Justin Bieber vs. The Beatles – text-mining w R

Kamil Stanuch, KoalaMetrics

2015-03-11 eRka #4, Kraków

VS.

VS.

600 mln
sprzedanych albumów

25 mln
sprzedanych albumów

VS.

„Hey Jude”
47 mln
wyświetleń na YouTube

„Baby”
1144 mln
wyświetleń na YouTube

Plan prezentacji

1. Przygotowanie danych do pracy
    A. Korpus
    B. Wstępne czyszczenie danych (funkcja tm_map)
    C. Budowa macierzy TermDocumentMatrix
    D. *Normalizacja: Term Frequency i Inverse Document Frequency (TF-IDF)
2. Analiza i wizualizacja
    A. Najczęstsze frazy (frequentTerms), korelacje (findAssocs)
    B. Wordcloud
    C. Analiza sentymentu
    D. Różnorodność leksykalna
3. Klasyfikator tekstów w oparciu o algorytm kNN

1.Przygotowanie danych

- Korpus
- Wstępne czyszczenie danych (funkcja tm_map)
- Budowa macierzy TermDocumentMatrix / DocumentTermMatrix
- *Normalizacja: Term Frequency i Inverse Document Frequency (TF-IDF)

1.A. Korpus

songTitle	songLyrics	songAuthor
All Around The World	[Bieber] You're beautiful, beautiful, you should know it (You're beautiful, beautiful, you sh	Justin Bieber
All Bad	[Verse 1:] It's another, if it ain't one thing Instigators, like puttin' fire on propane The wrong	Justin Bieber
All I Want For Christmas Is You	[Justin Bieber] I just can't wait [Mariah Carey] I don't want a lot for Christmas There is just	Justin Bieber
All I Want Is You	Sitting here, all alone Watching the snow fall Looking back at the days We threw them snow	Justin Bieber
All That Matters	Oh oh, just as sure as the stars in the sky I need you to show me the light Not just for the me	Justin Bieber
All Yours	You know it babe You know it babe I could open up your door like a gentleman If you open u	Justin Bieber
Alone	We were inseparable (inseparable) Everything I had to do I did it next to you (next to you) A	Justin Bieber
As Long As You Love Me	As long as you love me [3x] We're under pressure, Seven billion people in the world trying	Justin Bieber
Baby	Oh whoa [x3] You know you love me, I know you care Just shout whenever, and I'll be there	Justin Bieber
Baby	Oh wooaah [x3] You know you love me, I know you care Just shout whenever, and I'll be th	Justin Bieber
Backpack	You said, "I come in peace," so I took you home I gave you food and I gave you clothes I taug	Justin Bieber
Bad Day	No I didn't think you would let me down that easy Oh no girl And I didn't think it was over ur	Justin Bieber
Be Alright	Across the ocean, across the sea, Starting to forget the way you look at me now Over the mo	Justin Bieber
Be Alright (Acoustic Version)	Damn, I miss you Across the ocean, across the sea, Starting to forget the way you look at me	Justin Bieber
Beauty And A Beat	Yeah, Young Money, Nicki Minaj, Justin Show you off, tonight I wanna show you off (eh, eh,	Justin Bieber
Believe	(Believe) Believe, believe, believe I don't know how I got here I knew it wouldn't be easy B	Justin Bieber
Bigger	Love you The love, the love is bigger, The love, the love is bigger, The love, the love is bigge	Justin Bieber
Born To Be Somebody	There's a dream in my soul, A fire that's deep inside me. There's a me no one knows, Waitin	Justin Bieber
Boyfriend	[Verse 1:] If I was your boyfriend, I'd never let you go I can take you places you ain't never b	Justin Bieber
Boyfriend (Remix)	[Verse 1: 2 Chainz] (Boyfriend Remix) (2 chainz) 9 times out of 10 you a 10 If your schedule c	Justin Bieber
Broken	I guess they want a reaction I ain't gonna give it to em' Tryn' to get at me, yeah I ain't gonna f	Justin Bieber

1.A. Korpus

Ala ma kota, kot ma Alę.

1.A. Korpus

15/03/18

Ala ma kota, kot ma Alę.

	Ala	ma	kota	kot	Alę.
Document1	1	2	1	1	1

Document Term Matrix

	Document1
Ala	1
ma	2
kota	1
kot	1
Alę.	1

Term Document Matrix

1.A. Korpus


```
#0. LOAD LIBRARIES
library(tm)

#1. LOAD DATABASE
songsList <- read.csv2(„songsJustinBieber.csv", header=TRUE)

#2. BUILD A CORPUS
myCorpus <- Corpus(VectorSource(songsList$songLyrics)) # VectorSource
specifies that the source is character vectors.
```

1.B. Czyszczenie tekstu:
usuwanie znaków, spacji etc.

```
#4. CLEAR THE CORPUS
myCorpus <- tm_map(myCorpus, tolower) # convert to lower case
myCorpus <- tm_map(myCorpus, removePunctuation) # remove punctuation
myCorpus <- tm_map(myCorpus, removeNumbers) # remove numbers
myCorpus <- tm_map(myCorpus, stripWhitespace) # remove white space
myCorpus <- tm_map(myCorpus, removeWords, stopwords("english")) # remove
stopwords
```

1.B. Czyszczenie tekstu

Co gdy nie usuniemy popularnych fraz?

1.B. Czyszczenie tekstu:
usuwanie znaków, spacji etc.


```
#4. CLEAR THE CORPUS
myCorpus <- tm_map(myCorpus, tolower) # convert to lower case
myCorpus <- tm_map(myCorpus, removePunctuation) # remove punctuation
myCorpus <- tm_map(myCorpus, removeNumbers) # remove numbers
myCorpus <- tm_map(myCorpus, stripWhitespace) # remove white space
myCorpus <- tm_map(myCorpus, removeWords, stopwords("english")) # remove
stopwords

require(SnowballC)
myCorpus <- tm_map(myCorpus, stemDocument, language = "english") #
stemming
```

1.B. Czyszczenie tekstu:
stemming

Run
Running → run
Runner

1.C. TermDocumentMatrix

15/03/18

```
#5. BUILD DOCUMENT TERM MATRIX

myDTM <- DocumentTermMatrix(myCorpus, control = list(minWordLength =
3))
myDTM <- removeSparseTerms(myDTM, 0.99) #remove terms with more than
99% of zero occurance
```

KOALA

1.C. TermDocumentMatrix


```
#5. BUILD DOCUMENT TERM MATRIX

myDTM <- DocumentTermMatrix(myCorpus, control = list(minWordLength =
3))
myDTM <- removeSparseTerms(myDTM, 0.99) #remove terms with more than
99% of zero occurance
DTM <- as.data.frame(as.matrix(myDTM))
```

