Analysis of song lyrics to match genre

Assignment 1: Basic text processing

Jernej Janež (63130077), Rok Marinšek (63130146), Luka Podgoršek (63130189)

November 11, 2018

1 NLP task

For our assignment we decided to analyze song lyrics, extract keywords that correspond to specific genres and try to classify song by its lyrics to corresponding genre. First we found a dataset that cointained song lyrics. We preprocessed data, trained and tested a model and presented results with graphs. Some similar solutions already exist but perform similar task with neural networks or some other more complex methods. With this assignment we wanted to find out if our aproach can provide satisfactory results by using simple natural language processing techniques.

2 Data

We searched the internet for appropriate dataset. We found many different but in the end decided to use *380,000+ lyrics from MetroLyrics dataset* found on kaggle portal. This dataset had the attributes we needed to solve our task.
Dataset contained following attributes:

- song title,
- year,
- artist
- genre,
- lyrics.

2.1 Data preparation

Data we found was stored in *.csv* file. Because it contained more than *380 000* entries we decided to analyze songs that we're released after 2000 (latest songs in dataset). Afterwards we filtered songs to match predefind genres. We selected *hip-hop, pop and metal* and ended up with *68244* different songs. Then we removed lyrics that we're shorter than 100 words and longer than 1000 words. This way we removed outliers in data.

When we finished data preparation and selection we focused on the text preparation. First we removed special characters from text with regular expressions, converted words to lowercase and removed punctuations. Finally we removed non-english songs. This way we ended up with **5613** different songs.

#	Genre	Number of different songs
0	Hip-Hop	20902
1	Metal	16632
2	Pop	30710

Table 1: Number of songs per genre

In the end we saved filtered data into *.csv* file. In our model class we used this file as input to train our model. You can also use this file to replicate our results.

3 Model

To train our model we used preprocessed file. Model is build with logistic regression.

3.1 Train, test data and metrics

To train our model we used 80% of data and 20% to test our model. To measure score and performance of our model we used following metrics:

- accuracy,
- precision,
- recall,
- and f1 score.

In development phase we also played with regularization factor. We used above mentioned metrics to determine best regularization factor. In the end we set it to 0.1.

3.2 Resources, tools and corpora

We used several different python libraries. Pandas was used for data structures and data purging. Nltk corpus was used to determine stopwords and for lematization. Langdetect library was used to remove non-english lyrics. To build our model we used sklearn and preseted results with matplotlib.

4 Algorithm-Model description

4.1 Building bag of words with TF-IDF

Bag of words is an algorithm that counts how many times a word appears in a document. We used WhitespaceTokenizer function from the ntlk library to tokenize our song lyrics and removed

all the stop words, because words such as *"and"* or *"the"* appear frequently in all songs and are systematically discounted. Then we used the WordNetLemmatizer function also from the nltk library to lemmatize the text which grouped together the different inflected forms of a word so they can be analyzed as a single item. After that we vectorized our data with TF-IDF function called TfidfVectorizer from sklearn which measured the number of times words appeared in a given song (term frequency). The more songs a word appears in, the less valuable that word is as a signal. That's intended to leave only the frequent AND distinctive words as markers. Each word's TF-IDF relevance is a normalized data format that also adds up to one.

5 Results

Before we could interprete our model and it's predictions we had to test it. Testing phase consisted of 10 iterations. In each iteration we sampled data with different random seed, trained a model and tested it on sampled test data. In the end we averaged all metrics to score performance of our model. You can find results in table bellow.

Accuracy	Precision	Recall	F1
0.736	0.769	0.736	0.745

Table 2: Model scores

Beside performance scores our model returned list of keywords that have the most and the least value for classification. These results are presented in tables bellow. Graphical presetation of tables is included in appendix graphs section.

