
        if value=='proportion':
                result=len(getfunct)/len(Newlist)
                return(result)
        if value=='variation':
                result=len(set(getfunct))/len(getfunct)
                return(result)

def CNLPfunctionwords(tokenlist,value):
        with open('stopwords_list.txt','r') as f:
                stop=f.readlines()
                stops=[re.sub('\n','',w) for w in stop]
                f.close()
        getfunct=[w for w in tokenlist if w in stops]
        if value=='list':
                return(getfunct)
        if value=='proportion':
                result=len(getfunct)/len(tokenlist)
                return(result)
        if value=='variation':
                result=len(set(getfunct))/len(getfunct)
                return(result)
```

After we removed the function words from the token list, the remaining words were called content words. Within the range of content words, there are various categories such as verbs, nouns and adjectives. The part-of-speech tags were applied to annotate the tokens and only then we calculated the noun and verb proportion and variation.

The noun variation as well as two different verb variation measurements are defined as following:

$$\text{Noun Variation} = \frac{\text{Number of unique noun words}}{\text{total lexical words}} \tag{1}$$

$$\text{Verb Variation 1} = \frac{\text{Number of unique verb words}}{\text{total lexical words}} \tag{2}$$

$$\text{Noun Variation 2} = \frac{\text{Number of unique noun words}}{\text{total verb words}} \tag{3}$$

```
import pickle
from nltk import *
import sys, re
from nltk.corpus import stopwords

def abbreviation(tokenlist,returnvalue):
        HaveAbb=[w for w in tokenlist if "'" in w]
        Proportion=len(HaveAbb)/len(tokenlist)
        dout=[w.split("'")[0] for w in tokenlist]
        if returnvalue=='Abblist':
                return(HaveAbb)
        elif returnvalue=='Proportion':
                return(Proportion)
        elif returnvalue=='Newoutput':
                return(dout)
# remove all the stop words so we can have the lexical words for analysis
def removestop(tokenlist):
        stop=stopwords.words('english')
        dout=[w for w in tokenlist if w not in stop]
        return(dout)

# you can select NN, VBD or VBP etc, up to you. But we don't do more than noun
 ↪  or verb.
def gettag(taglist, tagname):
        tags=[items[0] for items in taglist if items[1]==tagname]
        return(tags)

def tFrequencyDist(taglist,tagname):
        tags=[items[0] for items in taglist if items[1]==tagname]
        FD=FreqDist(tags)
        return(FD.most_common(30))
```

3.3.4 Frequency Distribution of N-grams

In order to present a useful list of common words, we selected the top 30 most common nouns and verbs. With NLTK package, the part-of-speech tagger marks the tokens with their respective grammatical categories. Next, the tokens along with their frequencies will be reported. We then defined a Python function to select the top 30 most common words from the annotated token list. The following code is an example of the function which was defined.

```
from nltk import *

def FrequencyDist(taglist,tagname):
        tags=[items[0] for items in taglist if items[1]==tagname]
        FD=FreqDist(tags)
        return(FD.most_common(30))

if __name__ == "__main__":
        data=crawler(sys.argv[1:])
        tags=pos_tag(data)
        print(FrequencyDist(tags,'NN'))
        print(FrequencyDist(tags, 'VBP'))
        print(FrequencyDist(tags,'VBD'))
```

We also defined an n-gram function to combine the frequency distribution functions, in order to present the top 30 common bigrams and trigrams per song.

```
from nltk import *

def Ngrams(tokenlist,N):
        ngrams=list(ngrams(tokenlist,N))
        return(ngrams)

if __name__ == "__main__":
        data=crawler(sys.argv[1:])
        dataBigram=Ngrams(data,2)
        print(dataBigram)
```

3.4 Statistical analysis

We stored the data in a wide table with all the features, including the total characters in text, type-token ratio and the proportion of content words.
Exploratory data analysis, such as boxplot, scatter-plot and principle component analysis, were applied to examine the data. Scatter-plot was used to better see the differences between the high ranked and low ranked songs. Principle component analysis (PCA) was applied in order to better see the correlations between features of different years. To better read the PCA plot, we deleted one of those features which were highly correlated (e.g. since type token ratio highly correlates with the variety of function words, only one was used as a component). For extracting the scatter-plots the ggplot package was used. We also used heatmap to cluster the lyrics to have a clear view of similarities between each song. Heatmap is generated by R gplot package. The algorithms for calculating the similarity were maximum distance, and Ward clustering method.
At the end we applied one-way ANOVA and t-test to further examine those differences which were observed by visualization methods. All the analysis and figures were calculated and generated by R.

4 Results

4.1 Exploratory analysis : Box-plot, Scatter-plot, PCA and Heatmap

The following box-plots give the range of each feature. In figure 1, we divide the song texts by the factor of year. It shows that there are some features, which show a clear increasing or decreasing pattern among three time periods. For example, the total amount of characters (first row, first cell), total tokens (second row, middle cell) and noun variation (third row, middle cell). In figure 2, we broke down the songs into smaller groups by year and popularity. In the second box-plot, a decreasing trend starts to appear in the function words variation, which was not showing a clear pattern in the yearly box-plot.