1.C. TermDocumentMatrix

n-gram (NGramTokenizer)


```
#5A. BUILD DOCUMENT TERM MATRIX – N-GRAM

require(RWeka)
TrigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min =
1, max = 3))
myDTM <- DocumentTermMatrix(myCorpus, control = list(tokenize =
TrigramTokenizer))

myDTM <- removeSparseTerms(myDTM, 0.99) #remove terms with more than
99% of zero occurance
DTM <- as.data.frame(as.matrix(myDTM))
```

15/03/18

1.C. TermDocumentMatrix

	aint	alon	alright	alway	anoth	around	ask	away	babi	back	bad	believ	best	better	bieber	big	blue	boy	break
1	0	0	0	0	0	24	0	0	2	7	0	0	0	0	2	0	0	0	0
2	10	0	0	0	1	0	0	0	0	8	18	0	1	0	0	0	0	0	0
3	0	0	0	0	0	0	2	0	10	0	0	0	0	0	6	0	0	0	0
4	0	2	0	0	0	0	0	2	3	2	0	1	0	0	0	0	0	0	0
5	4	0	0	0	0	0	0	0	0	1	0	2	0	1	0	0	0	0	0
6	0	0	0	0	0	2	0	4	0	0	0	0	0	0	0	0	0	0	0
7	0	10	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0
8	1	0	0	1	0	0	1	0	2	1	0	0	1	0	0	2	0	0	0
9	1	0	0	6	1	1	0	0	56	0	1	2	0	0	0	0	0	0	1
10	1	0	0	6	2	1	0	0	55	0	1	3	0	0	0	0	0	0	0
11	0	1	0	0	1	1	0	2	0	0	0	0	1	0	1	0	0	0	0
12	0	1	0	0	0	0	0	3	2	0	5	0	0	0	0	0	0	0	0
13	0	1	13	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
14	0	1	13	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	3	0	0	0	0	0	1	1	0	0	2

1.D. Normalizacja TF-IDF*

Term Frequency – Invert Document Frequency

	beauty	care	the	nail	hair
Document1	9	10	10	9	0
Document2	10	9	8	7	0
Document3	8	9	8	2	9
Document4	8	9	11	0	15
Document5	9	9	10	0	0
Document6	9	9	10	11	0
Document7	12	11	12	0	10
Document8	10	10	8	0	2
Document9	11	10	11	0	9
Document10	8	12	8	0	11

1.D. Normalizacja TF-IDF*

Term Frequency – Invert Document Frequency

	beauty	care	the	nail	hair
Document1	9	10	10	9	0
Document2	10	9	8	7	0
Document3	8	9	8	2	9
Document4	8	9	11	0	15
Document5	9	9	10	0	0
Document6	9	9	10	11	0
Document7	12	11	12	0	10
Document8	10	10	8	0	2
Document9	11	10	11	0	9
Document10	8	12	8	0	11

$$IDF(t) = 1 + log \frac{Total\ number\ of\ documents}{Number\ of\ documents\ containing\ t}$$

1.D. Normalizacja TF-IDF*

Term Frequency – Invert Document Frequency

	beauty	care	the	nail	hair
Document1	9	10	10	9	0
Document2	10	9	8	7	0
Document3	8	9	8	2	9
Document4	8	9	11	0	15
Document5	9	9	10	0	0
Document6	9	9	10	11	0
Document7	12	11	12	0	10
Document8	10	10	8	0	2
Document9	11	10	11	0	9
Document10	8	12	8	0	11
IDF	1	1	1	1,39794	1,221849

$$IDF(t) = 1 + log\frac{Total\ number\ of\ documents}{Number\ of\ documents\ containing\ t}$$

1.D. Normalizacja TF-IDF*

Term Frequency – Invert Document Frequency

	beauty	care	the	nail	hair
Document1	9	10	10	9	0
Document2	10	9	8	7	0
Document3	8	9	8	2	9
Document4	8	9	11	0	15
Document5	9	9	10	0	0
Document6	9	9	10	11	0
Document7	12	11	12	0	10
Document8	10	10	8	0	2
Document9	11	10	11	0	9
Document10	8	12	8	0	11
IDF	1	1	1	1,39794	1,221849

	beauty	care	the	nail	hair
Document1	9	8	12	12,58146	0
Document2	8	12	11	9,78558	0
Document3	9	10	9	2,79588	10,99664
Document4	9	11	10	0	18,32773
Document5	9	10	12	0	0
Document6	9	9	12	15,37734	0
Document7	11	9	10	0	12,21849
Document8	9	12	12	0	2,443697
Document9	12	9	11	0	10,99664
Document10	9	8	8	0	13,44034

$$TF(t,d) * IDF(t) = TFIDF(t,d)$$

1.D. Normalizacja TF-IDF*

Term Frequency – Invert Document Frequency

	beauty	care	the	nail	hair
Document1	9	10	10	9	0
Document2	10	9	8	7	0
Document3	8	9	8	2	9
Document4	8	9	11	0	15
Document5	9	9	10	0	0
Document6	9	9	10	11	0
Document7	12	11	12	0	10
Document8	10	10	8	0	2
Document9	11	10	11	0	9
Document10	8	12	8	0	11
IDF	1	1	1	1,39794	1,221849

	beauty	care	the	nail	hair
Document1	9	8	12	12,58146	0
Document2	8	12	11	9,78558	0
Document3	9	10	9	2,79588	10,99664
Document4	9	11	10	0	18,32773
Document5	9	10	12	0	0
Document6	9	9	12	15,37734	0
Document7	11	9	10	0	12,21849
Document8	9	12	12	0	2,443697
Document9	12	9	11	0	10,99664
Document10	9	8	8	0	13,44034

$$TF(t,d) * IDF(t) = TFIDF(t,d)$$

1.D. Normalizacja TF-IDF*

TermFrequency – Invert Document Frequency

	beuty	care	the	nail	hair
Document1	9	10	10	9	0
Document2	10	9	8	7	0
Document3	8	9	8	2	9
Document4	8	9	11	0	15
Document5	9	9	10	0	0
Document6	9	9	10	11	0
Document7	12	11	12	0	10
Document8	10	10	8	0	2
Document9	11	10	11	0	9
Document10	8	12	8	0	11
IDF	1	1	1	1,39794	1,221849

	beauty	care	the	nail	hair
Document1	9	8	12	12,58146	0
Document2	8	12	11	9,78558	0
Document3	9	10	9	2,79588	10,99664
Document4	9	11	10	0	18,32773
Document5	9	10	12	0	0
Document6	9	9	12	15,37734	0
Document7	11	9	10	0	12,21849
Document8	9	12	12	0	2,443697
Document9	12	9	11	0	10,99664
Document10	9	8	8	0	13,44034

$$TF(t,d) * IDF(t) = TFIDF(t,d)$$

1.D. Normalizacja TF-IDF*

TermFrequency – Invert Document Frequency

```
#5. BUILD DOCUMENT TERM MATRIX

myDTM <- DocumentTermMatrix(myCorpus, control = list(minWordLength = 3,
weighting= weightTfIdf))
DTM <- as.data.frame(as.matrix(removeSparseTerms(myDTM, 0.95)))
#remove terms with more than 95% of zero occurance
```

