CS464 Machine Learning
Project Progress Report

“Song Lyrics Decade and Popularity Classification”

Group Members
Elif Kurtay 21803373 Cansu Moran 21803665 Atakan Dönmez 21803481
Öykü Irmak Hatipoğlu 21802791 Elif Gamze Güliter 21802870

Table of Content

1. Introduction

In this project, we aim to perform two different classification tasks on song lyrics using different text vectorization techniques with Recurrent Neural Networks(RNN) and Multinomial Naive Bayes classifiers. First, we will classify song lyrics into decades to observe what kind of distinction the song lyrics have in different decades, for example 70s, 80s or 90s. The second classification task we will perform is based on the popularity of the songs in each decade with a scale of 1 to 100. Since having a class for each popularity value will not be sensible, we will perform regression to predict popularity as a continuous value. We will train a different regression model for each decade and we will try to observe the predictions on how popular a song would be if it was released in each decade. Hence, we will be able to compare a song’s predicted popularity in its predicted release decade versus other decades as a result of both classification tasks.

In order to predict the era and popularity of a song, we will train two models that are popular in text classification tasks: Multinomial Naive Bayes and RNN with Long Short-Term Memory (LSTM). For RNN, we will use three different word vectorization techniques: Word-to-Vector(Word2Vec), Global Vector(GloVe) and Bag-of-Words(BoW) with Term Frequency-Inverse Document Frequency(TF-IDF). We will use BoW with TF-IDF also for the Multinomial Naive Bayes model.

In the following section, Background Information, we will present previous work based on lyrics classification, and how we will benefit from them in addition to information about the different techniques we will use for our project. Following that, in section Work Done So Far, we will explain at which stage of the project we are currently in with some of the results we have acquired so far. In the Remaining Tasks section we will list which steps we will need to perform to finalize the project. In the last section, Division of Work, we will display how we separated the work into work packages and which team members are allocated for each work package.

2. Background Information

Since our main goal is to perform decade and popularity classification tasks on song lyrics, we studied previous research where classification tasks are performed on song lyrics. We found a time series analysis on American songs that compare simplicity in the lyrics [1]. However, recent research is focused on classification of music genres according to the lyrical data. We believed the models and data preparation used for music genre classification is similar enough to be experimented for period and popularity classification. Therefore, we based our research on music genre classification.

In genre classification, previous research has shown that lyrical data performs the weakest when compared to other forms of data [2]. Previous non-neural lyrical classifiers, such as SVMs, k-NN, and Naive Bayes, are found to have struggled to achieve a classification accuracy higher than 50% [3]. Often the lyrical data is combined with other forms of data to produce superior classifiers. Mayer et al. combine audio and lyrical data to produce a highest accuracy of 63.50% within 10 genres via SVMs [4]. Mayer and Rauber then use a cartesian ensemble of lyric and audio features to gain a highest accuracy of 74.08% within 10 genres [5]. However, since we want our sole focus to be text classification, we decided not to leverage other numerical auditory data(such as tempo, vividness, …) and thus accept to not achieve high accuracy results. On the other hand, we are using a dataset that also includes this type of information on the audio of the songs such as but not limited to instrumentalness, energy, tempo, speechiness etc. While, as mentioned before, we are not using this data to perform classification, we used some of these attributes of the audio to identify whether a track actually has lyrics or not, to ease the data retrieval process.

Three text vectorization techniques are being used for the lyrical dataset: Global Vectors (GloVe), Word-to-Vector (Word2Vec), and Bag-of-Words (BoW) with Term Frequency-Inverse Document Frequency(TF-IDF). Word2Vec is a technique for natural language processing that transforms each word into a vector of numbers paying attention to the relation of the word with its neighbors. GloVe is another word representation technique valuing the context of a word by its relation to other words. However, GloVe takes the words' co-occurrence count in all documents into consideration whereas its predecessor Word2Vec only considers the neighboring words in one document [6], [7], [8]. BoW representation vectorizes the word without a context unlike the previous models. It only represents a word in terms of its frequency in the document but we plan to use BoW technique with TF-IDF which gives a value to a word according to its importance and difference with regard to other documents [9]. Further detailed information about each technique can be found in their respective sections under 3.b. Data Preprocessing Before Training Models.