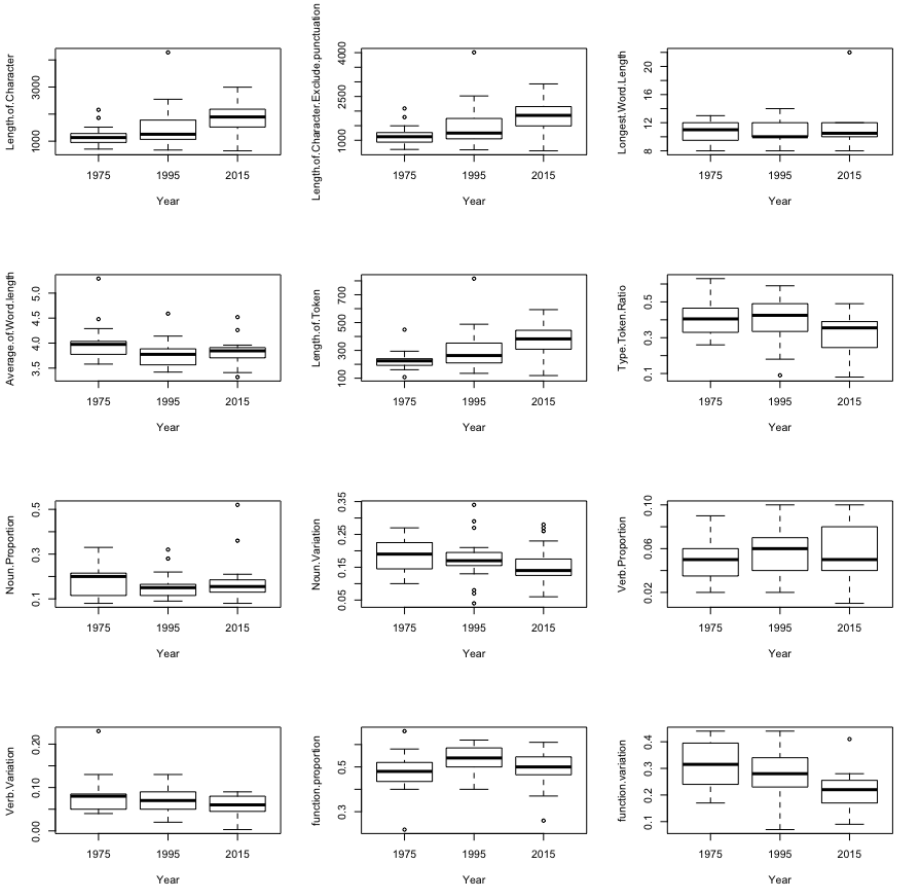

Figure 1: Feature summary by year

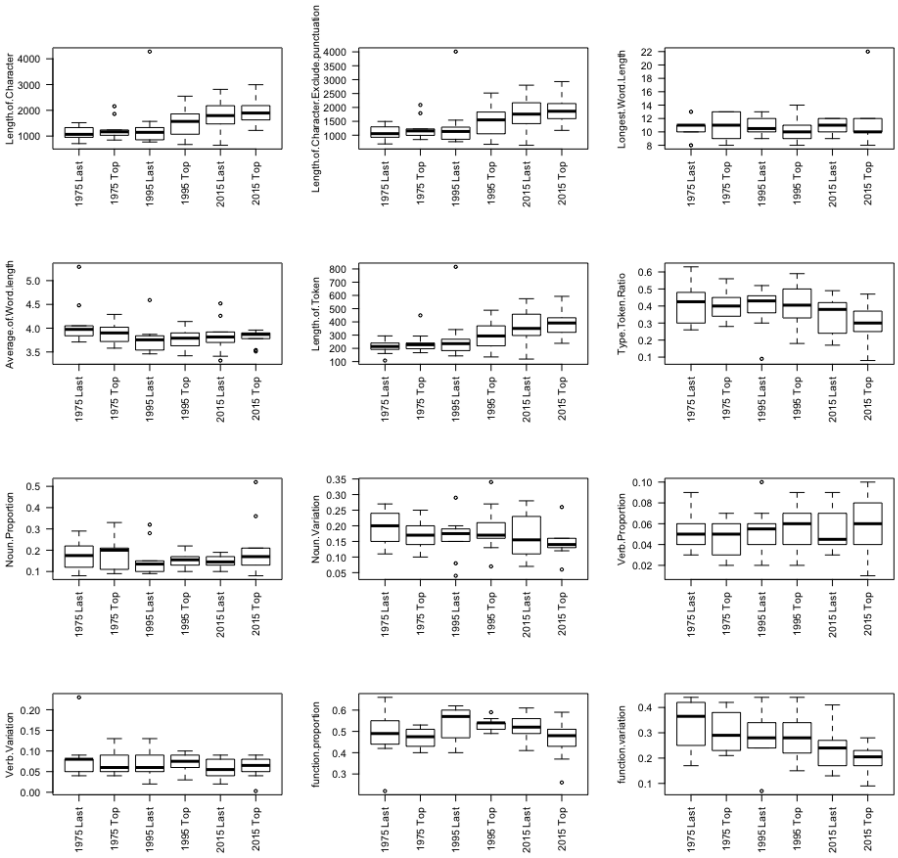

Figure 2: Boxplot summary by year and popularity

4.2 Scatter-plots

Coloured scatter-plot is a strong medium to show the differences in data-set. As we can see in the plots, the total distribution of data between high-ranked songs and low-ranked songs is not that distinct. The difference between plots is, however, in the years. As an example the left-down plot shows how number of tokens has started to increase along with the years. On the other hand the variety of function words decreased as the years increased. The proportion of function words however, increased at nearly the same rate.