1.D. Normalizacja TFIDF*

15/03/18

400 observations of 293 variables

	about	after	again	aint	all	alone	alright	always	and	another
1	0.007597288	0.000000000	0.009758416	0.000000000	0.055712628	0.000000000	0.000000000	0.000000000	0.0007975114	0.00000000
2	0.000000000	0.000000000	0.000000000	0.145097666	0.070246357	0.000000000	0.000000000	0.000000000	0.0014454895	0.02145987
3	0.010703871	0.000000000	0.000000000	0.000000000	0.027302228	0.000000000	0.000000000	0.000000000	0.0056180967	0.00000000
4	0.000000000	0.000000000	0.013475907	0.000000000	0.043485840	0.026081779	0.000000000	0.000000000	0.0154185542	0.00000000
5	0.015107750	0.000000000	0.019405307	0.063677147	0.028901359	0.000000000	0.000000000	0.000000000	0.0031718169	0.00000000
6	0.022124320	0.000000000	0.000000000	0.000000000	0.063486248	0.000000000	0.000000000	0.000000000	0.0162572203	0.00000000
7	0.000000000	0.000000000	0.000000000	0.000000000	0.021476593	0.209318738	0.000000000	0.000000000	0.0053031970	0.00000000
8	0.000000000	0.000000000	0.000000000	0.008098474	0.002450454	0.000000000	0.000000000	0.008980428	0.0040339241	0.00000000
9	0.000000000	0.000000000	0.000000000	0.008416541	0.007640087	0.000000000	0.000000000	0.055998804	0.0109001259	0.01244802
10	0.000000000	0.000000000	0.000000000	0.008493522	0.012849943	0.000000000	0.000000000	0.056510988	0.0118459624	0.02512375
11	0.000000000	0.013136560	0.000000000	0.000000000	0.000000000	0.009988767	0.000000000	0.000000000	0.0033742733	0.01252369
12	0.000000000	0.000000000	0.000000000	0.000000000	0.007024636	0.027385868	0.000000000	0.000000000	0.0069383494	0.00000000
13	0.000000000	0.000000000	0.000000000	0.000000000	0.004606319	0.017957946	0.288336710	0.016881242	0.0121326328	0.00000000
14	0.000000000	0.000000000	0.000000000	0.000000000	0.004483810	0.017480341	0.280668181	0.016432273	0.0118099564	0.00000000
15	0.053519356	0.000000000	0.000000000	0.000000000	0.034127785	0.000000000	0.000000000	0.000000000	0.0044944773	0.00000000

Analiza i wizualizacja

- Jakich słów najczęściej używa Justin Bieber i The Beatles?
- Jaka jest dziewczyna u Beatlesów i u Justina Biebera?
- Jak zbudować chmurę tagów?
- Który artysta ma większy zasób słów, a który jest bardziej monotematyczny?

2.A. Podstawowe statystyki:

Najpopularniejsze frazy (findFreqTerms)

```
#Find the most frequent terms

findFreqTerms(myDTM, lowfreq =
200)
```

- findFreqTerms(myDTM, lowfreq = 200)

- "baby"
- "can"
- "cause"
- "dont"
- "girl"
- "just"
- "know"
- "like"
- "love"
- "one"
- "wanna"
- "yeah" [Bieber]

- findFreqTerms(myDTM, lowfreq = 200)

- "baby"
- "can"
- "dont"
- "got"
- "know"
- "like"
- "love"
- "now"
- "say"
- "see"
- "well"
- "yeah"
- "youre„ [Beatles]

2.A. Podstawowe statystyki:

Frazy skorelowane z frazą „girl” (findAssocs)

15/03/18

```
#Find terms associated with x

findAssocs(myDTM, terms = „girl",
corlimit = 0.35)
```

```
findAssocs(myDTM,
terms = „girl”,
corlimit = 0.35)
swag 0.59
near 0.58
town 0.49
read 0.47
hands 0.46
one 0.45
yeah 0.42
spent 0.40
less 0.38
inside 0.37
pretty 0.37
shes 0.37
faces 0.36
name 0.36
lonely 0.35
saw 0.35 [Bieber]
```

```
findAssocs(myDTM,
terms = „girl”,
corlimit = 0.35)
catch 0.40
another 0.38
sand 0.38 [Beatles]
```