(+) Keywords	Weight	(-) Keywords	Weight
death	2.14	love	-3.04
blood	1.95	like	-2.23
dead	1.80	oh	-2.10
fucking	1.68	baby	-2.07
end	1.57	girl	-2.01
pain	1.51	i'm	-1.94
hate	1.48	ain't	-1.93
lie	1.47	yeah	-1.92
die	1.44	got	-1.90
fear	1.42	get	-1.61

(+) Keywords	Weight	(-) Keywords	Weight
nigga	3.90	i've	-1.27
shit	2.72	eye	-1.11
ain't	2.63	lie	-1.07
yo	2.36	dream	-0.96
like	2.32	nothing	-0.86
bitch	2.24	away	-0.83
girl	1.90	heart	-0.80
fuck	1.75	alone	-0.75
cause	1.74	care	-0.73
'em	1.71	world	-0.71

Table 3: Keywords for: **metal** (left) & **hip-hop** (right)

5.1 Postagging

5.2 Understanding wrong classifications

If you look at confusion matrix confusion matrix you can notice that most commongly miss-classified songs are pop songs. We didn't expect to classify every song correctly, but pop songs

(+) Keywords	Weight	(-) Keywords	Weight
love	2.17	nigga	-2.47
oh	1.85	shit	-2.45
heart	1.58	fuck	-2.21
boy	1.32	death	-1.85
gonna	1.22	fucking	-1.72
baby	1.06	dead	-1.65
blue	1.05	blood	-1.53
'cause	1.03	bitch	-1.49
could	1.00	die	-1.47
kiss	0.99	hate	-1.40

Table 4: Keywords for **pop**

Table 5: Nouns for: **metal** (left) & **hip-hop** (right)

Table 6: Nouns for **pop**

have larger error rate than metal or hip-hop songs. We tried to understand why this happens. Our hypothesis was that many pop songs artist feature artist from hip-hop or metal.

To test this we extracted songs that we not correctly classified. Afterwards we checked if song title contains word featuring or ft or feat. We counted percentage of these songs. You can see results in table bellow.

6 Conclusion

7 Github repository

Github repository: https://github.com/marok39/onj-02

Appendix

A Graphs

Here you can find visualization of all results. These files can be found alongside report in /img directory.