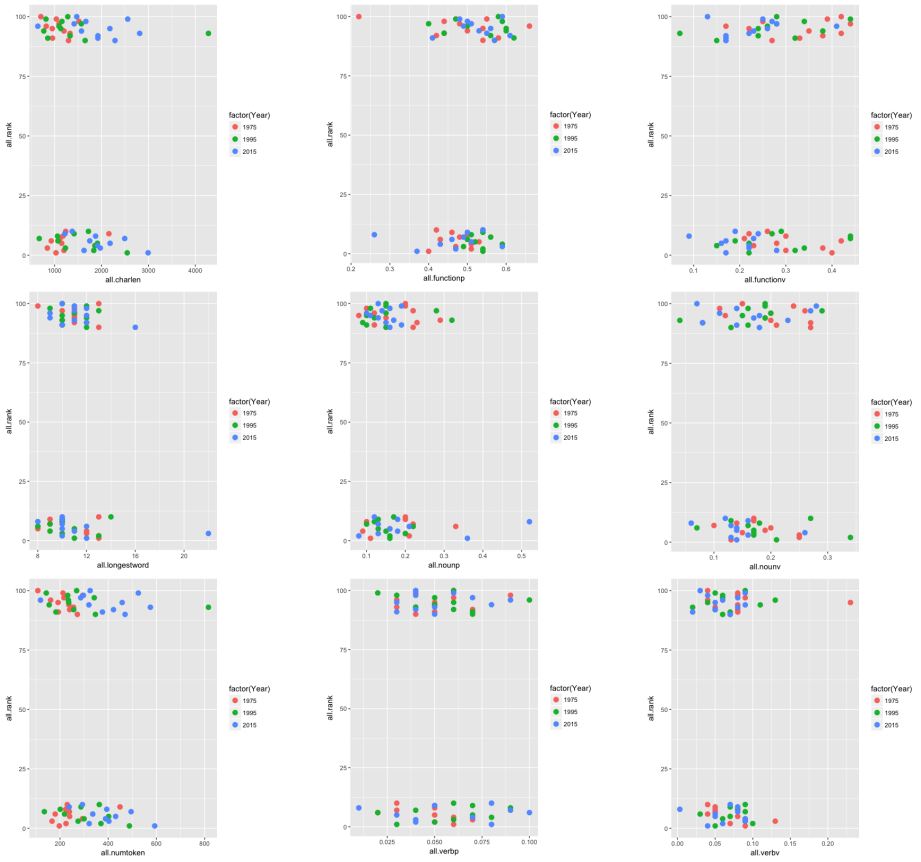

Figure 3: Scatter-plots with features on the x-axis and rank on the y-axis, with the year as factor

We also applied scatter-plots on type-token ratio, number of tokens and function word variation (Figure 4). We can see how with the increase of type-token-ratio the function word variation also increases. The figure on the left shows the increasing number of tokens mostly belong to 2015-songs; however, the number of types has decreased or remained constant which the final result would be a lower type-token ratio. Both figures clearly show how 1975-songs are lexically richer compared to 2015-songs.

4.3 Principle Component Analysis

PCA was used in order to give a better overview regarding the correlation between features.