2.B. Wordcloud

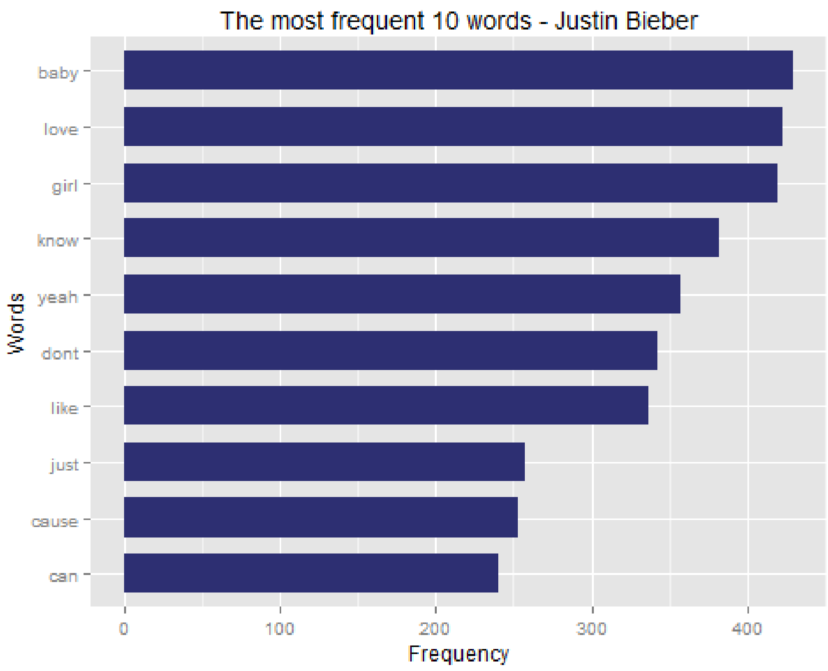
Justin Bieber vs. The Beatles

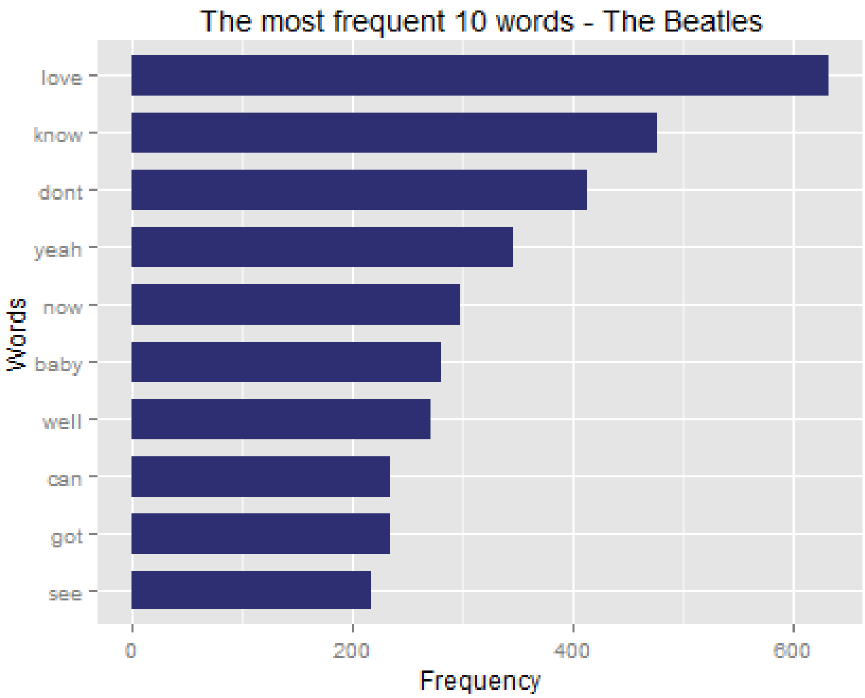
2.B. Wordcloud

Justin Bieber vs. The Beatles


```
#Worcloud - Justin Bieber
require(wordcloud)
library(RColorBrewer)
wordsFreqBieber <- sort(colSums(DTM),decreasing=TRUE)

dfFreqBieber <- data.frame(word =
names(wordsFreqBieber),freq=wordsFreqBieber)

pal <- brewer.pal(9, "Purples") #color palette

png(filename = "D:/wordCloud-JustinBieber.png",
width=1024,height=1024)

wordcloud(dfFreqBieber$word,dfFreqBieber$freq,
scale=c(10,.3),min.freq=2,max.words=150, random.order=F,
use.r.layout=T, rot.per=0.05, colors=pal, vfont=c("sans
serif","plain"))
dev.off()
```