3. Work Done So Far

a. Data Retrieval and Dataset Preparation

We used the Spotify Data 1921-2020 dataset on Kaggle [10] to extract artists, names, popularities, release years and likewise properties of songs. The classification is performed on English lyrics, and thus any song in a language other than English is removed from the dataset. The language of the songs are determined with a “langdetect” package found in Python 3.7 library package [11]. This library is a machine learning classification library. Therefore, it is observed that results on the same data can sometimes alternate. However, since we could not modify the algorithm or have the time for a manual control, we accepted possible wrong labeling as noise in the data. Furthermore, due to the time consuming lyric extraction API, we wanted to reduce the data on the basis of language before extracting the lyrics. Hence the language detection algorithm is used on the song titles of the songs. In case of unidentified results on song titles, such as a song title on being a numerical value, our algorithm is made to look at artist names to determine the language of the song. This technique is proven to diminish all errors. However, the accuracy is not perfect as seen from the resulting data. Nevertheless, we still decided to keep the results with the noise. The languages found in the dataset with instrumentalness value less than 0.5 (meaning the song is predicted to contain lyrics) can be observed in Figure 1.

```
en       76481
de        9662
es        7024
it        4987
pt        4022
fr        3938
id        3419
tl        3171
nl        3018
af        2280
ru        2000
so        1879
no        1749
ca        1710
sw        1681
ro        1503
da        1455
cy        1351
fi        1315
et        1137
tr        1068
pl         960
sv         851
ko         628
sl         559
lt         541
el         513
hr         401
hu         386
sk         353
vi         268
sq         250
cs         229
zh-cn      179
zh-tw      101
lv          92
ja          73
he          43
ar          41
uk          31
bg           6
mk           5
th           3
fa           3
Name: language, dtype: int64
```

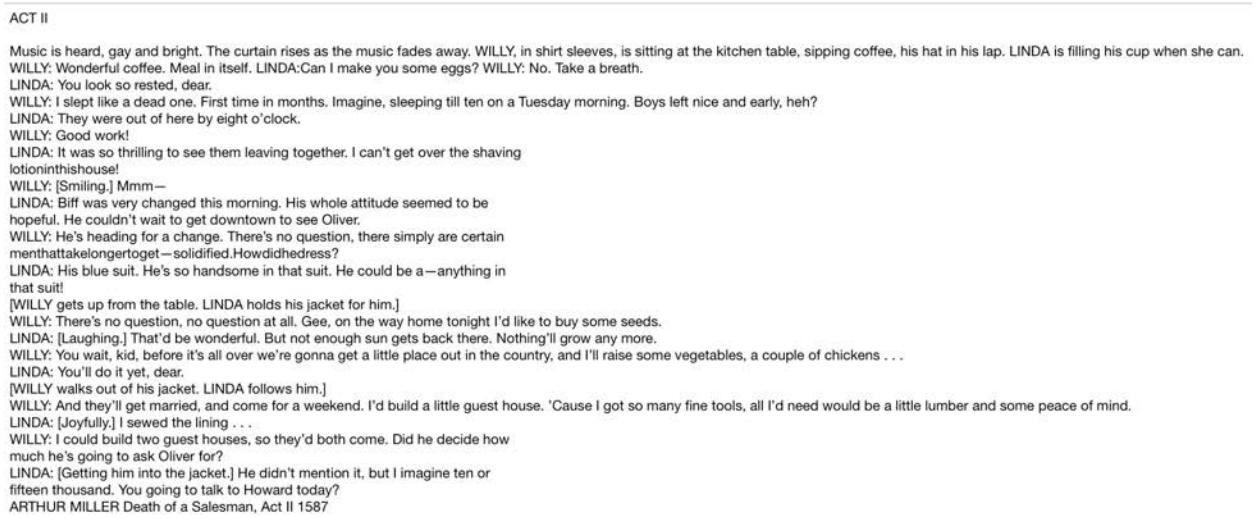
Figure 1: The number of languages found in the dataset with instrumentalness value less than 0.5.

Furthermore, the period label is obtained from the "year" feature which is the year value of the release dates of the songs. The amount of English songs per period is found in Figure 2.

```
90    20000
80    20000
70    20000
60    20000
0     20000
50    19950
10    19900
40    14968
30     8889
20     6202
Name: period, dtype: int64
Total sum:  169909
```

Figure 2: Number of English songs per period found in the Spotify dataset from Kaggle.