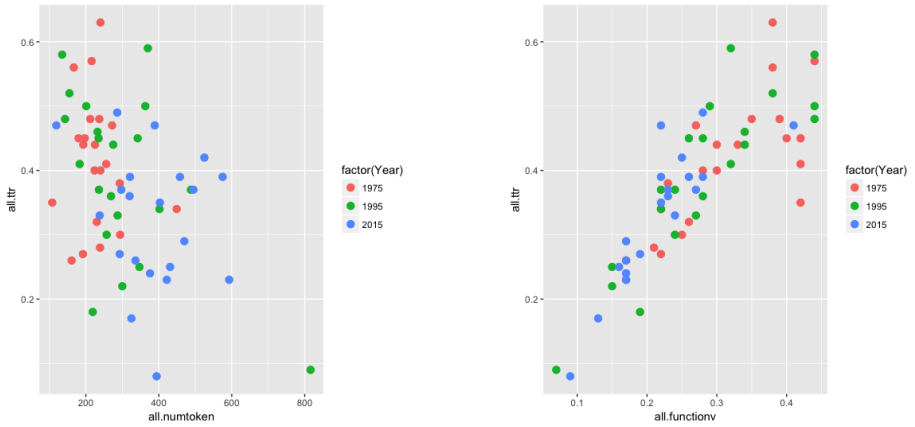

Figure 4: type-token-ratio on y-axis and number of tokens and function words' variation on the x-axis, along with years as the factor

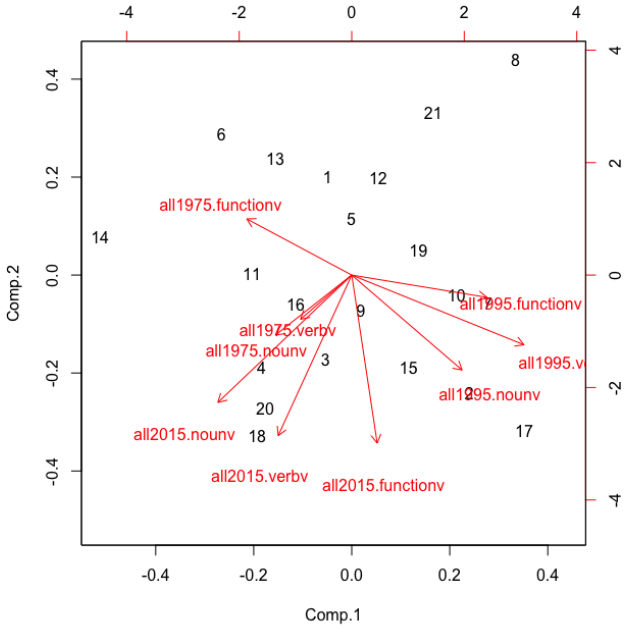

Figure 5: Principle component analysis plot on verb, noun and function word variation

As we can see in the plot, there is closer correlation between noun and verb variation compared to function word variation. It is also shown that there is a

positive correlation between the noun variation of 1975 and 2005.
Interestingly, as we can see in figure 6, type token ratio has a higher correlation with function words rather than noun or verb variations.

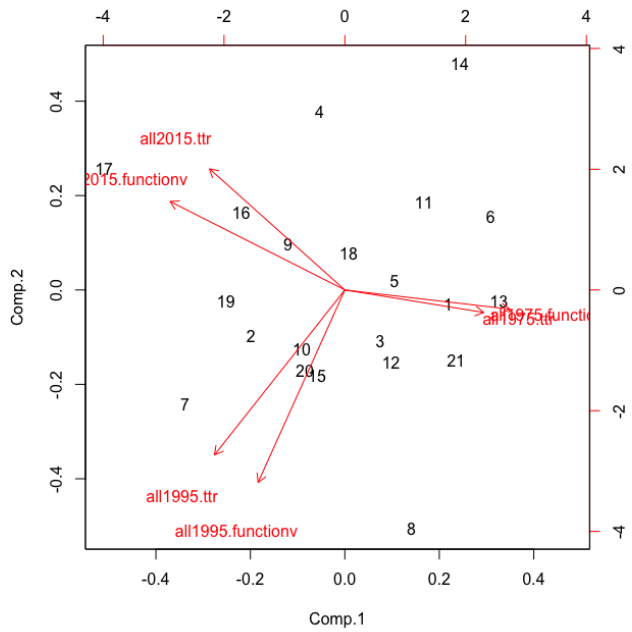

Figure 6: Principle component analysis plot on function words variation along with type token ratio

4.4 Heatmap

This figure gives the information not only on the most informative features but also the similarity or dissimilarity among songs (Figure. 7). The songs are clustered into 4 groups and there are three groups of features. In the Heatmap, the white and dark red color indicate the large deviation from the rest of values in the specific feature. As for the sidebar color, the darker colors indicate popular songs and the lighter colors mark the least favorite songs on the Billboard Hot 100 ranking. Blue, green and brown represent the year 1975, 1995 and 2015 correspondingly.

From the right-hand side to the left-hand side, there are three feature groups, the measurement of total characters, total characters (removing punctuation) as well as total tokens are the most informative features because there are clear

differences between orange colors. There are no clear patterns to be seen in the rest of the features. In addition, the branch length indicates the differences between feature groups. The total characters feature is vastly different from the rest of the measurements.

The side bar color presents an interesting trend. For top to bottom, the first cluster shows that the songs that were popular in 1975 and 1995 are often categorized in the same subgroup with the least favorite songs in 2015. However, there are some top 10 and bottom 10 songs in 2015 that are located in the first and second clusters as well. In the third and fourth clusters, the majority of samples are songs in 1975 and 1995. The branch lengths are longer between first and second song cluster, whereas the difference between the third and fourth song subgroups is not as huge as the first two groups. Therefore, we can roughly infer that the evolution of song text was slow between the year 1975 and the year 1995. However, the changes sped up from 1995 onward.