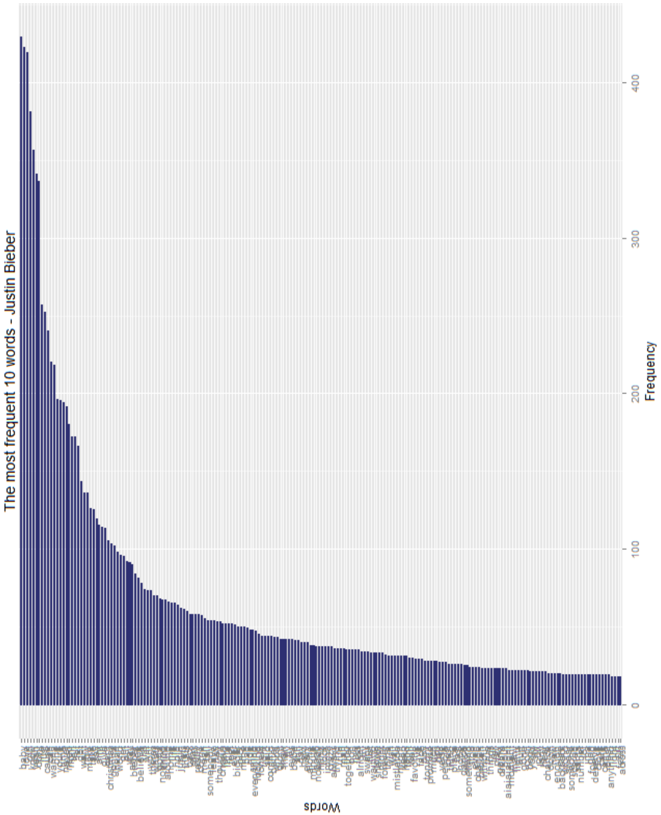
2.C. Rozkład częstości fraz

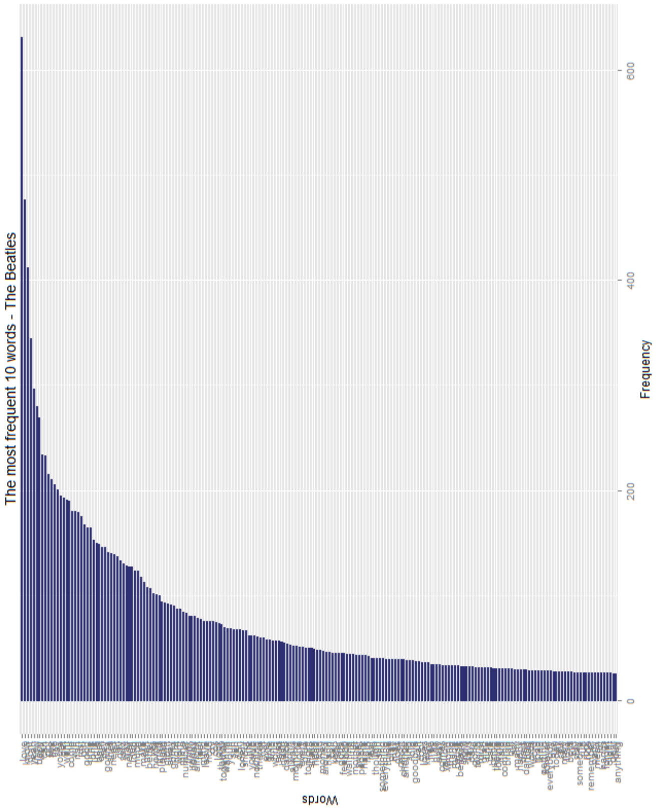




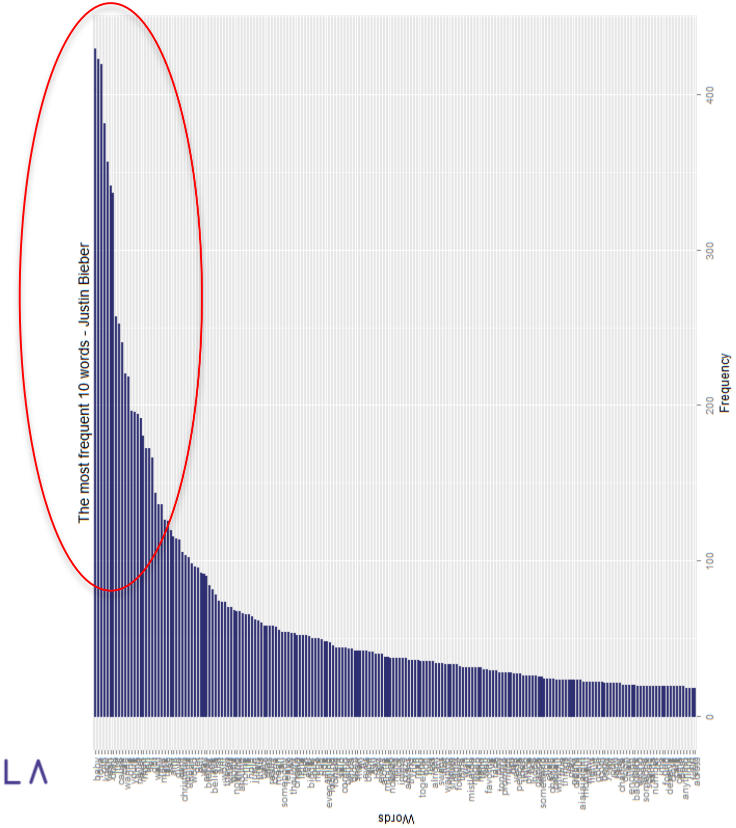
15/03/18

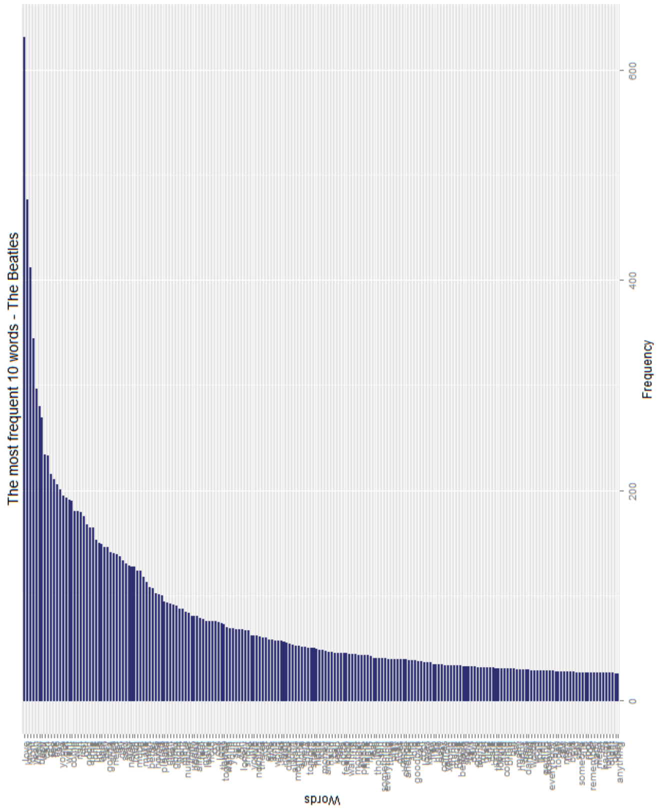
2.C. Rozkład częstości fraz





2.C. Rozkład częstości fraz





2.D. Topic Modeling

15/03/18

```
#Topic Modelling
library(topicmodels)
lda <- LDA(DTM, k=4)
topicsTerms <- terms(lda, 5)
topicsTerms
```

	Topic 1	Topic 2	Topic 3	Topic 4
1	love	girl	love	baby
2	dont	like	christmas	yeah
3	know	dont	tell	girl
4	just	yeah	never	one
5	need	know	kiss	wanna

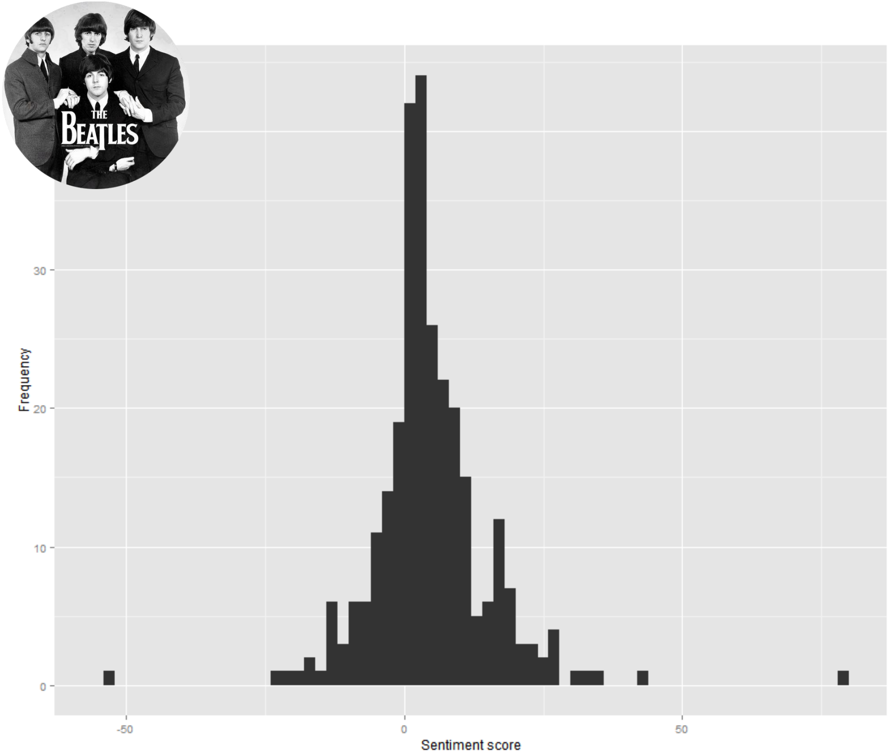
	Topic 1	Topic 2	Topic 3	Topic 4
1	love	come	now	yeah
2	dont	girl	know	good
3	know	man	baby	back
4	ill	youre	well	get
5	need	little	dont	much