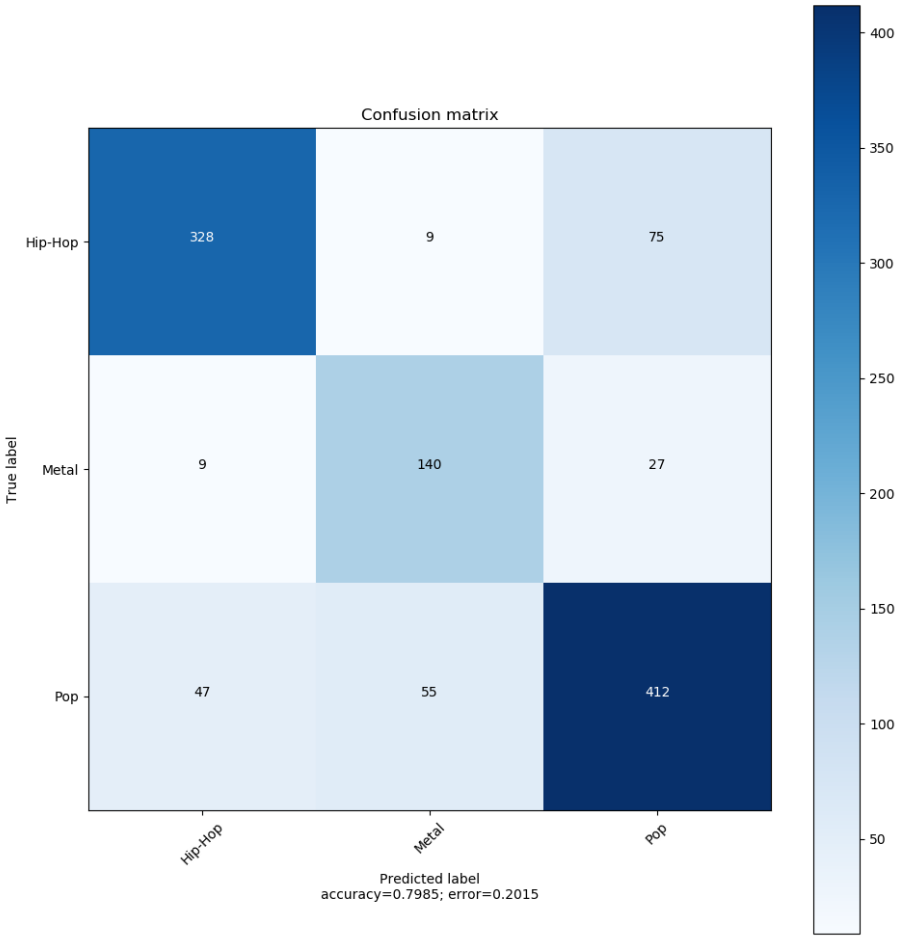

Figure 1: Confisuin matrix

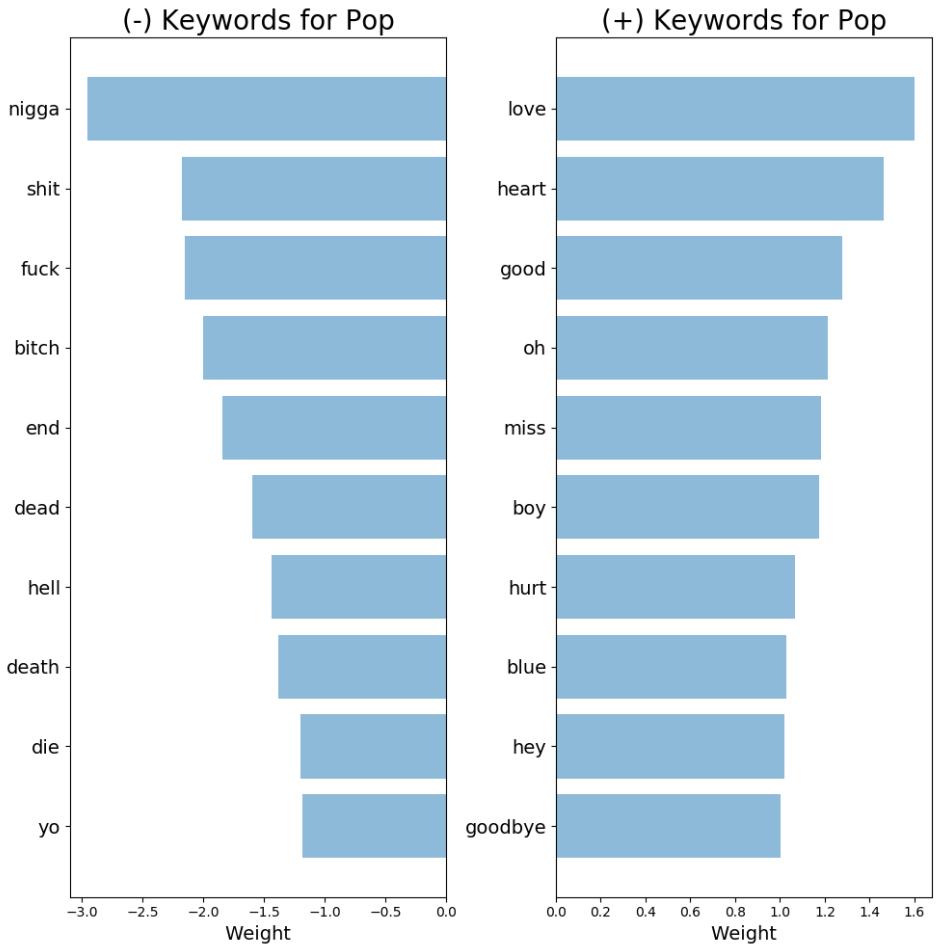

Figure 2: Keywords visualization for pop