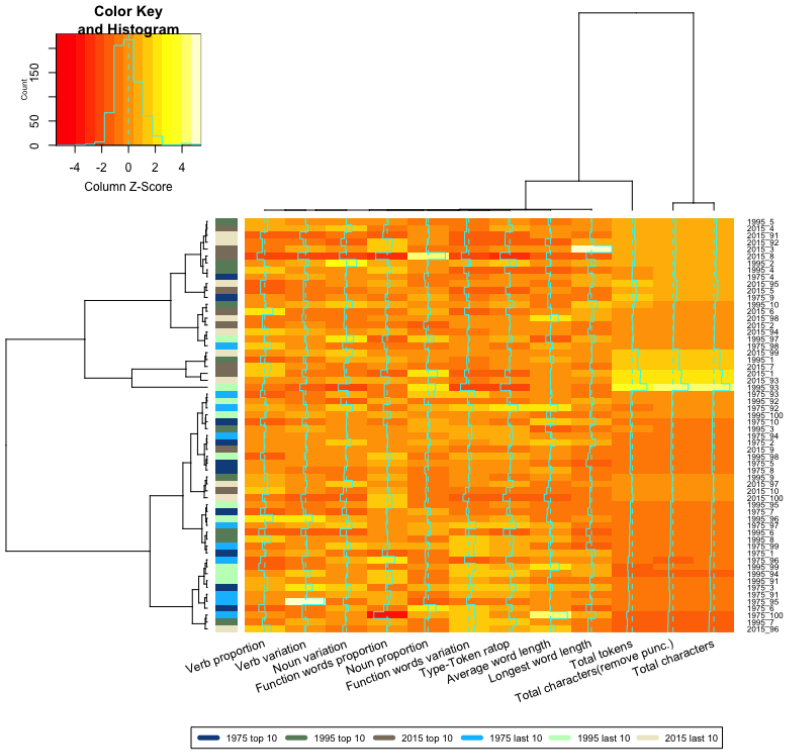

Figure 7: Clustering

4.5 Feature Analysis

The following sections will give detailed information on the maximum, mean and minimum of our small corpus. Afterwards, statistical analysis explains the significance of the so called changes among groups.

4.6 Length

We have calculated the length of each song and got an average of 1167 characters for 1975-songs, 1489 for 1995-songs and 1899 for 2015 songs. We observe less difference between the minimum and maximum value for 1975-songs than for 2015-songs. As we can also observe in Table 2, there seems to be more consistency between the length of the songs of 1975.

	Length of Characters			Length of longest words		
	1975	1995	2015	1975	1995	2015
Min	708	673	643	8	8	8
Mean	1167	1489	1899	10.76	10.76	11.13
Max	2158	4280	2994	13	14	22

Table 2: Characters and longest words length

4.7 Function Words

As can be seen in Table 3, the mean of function word variation decreased considerably from the 1975-songs to the 2015-songs; however, the proportion has not increased at the same rate.

	Proportion of function words			Variation of function words		
	1975	1995	2015	1975	1995	2015
Min	0.22	0.4	0.26	0.17	0.07	0.09
Mean	0.47	0.53	0.49	0.31	0.28	0.21
Max	0.66	0.62	0.61	0.44	0.44	0.41

Table 3: Proportion and variation of content words

4.8 Lexical Richness

	Type Toke Ratio		
	1975	1995	2015
Min	0.26	0.09	0.08
Mean	0.41	0.39	0.32
Max	0.63	0.59	0.49

Table 4: Type Token Ration per Year

In order to be able to spot most changes over the years, noun and verb proportions and variations have been retrieved. The results are presented in the following table:

	Proportion of Verbs			Variation of Verbs		
	1975	1995	2015	1975	1995	2015
Min	0.02	0.02	0.01	0.04	0.02	0.003
Mean	0.05	0.05	0.05	0.07	0.07	0.06
Max	0.09	0.1	0.1	0.23	0.13	0.09
	Proportion of Nouns			Variation of Nouns		
Min	0.08	0.09	0.08	0.1	0.04	0.06
Mean	0.17	0.15	0.17	0.18	0.17	0.15
Max	0.33	0.32	0.52	0.27	0.34	0.28

Table 5: Characters and Longest words length

4.9 One-way ANOVA and student t-test

The One-way ANOVA examines the changes for the three periods and the popularity. In the three periods in table 6, we found that the results are consistent with the previous analysis. The length measurements and function variation achieved high statistical significance. Whereas the type-token ratio and the function words proportion were not significant; however, the p-value landed at the border. The t-test on popularity in table 7 tells a different story. Between the top 10 and least favorite 10 songs, there were no significant changes throughout the years. The type-token ratio, noun variation, verb variation and function word variation all showed considerable differences between the favorite songs and non-favorite songs.

Table 6: One-way ANOVA among Years

	F score	p-value
Total characters	6.766	0.002
Total characters (remove punc.)	7.161	0.002
Longest word length	0.277	0.759
Average word length	2.944	0.061
Total tokens	8.473	0.001
Type-Token ratio	3.146	0.051
Noun proportion	0.501	0.609
Noun variation	1.194	0.311
Verb proportion	0.483	0.62
Verb variation	1.333	0.272
Function words proportion	3.042	0.056
Function words variation	7.471	0.001

Table 7: Popularity comparison (Year)

	1975		1995		2015	
Features	t value	p-value	t value	p-value	t value	p-value
Total characters	0.215	0.831	0.025	0.98	0.053	0.958
Total characters (remove punc.)	0.195	0.846	0.059	0.953	0.036	0.971
Longest word length	-4.593	<0.001	-4.636	<0.001	-4.554	<0.001
Average word length	-4.292	<0.001	-4.268	<0.001	-4.272	<0.001
Total tokens	-0.836	0.41	-0.686	0.497	-0.592	0.557
Type-Token ratio	-4.113	0.001	-4.111	0.001	-4.112	0.001
Noun proportion	-4.101	0.001	-4.101	0.001	-4.098	0.001
Noun variation	-4.103	0.001	-4.1	0.001	-4.102	0.001
Verb proportion	-4.096	0.001	-4.096	0.001	-4.096	0.001
Verb variation	-4.098	0.001	-4.097	0.001	-4.096	0.001
Function words proportion	-4.116	0.001	-4.118	0.001	-4.12	0.001
Function words variation	-4.11	0.001	-4.106	0.001	-4.106	0.001

4.10 Common N-grams

The common unigrams, bigrams and trigrams and the most frequently used words or phrases during these three years are shown in the table below.