LyricsGenius API [12] was used to extract the lyrics of each song in the Kaggle dataset from genius.com. We were initially planning to use Az API [13] to extract the lyrics from azlyrics.com. However, the API had a low retrieval rate for songs present in the Azlyrics website and often had connection issues. Therefore, we found another API which didn't have as many connection problems. Since we still ran into a timeout problem with the LyricsGenius API, we added three retries to reconnect and try to fetch the lyrics. With this technique, we were able to retrieve lyrics for the given songs, however, we still had a problem of getting incorrect lyrics for certain songs that didn't actually have lyrics on the Genius website. In such cases, the API is designed to return the closest match to the given song, which is not the desired output in our scenario. For example, for the song "Fascinating Rhythm" by Don Rolke [14] , the API found the "lyrics" displayed in Figure 3.



ACT II

Music is heard, gay and bright. The curtain rises as the music fades away. WILLY, in shirt sleeves, is sitting at the kitchen table, sipping coffee, his hat in his lap. LINDA is filling his cup when she can.
WILLY: Wonderful coffee. Meal in itself. LINDA:Can I make you some eggs? WILLY: No. Take a breath.
LINDA: You look so rested, dear.
WILLY: I slept like a dead one. First time in months. Imagine, sleeping till ten on a Tuesday morning. Boys left nice and early, heh?
LINDA: They were out of here by eight o'clock.
WILLY: Good work!
LINDA: It was so thrilling to see them leaving together. I can't get over the shaving
lotioninthishouse!
WILLY: [Smiling.] Mmm—
LINDA: Biff was very changed this morning. His whole attitude seemed to be
hopeful. He couldn't wait to get downtown to see Oliver.
WILLY: He's heading for a change. There's no question, there simply are certain
menthattakelongertoget—solidified.Howdidhedress?
LINDA: His blue suit. He's so handsome in that suit. He could be a—anything in
that suit!
[WILLY gets up from the table. LINDA holds his jacket for him.]
WILLY: There's no question, no question at all. Gee, on the way home tonight I'd like to buy some seeds.
LINDA: [Laughing.] That'd be wonderful. But not enough sun gets back there. Nothing'll grow any more.
WILLY: You wait, kid, before it's all over we're gonna get a little place out in the country, and I'll raise some vegetables, a couple of chickens . . .
LINDA: You'll do it yet, dear.
[WILLY walks out of his jacket. LINDA follows him.]
WILLY: And they'll get married, and come for a weekend. I'd build a little guest house. 'Cause I got so many fine tools, all I'd need would be a little lumber and some peace of mind.
LINDA: [Joyfully.] I sewed the lining . . .
WILLY: I could build two guest houses, so they'd both come. Did he decide how
much he's going to ask Oliver for?
LINDA: [Getting him into the jacket.] He didn't mention it, but I imagine ten or
fifteen thousand. You going to talk to Howard today?
ARTHUR MILLER Death of a Salesman, Act II 1587

Figure 3: A snippet of the lyrics retrieved with LyricsGenius for the song "Fascinating Rhythm" by Don Rolke.

It can be seen that the displayed information is not actually lyrics and is completely irrelevant from the song. We realized that these songs that had the incorrect data were instrumental songs. Therefore, the API was retrieving the closest match to the song name and artists on the genius.com website. To eliminate this problem, we filtered the songs according to their

speechiness and instrumentalness values which were found in the Kaggle dataset. According to [15], instrumentalness reflects whether a song contains vocals. Songs with instrumentalness value above 0.5 are predicted to be instrumental by Spotify. On the other hand, the speechiness value of a song depends on the spoken words on the track. Songs with speechiness value lower than 0.33 are predicted to be non-speech-like tracks. Since our main concern is finding whether a song is instrumental, we filtered our songs according to their instrumentalness value first. The number of English songs with instrumentalness value lower than 0.5 is shown in Figure 4.

```
70    11815
80    11020
60    10833
0     10153
10    10086
90     9732
50     7045
40     2435
30     1923
20     1439
Name: period, dtype: int64
Total sum:  76481
```

Figure 4: Number of English songs with instrumentalness value less than 0.5 per period.

After the filtering, we checked whether some instrumental songs we have detected before were still in the dataset. Figure 5 displays songs that were still in the dataset after the filtering was applied. As mentioned before, “Fascinating Rhythm” by Don Ralke is an instrumental track and as seen in the figure, it was not filtered with the instrumentalness value. Moreover, out of the three songs displayed, only “Fascinating Rhythm” by Ella Fitzgerald contains lyrics. Since these values are predicted by Spotify, it is open to errors, as seen in the instrumentalness and speechiness values in the figure. Taking songs with instrumentalness lower than 0.5 was not sufficient to filter instrumental songs in the dataset, therefore we decided to filter the songs according to their speechiness values as well. Figure 6 shows the number of songs with instrumentalness value lower than 0.5 and speechiness value larger than 0.33.