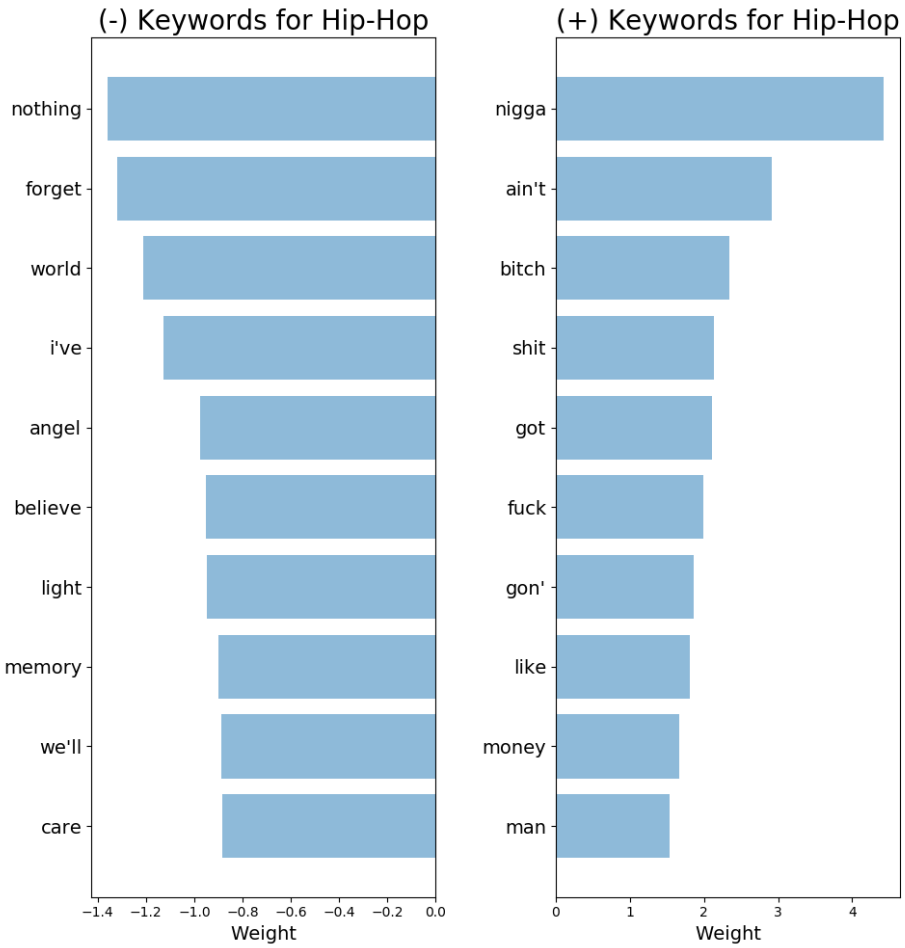

Figure 3: Keywords visualization for hip-hop

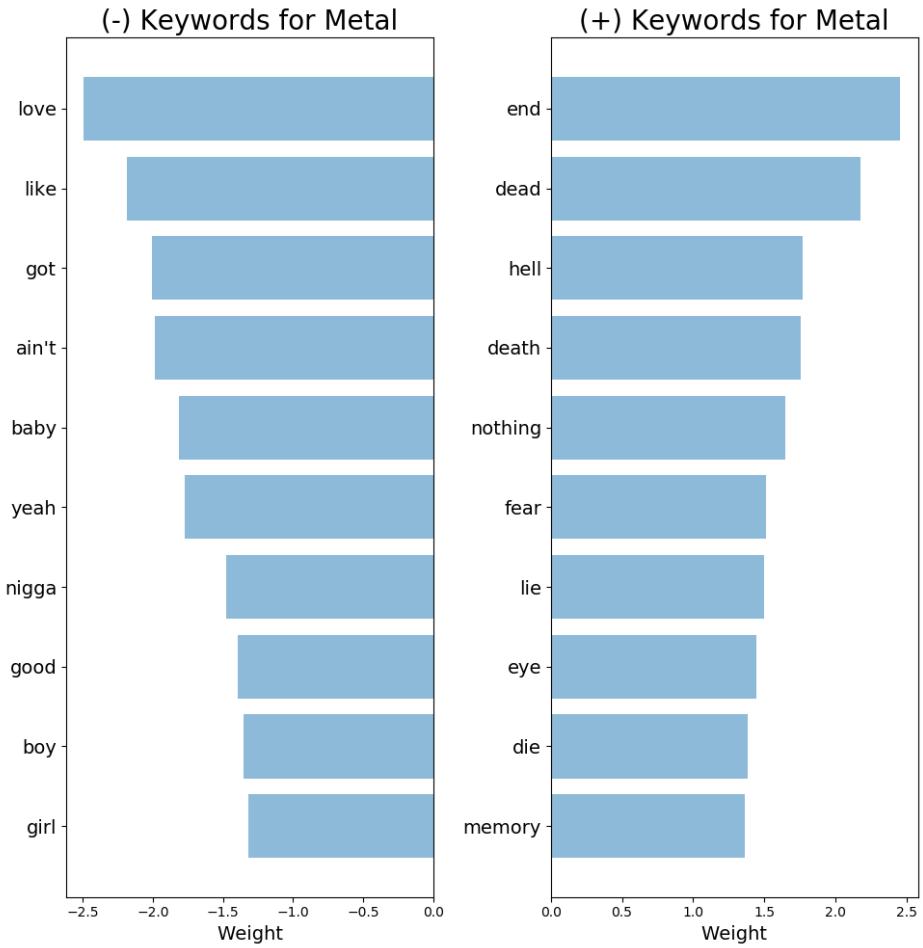

Figure 4: Keywords visualization for metal