4.10.1 Unigram

In table 8, the most frequent words are similar among the different years. However, there are song-tokens that do not carry meaning in the year 2015.
The NLTK package and CoreNLP package have some differences in tokenization

of the contractions. (e.g. in the case of *"I'm"*, NLTK tokenizes it as two tokens of *"I"* and *"am"*, however, CoreNLP, keeps it as one. Although we calculate the n-gram's count and normally would consider *I'm* as two tokens, since we are dealing with rhythmic poems, we preferred coreNLP's performance more than that of NLTK.

Table 8: Yearly top 30 unigram

	Unigram		
	Year 1975	Year 1995	Year 2015
1	('you', 208)	('you', 332)	('you', 253)
2	('i', 157)	('i', 241)	('i', 236)
3	('the', 152)	('the', 171)	('me', 197)
4	('and', 113)	('hey', 163)	('the', 190)
5	('a', 107)	('and', 142)	('it', 153)
6	('to', 90)	('to', 123)	('and', 151)
7	('my', 85)	('me', 94)	('my', 131)
8	('me', 84)	('my', 93)	('up', 121)
9	('in', 59)	('in', 82)	('a', 109)
10	('on', 50)	('it', 82)	("i'm", 106)
11	('of', 49)	('a', 81)	('oh', 100)
12	("i'm", 47)	('where', 60)	('your', 100)
13	("ain't", 45)	('did', 60)	('watch', 99)
14	('fame', 44)	('on', 58)	('to', 94)
15	('your', 43)	('come', 58)	('doh', 83)
16	('love', 43)	('know', 57)	('on', 78)
17	('be', 38)	('been', 55)	('love', 77)
18	('it', 37)	('so', 54)	('is', 73)
19	('so', 35)	('that', 53)	("don't", 70)
20	('got', 32)	('for', 52)	('she', 67)
21	('what', 30)	('oh', 51)	('all', 66)
22	('only', 27)	('of', 49)	('that', 65)
23	('is', 27)	("i'm", 49)	('for', 64)
24	('with', 27)	('is', 48)	('we', 63)
25	('one', 27)	('from', 47)	('ooh', 62)
26	('can', 25)	('all', 46)	('just', 58)
27	('god', 24)	('but', 45)	('how', 55)
28	('just', 24)	('go', 41)	('with', 54)
29	('get', 23)	('your', 41)	('in', 52)
30	('like', 23)	('love', 39)	('be', 50)

4.10.2 Bigrams

In the following Bigram table, we see that meaningless words such as *doh* or *ooh* were not common in the songs from 1995 and 1975. However, these words became a trend in year 2015.

	Bigrams		
	Year 1975	Year 1995	Year 2015
1	(('fame', 'fame'), 31)	(('hey', 'hey'), 149)	(('watch', 'me'), 77)
2	(('you', "ain't"), 29)	(('did', 'you'), 58)	(('doh', 'doh'), 76)
3	(('jive', "talkin'"), 17)	(('where', 'did'), 55)	(('oh', 'oh'), 41)
4	(('only', 'you'), 17)	(('you', 'come'), 42)	(('up', 'up'), 36)
5	(("ain't", 'you'), 15)	(('cotton-eye', 'joe'), 38)	(('your', 'love'), 34)
6	(('adored', 'you'), 14)	(('come', 'from'), 36)	(('bop', 'bop'), 32)
7	(('thank', 'god'), 13)	(('in', 'the'), 31)	(('now', 'watch'), 32)
8	(('bad', 'blood'), 13)	(('oh', 'i'), 29)	(('how', 'deep'), 30)
9	(('on', 'the'), 13)	(('to', 'me'), 20)	(('deep', 'is'), 30)
10	(('i', 'got'), 13)	(('and', 'i'), 20)	(('is', 'your'), 30)
11	(('one', 'of'), 13)	(('you', 'go'), 20)	(('uptown', 'funk'), 29)
12	(('you', 'can'), 13)	(('from', 'where'), 19)	(('you', 'up'), 23)
13	(('country', 'boy'), 13)	(('you', 'and'), 19)	(('funk', 'you'), 23)
14	(('i', 'love'), 13)	(('joe', "i'd"), 18)	(('i', "don't"), 23)
15	(('in', 'the'), 13)	(('go', 'where'), 18)	(('me', 'ooh'), 21)
16	(('and', 'the'), 12)	(('for', 'cotton-eye'), 18)	(('ooh', 'ooh'), 21)
17	(('of', 'these'), 12)	(('ago', 'where'), 18)	(('and', 'i'), 20)
18	(('my', 'eyes'), 12)	(('long', 'time'), 18)	(('up', 'uptown'), 20)
19	(("ain't", 'seen'), 12)	(('it', "hadn't"), 18)	(('on', 'for'), 20)
20	(('a', 'country'), 12)	(("i'd", 'been'), 18)	(('me', 'watch'), 18)
21	(('get', 'dancing'), 12)	(("hadn't", 'been'), 18)	(("don't", 'believe'), 18)
22	(('to', 'god'), 11)	(('married', 'long'), 18)	(('me', 'just'), 18)
23	(("i'm", 'a'), 11)	(('if', 'it'), 18)	(('believe', 'me'), 18)
24	(('what', 'you'), 11)	(('from', 'cotton-eye'), 18)	(('all', 'night'), 18)
25	(('god', "i'm"), 11)	(('been', 'for'), 18)	(('night', 'long'), 18)
26	(('like', 'a'), 11)	(('been', 'married'), 18)	(('ooh', 'watch'), 18)
27	(("swearin'", 'to'), 11)	(('time', 'ago'), 18)	(('up', 'all'), 18)
28	(('eyes', 'adored'), 10)	(('on', 'the'), 17)	(('just', 'watch'), 18)
29	(('how', 'i'), 9)	(('i', 'know'), 16)	(("i'm", 'gonna'), 17)
30	(('and', 'you'), 9)	(('in', 'my'), 15)	(('you', 'ooh'), 17)