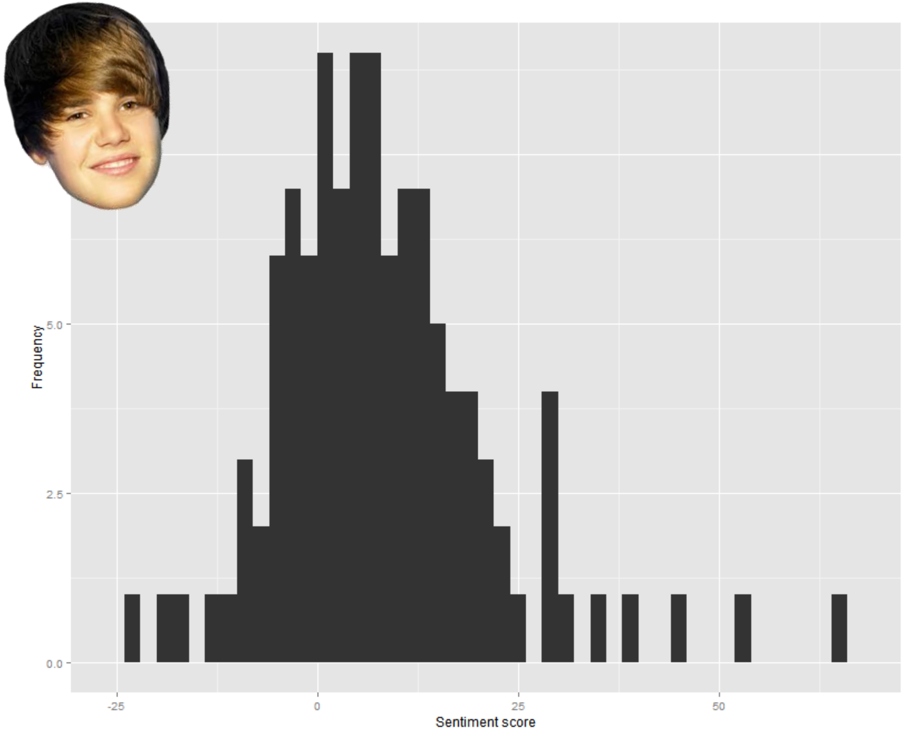
2.E. Analiza sentymentu

Leksykon Hu & Liu: http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

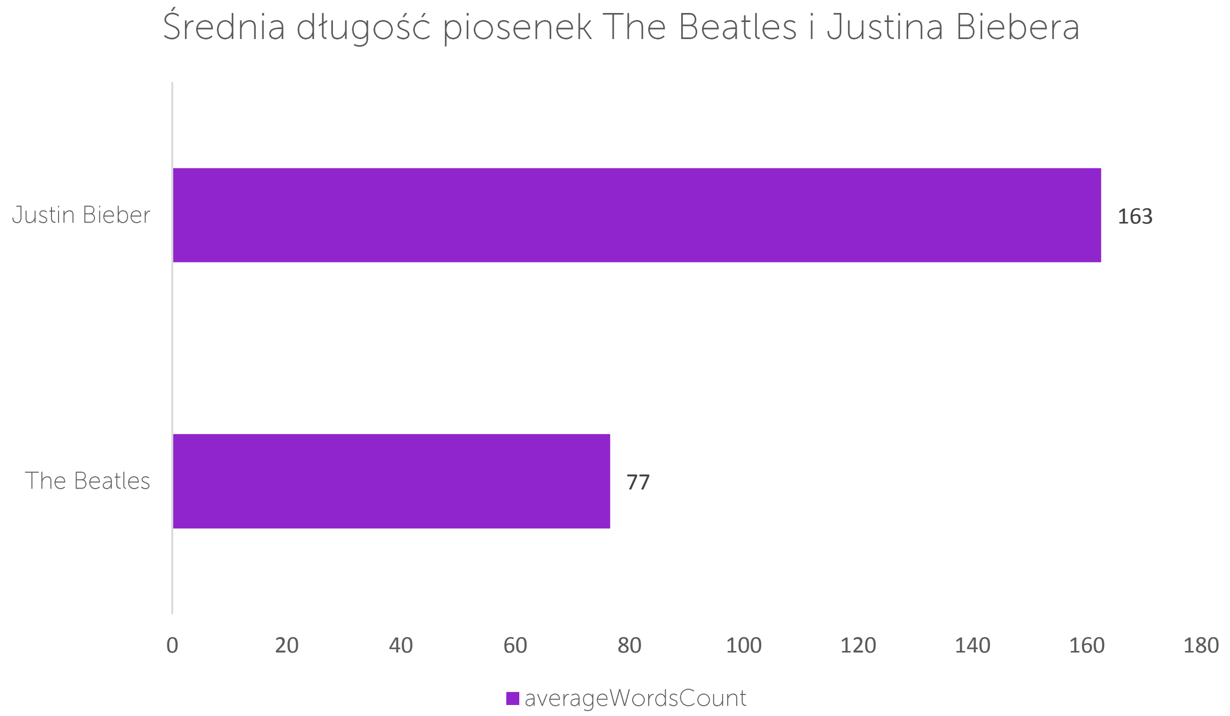
Funkcja score.sentiment J. Breen, *R by example: mining Twitter for consumer attitudes towards airlines* http://www.slideshare.net/jeffreybreen/r-by-example-mining-twitter-for

2.E. Analiza sentymentu



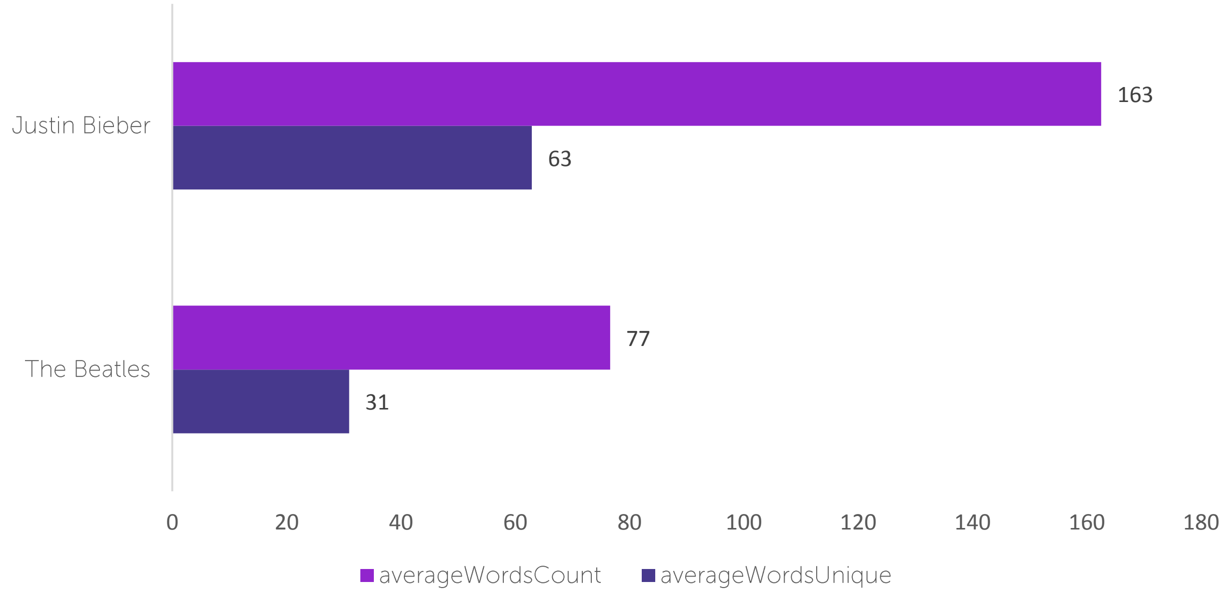


2.C. Zróżnicowanie leksylane



2.C. Zróżnicowanie leksylane

Średnia długość i liczba unikalnych słów w piosenkach
The Beatles i Justina Biebera