Table 9: Yearly top 30 bigrams

4.10.3 Trigrams

In the table of popular Trigrams, we see that repetitive words were used often in 2015-songs. These words are either non-words or function words. The phenomenon of using repetitive words was not frequent in 1975 or 1995.

	Trigrams		
	Year 1975	Year 1995	Year 2015
1	(('fame', 'fame', 'fame'), 25)	(('hey', 'hey', 'hey'), 144)	(('doh', 'doh', 'doh'), 69)
2	(('you', "ain't", 'you'), 14)	(('where', 'did', 'you'), 54)	(('now', 'watch', 'me'), 32)
3	(("ain't", 'you', "ain't"), 14)	(('you', 'come', 'from'), 36)	(('is', 'your', 'love'), 30)
4	(('a', 'country', 'boy'), 12)	(('did', 'you', 'come'), 36)	(('deep', 'is', 'your'), 30)
5	(('one', 'of', 'these'), 12)	(('cotton-eye', 'joe', "i'd"), 18)	(('how', 'deep', 'is'), 30)
6	(("swearin'", 'to', 'god'), 11)	(("hadn't", 'been', 'for'), 18)	(('bop', 'bop', 'bop'), 28)
7	(('thank', 'god', "i'm"), 11)	(("i'd", 'been', 'married'), 18)	(('oh', 'oh', 'oh'), 25)
8	(("i'm", 'a', 'country'), 11)	(('been', 'for', 'cotton-eye'), 18)	(('funk', 'you', 'up'), 23)
9	(('god', "i'm", 'a'), 11)	(('time', 'ago', 'where'), 18)	(('uptown', 'funk', 'you'), 23)
10	(('eyes', 'adored', 'you'), 10)	(('been', 'married', 'long'), 18)	(('watch', 'me', 'ooh'), 21)
11	(('my', 'eyes', 'adored'), 9)	(('joe', "i'd", 'been'), 18)	(('up', 'uptown', 'funk'), 20)
12	(('of', 'these', 'nights'), 9)	(('from', 'where', 'did'), 18)	(('you', 'up', 'uptown'), 20)
13	(("ain't", 'got', 'no'), 8)	(('come', 'from', 'where'), 18)	(('all', 'night', 'long'), 18)
14	(('you', "ain't", 'got'), 8)	(('married', 'long', 'time'), 18)	(('watch', 'me', 'watch'), 18)
15	(('i', 'got', 'me'), 8)	(('it', "hadn't", 'been'), 18)	(('up', 'all', 'night'), 18)
16	(('only', 'you', 'only'), 8)	(('did', 'you', 'go'), 18)	(('me', 'watch', 'me'), 18)
17	(('you', 'only', 'you'), 8)	(('you', 'go', 'where'), 18)	(('believe', 'me', 'just'), 18)
18	(("i'm", 'on', 'fire'), 7)	(('long', 'time', 'ago'), 18)	(("don't", 'believe', 'me'), 18)
19	(('seen', "nothin'", 'yet'), 7)	(('if', 'it', "hadn't"), 18)	(('up', 'up', 'all'), 18)
20	(('to', 'see', 'what'), 7)	(('for', 'cotton-eye', 'joe'), 18)	(('up', 'up', 'up'), 18)
21	(("ain't", 'seen', "nothin'"), 7)	(('from', 'cotton-eye', 'joe'), 18)	(('ooh', 'watch', 'me'), 18)
22	(('you', 'can', 'make'), 6)	(('ago', 'where', 'did'), 18)	(('me', 'just', 'watch'), 18)
23	(('someone', 'saved', 'my'), 6)	(('come', 'from', 'cotton-eye'), 18)	(('on', 'for', 'tonight'), 16)
24	(('rate', 'romance', 'low'), 6)	(('go', 'where', 'did'), 18)	(('here', 'oh', 'oh'), 16)
25	(('i', 'will', 'i'), 6)	(('i', 'oh', 'i'), 14)	(('ooh', 'ooh', 'ooh'), 15)
26	(('di', 'di', 'dit'), 6)	(("i'll", 'stand', 'by'), 14)	(('me', 'ooh', 'watch'), 15)
27	(('you', 'just', "ain't"), 6)	(('stand', 'by', 'you'), 14)	(('duff', 'duff', 'duff'), 13)
28	(('country', 'boy', 'well'), 6)	(('oh', 'i', 'oh'), 14)	(('holding', 'on', 'for'), 13)
29	(('yeah', 'jive', "talkin'"), 6)	(('on', 'the', 'gray'), 13)	(('i', 'see', 'you'), 13)
30	(('get', 'dancing', 'dancing'), 6)	(('in', 'the', "gangsta's"), 12)	(('nae', 'nae', 'now'), 12)

Table 10: Yearly top 30 trigrams

5 Discussion

5.1 Features

In the result section, the section which covers the length of characters and words, we saw that there is greater consistency in the 1975-songs and less consistency in the 2015-songs. When it gets to the length of the longest words, we can define this consistency as having rhyme between words. It is indeed more difficult to find a rhyme for a 22 character-word, compared to a 13 character-word. We would conclude that rhyme did not get as much attention in 2015 songs as it did in 1975 or 1995. It is also interesting how the length of characters gradually increased over the years. Generally, with regard to length of characters and longest words, we can say that there is less difference between 1975-songs and 1995-songs.

Eder et al. [2016] discuss the reasons why function words are such important features in determining the author of a writing. Among these reasons, they mention that authors from the same time period use the same range of function words; thus, only knowing about the function words could lead to finding out which period the author comes from.
As we saw, the character lengths increased from the 1975-songs to the 2015-songs. We were interested in knowing what was added to the songs to make them longer, whether content words or function words.
Comparing the results of the function word variation and the function word proportion would suggest that the function word variation has noticeably decreased over the years. On the other hand, the proportion has not changed. This would suggest that the richness of language has decreased. The songs are getting longer. However, the word variation are not increasing.
With the help of statistical analysis and linguistic features, we would say that the high-ranked and low-ranked songs of the same year do not differ greatly linguistically; however, most of the linguistic changes belong to different time periods. Songs from 1975 have shorter and richer content. Moving along these years songs are getting lengthier and less meaningful. We saw an interesting phenomenon whereby function word variation correlates better with type token ratio compared to noun or verb variations.
Function words are getting more attention in authorship attribution tasks. Segarra et al. [2015] have shown that accuracy rises when information on relational data between function words is added to the frequency based methods.

5.2 Challenges

There were two major factors affecting the text processing; database and tokenization methods. Most of the text was obtained from AZlyrics.com website. Three songs were curated from Google Play Music web page. AZlyrics is open for the public to submit or correct texts. However, various participants have various documentation techniques. Some people prefer to write down every word, but there are rare occasions where people shortened the text by marking the repeating times of the chorus. Thus, the issues created some bias in measuring the length of characters.

The tokenization method greatly affects the analysis. In order to keep as much information as we can, we decided to use a regular expression tokenizer and split the plain text by white space. This method results in "you're" or "I'll"

be counted as one token instead of two. Taking into account the rhythm of a song, it is important to treat "you're" and "you are" differently. The amount of abbreviation also depends on authors' writing styles. When counting function words, NLTK tends to treat "you're" as two different tokens of "you" and "'re". The stop-words list which was provided by CoreNLP captures the difference. Our code returns both data results but we use the one from the CoreNLP.

6 Conclusion

The high-ranked (top ten) and low-ranked (bottom ten) songs from the same year, for three different years (based on Baby boomers' Generation, Generation X and Generation Y) were chosen. Songs were statistically analyzed for the necessary features and values instanced. From 1975 to 2015, the quality of lyrics decreased. Lyrics have gotten lengthier without getting richer. The variety of function word variation is also higher, which this feature has nicely correlated with the type-token ratio. Putting all the analyses together we would say that there was more meaning in the 1975-songs.

References

Steven Bird. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics, 2006.

Maciej Eder, Jan Rybicki, and Mike Kestemont. Stylometry with R: A Package for Computational Text Analysis. *The R Journal*, 8(1): 107–121, 2016. URL `https://journal.r-project.org/archive/2016/RJ-2016-007/index.html`.

Vlado Kešelj, Fuchun Peng, Nick Cercone, and Calvin Thomas. N-gram-based author profiles for authorship attribution. In *Proceedings of the conference pacific association for computational linguistics, PACLING*, volume 3, pages 255–264, 2003.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

S. Segarra, M. Eisen, and A. Ribeiro. Authorship attribution through function word adjacency networks. *IEEE Transactions on Signal Processing*, 63(20): 5464–5478, 2015.