2.C. Zróżnicowanie leksylane

$$wordsDiversityIndex = \frac{averageWordsUnique}{averageWordsCount}$$

2.C. Zróżnicowanie leksylane

$$wordsDiversityIndex = \frac{averageWordsUnique}{averageWordsCount}$$

The Beatles	Justin Bieber
0,44	0,41

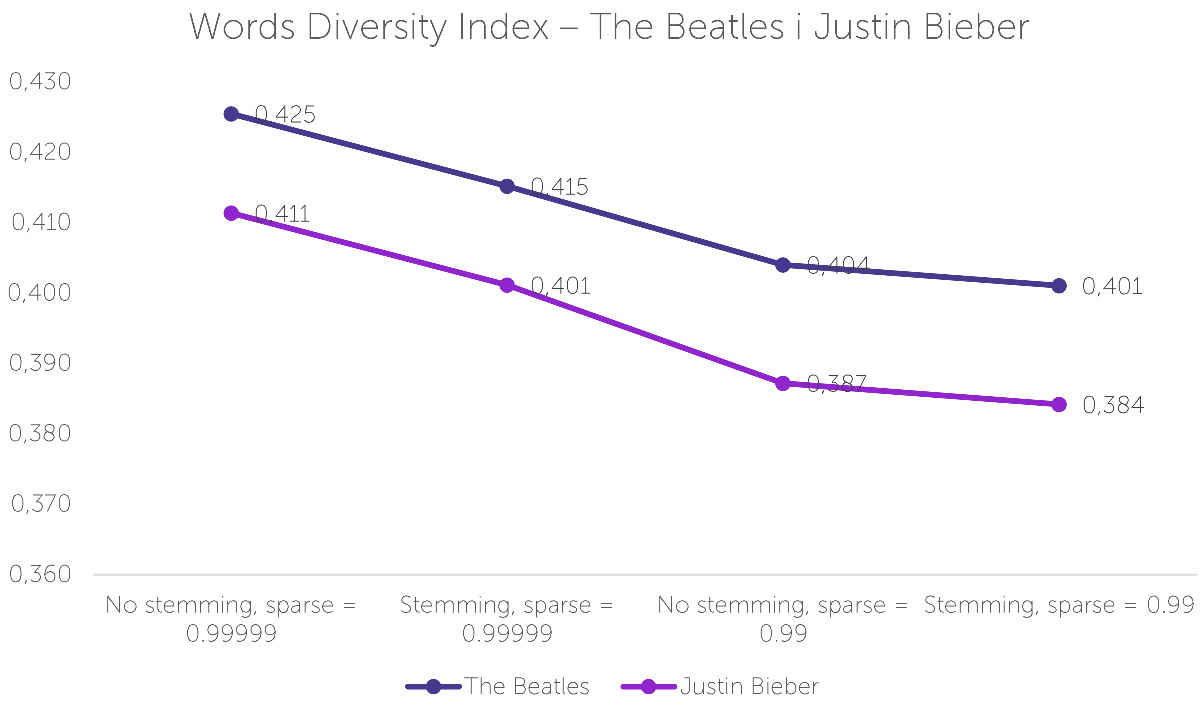
2.C. Zróżnicowanie leksylane

15/03/18

Ale chwila, chwila…

- Sparsity. Jaki przyjeliśmy próg sparsity, przy usuwaniu?
- Stemming. Sprowadzenie słów do rdzenia obniża zróżnicowanie
- Czy różnice między zbiorami tekstów są zbliżone do siebie przy różnych parametrach?

2.C. Zróżnicowanie leksylane



3. Klasyfikator tekstów w oparciu o algorytm kNN

- Czy jesteśmy w stanie rozróżnić teksty artystów?
- Czy jest to w stanie zrobić maszyna?
- Klasyfikator z wykorzystaniem kNN w R
- Pułapki


I WANT
TO PLAY A GAME...
MemGenerator.pl

“

Why she had to go I don't know she wouldn't say I said something wrong, now I long for yesterday

Why she had to go I don't know she wouldn't say I said something wrong, now I long for yesterday

"

Baby, baby, baby oooh Like baby, baby, baby nooo Like baby, baby, baby oooh I thought you'd always be mine

“

Baby, baby, baby oooh Like baby, baby, baby nooo Like baby, baby, baby oooh I thought you'd always be mine

Well, I got a baby crazy for me Yeah, I got a baby won't let me be Who, baby baby,

"

Well, I got a baby crazy for me Yeah, I got a baby won't let me be Who, baby baby,

“

I wanna be your lover baby I wanna be your man I wanna be your lover baby I wanna be your man

“

I wanna be your lover baby I wanna be your man I wanna be your lover baby I wanna be your man

“

Cause I put on my raincoat, my yellow raincoat Baby, it's keeping me dry I put on my raincoat, my yellow raincoat You know exactly why When the wind blows, and the sun goes away

“

Cause I put on my raincoat, my yellow raincoat Baby, it's keeping me dry I put on my raincoat, my yellow raincoat You know exactly why When the wind blows, and the sun goes away

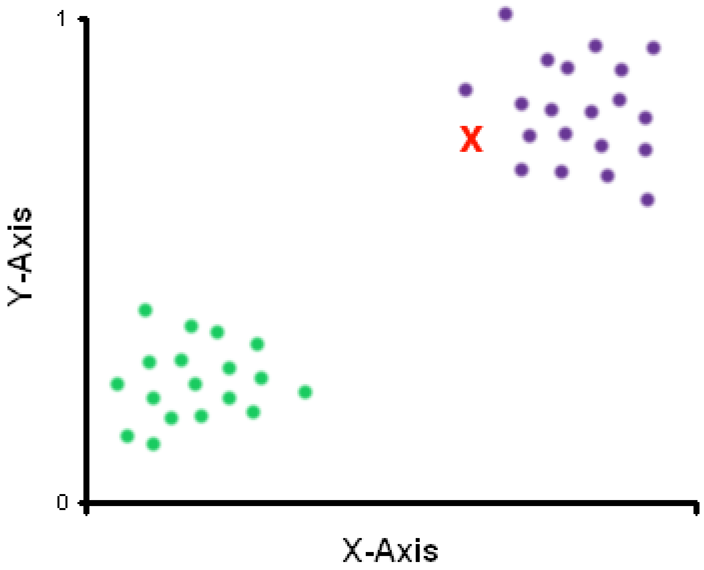
“

You're going to lose that girl If you don't take her out tonight She's going to change her mind

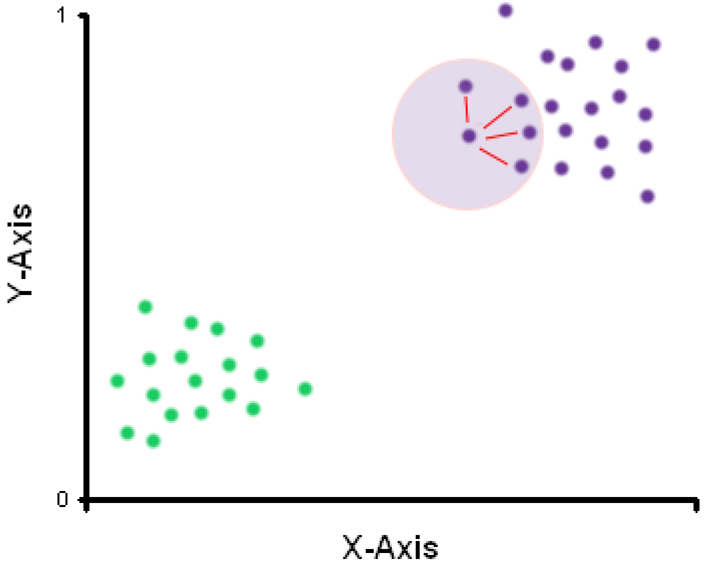
You're going to lose that girl If you don't take her out tonight She's going to change her mind

3. Klasyfikator tekstów w oparciu o kNN

15/03/18



3. Klasyfikator tekstów w oparciu o kNN



3A. Budowa korpusu

15/03/18

```
# libraries
library(tm)
library(SnowballC)
library(class)

#1. LOAD DATABASE
songsList <- read.csv2("lyrics.csv") #Bieber and Beatles songs

#2. BUILD A CORPUS
myCorpus <- Corpus(VectorSource(songsList$songLyrics)) # VectorSource specifies that the source is
character vectors.

# Clear the corpus
myCorpus <- tm_map(myCorpus, tolower) # convert to lower case
myCorpus <- tm_map(myCorpus, removePunctuation) # remove punctuation
myCorpus <- tm_map(myCorpus, removeNumbers) # remove numbers
myCorpus <- tm_map(myCorpus, stripWhitespace) # remove white space
myCorpus <- tm_map(myCorpus, removeWords, stopwords("english")) # remove stopwords
myCorpus <- tm_map(myCorpus, stemDocument, language = "english") # stemming
```

3A. DTM

```
#3. CREATE DOCUMENT TERM MATRIX

rawDTM <- DocumentTermMatrix(myCorpus, control = list(minWordLength = 3))
rawDTM <- removeSparseTerms(rawDTM, 0.95)
DTM <- as.data.frame(as.matrix(rawDTM))

#Add columns with artist, song title, length to DTM
DTM["songAuthor"] <- songsList["songAuthor"]

#Create DTM for each artist - dataframes need to be the same size
DTMBeatles <- DTM[DTM$songAuthor == "The Beatles",]
DTMBeatlesS <- DTMBeatles[ sample( which(DTMBeatles$songAuthor=='The Beatles'), 112 ), ] #same amount of
songs
DTMBieber <- DTM[DTM$songAuthor == "Justin Bieber",]
DTMBieberS <- DTMBieber[ sample( which(DTMBieber$songAuthor=='Justin Bieber'), 112 ), ] #same amount of
songs
DTMS <- rbind(DTMBieberS, DTMBeatlesS)
```

3A. Zbiór treningowy i testowy


```
########################################## KNN #############################################

#4. SAMPLING - select 50% cases out of each singer for training
sampleSize <- 0.5

train.idx_beatles <- sample(nrow(DTMBeatlesS), nrow(DTMBeatlesS)*sampleSize)
train.idx_bieber <- sample(nrow(DTMBieberS), nrow(DTMBieberS)*sampleSize)
train.idx <- rbind(train.idx_beatles, train.idx_bieber)

test.idx <- (1:nrow(DTMS)) [-train.idx] #cases except train

train.beatles <- DTMS[train.idx_beatles,]
train.bieber <- DTMS[train.idx_bieber,]
```

3A. KNN

15/03/18

```
#5. KNN
DTMkNN <- DTMS[, !colnames(DTMS) %in% "songAuthor"] #dataframe for knn function (without classification
column)

knn.pred <- knn(DTMkNN[train.idx,], DTMkNN[test.idx,], DTMsongAuthor[train.idx], k=3, prob=TRUE)
#data.frame[vector] <- select rows with certain IDs or values

#display results
conf.mat <-table("Predictions" = knn.pred, Actual = DTMsongAuthor[test.idx])
conf.mat
accuracy <- sum(diag(conf.mat) / length(test.idx) * 100)
accuracy
```

4. Materiały/ źródła

15/03/18

- **Text Mining the Complete Works of William Shakespeare** (http://www.r-bloggers.com/text-mining-the-complete-works-of-william-shakespeare/)
- **Word cloud generator in R : One killer function to do everything you need** (http://www.sthda.com/english/wiki/word-cloud-generator-in-r-one-killer-function-to-do-everything-you-need)
- **Text Mining with R -- an Analysis of Twitter Data** (http://www.slideshare.net/rdatamining/text-mining-with-r-an-analysis-of-twitter-data)
- **R by example: mining Twitter for consumer attitudes towards airlines** (http://www.slideshare.net/jeffreybreen/r-by-example-mining-twitter-for)
- **Visualizing topic models** (http://tedunderwood.com/2012/11/11/visualizing-topic-models/)
- **Sentiment Analysis with "sentiment„** (https://sites.google.com/site/miningtwitter/questions/sentiment/sentiment)
- **Statistics meets rhetoric: A text analysis of "I Have a Dream" in R** (http://www.r-bloggers.com/statistics-meets-rhetoric-a-text-analysis-of-i-have-a-dream-in-r/)

Dzięki za uwagę

Kamil Stanuch
kamil.stanuch@koalametrics.com
+48 501 93 08 93
www.koalametrics.com