	artists	id	instrumentalness	name	popularity	speechiness
2108	['Don Ralke']	6NTceHSsE8bSWBE0hEVUUl	0.000973	Fascinating Rhythm	12	0.1360
100580	['Stan Kenton']	1rZJBZO40kMWPLMl5BuiPo	0.000202	Fascinating Rhythm	9	0.0389
138224	['Ella Fitzgerald']	03whQz2AOGIKgNU3NQW6a0	0.000010	Fascinating Rhythm	21	0.0764

Figure 5: Songs named “Fascinating Rhythm” in the dataset with their instrumentalness and speechiness values.

```
10    174
90    159
50    145
70    102
0     102
40     91
60     75
80     69
30     55
20     35
Name: period, dtype: int64
Total sum:  1007
```

Figure 6: Number of songs in each period with instrumentalness value lower than 0.5 and speechiness value larger than 0.33.

When we filtered the songs according to their instrumentalness and speechiness values, the number of songs in the dataset dropped quite significantly. While we are aware that having less songs in the dataset can affect the performance of our models, having random noisy data would also have such an effect. Therefore, filtering these songs was a trade-off and we decided that filtering was necessary due to the high number of instrumental songs in the dataset. Since there is a lower probability of the website containing lyrics for songs earlier than 60s these labels have been removed from the dataset to decrease miss rate during retrieval and hence prevent any random lyrics being retrieved by the API.

We also found some API related word errors in the correct lyrics data. For example, some lyrics finished with the phrase “16EmbedShare URLCopyEmbedCopy” which shouldn’t be part of the lyrics. We removed such API related irrelevant phrases from the retrieved lyrics. Other preprocessing of the lyrics performed that are unrelated to the API are shared in the Data Cleaning section under Data Preprocessing Before Training Models.

The retrieval process took approximately 1.21 minutes per song, which means that it would take 12 days to retrieve all English songs, without any filtering. Therefore filtering also decreased the retrieval time significantly. Retrieving all the songs after filtering took approximately a day. We formed the dataset using the name, artist, popularity, period and lyrics of each song in addition to a unique id. However, because we were working concurrently on data retrieval and word vectorization techniques, in order to have a certain amount of data to test the word vectorization techniques on, we also formed a sample dataset with 10 songs per period. All the word vectorization techniques were tried on the sample dataset to shorten the time of getting results from the techniques and make it easier to analyze them to see whether there were any problems.

After the formation of the sample dataset, we wanted to go over the lyrics to eliminate any non-English words. However, the language detection algorithm did not achieve good results on the data as can be observed in Figures 7 and 8. Many words that are actually English are deleted in the songs after the English word detection. Hence, we decided to keep the lyrics without a second English filter.

id	period	artists	name	popularity	lyrics
›Vvuv	60	['Sam Cooke']	(Ain't That) Good News	13	oh baby comin home tomorrow good news man news...
‘flBP	60	['Marilyn Monroe']	My Heart Belongs to Daddy	14	name lolita er supposed play boy moi mon coeur...
ΕVUUl	60	['Don Ralke']	Fascinating Rhythm	12	act ii music heard gay bright curtain rise mus...
ɜOOg	60	['Mel Tormé', 'The Marty Paich Orchestra']	Too Darn Hot	14	like sup baby tonight refill cup baby tonight ...
‘wLR2	60	['Johnny Cash']	When Papa Played the Dobro	14	papa hobo delivered doctor cause pay fee going...

Figure 7: Sample dataset with lyrics after cleaning on the lyrical data is performed.

id	period	artists	name	popularity	lyrics
Vvuv	60	['Sam Cooke']	(Ain't That) Good News	13	tomorrow tomorrow told told found know tomorro...
lBP	60	['Marilyn Monroe']	My Heart Belongs to Daddy	14	tearing follow heart heart heart heart though ...
VUUl	60	['Don Ralke']	Fascinating Rhythm	12	heard bright coffee hat coffee breath rested f...
OOg	60	['Mel Tormé', 'The Marty Paich Orchestra']	Too Darn Hot	14	tonight tonight tonight tonight tonight tonigh...
vLR2	60	['Johnny Cash']	When Papa Played the Dobro	14	fee ease played around knew played top sound p...

Figure 8: Sample dataset with lyrics after cleaning and an English language check done on each word on the lyrical data is performed.

b. Data Preprocessing Before Training Models

- **Data Cleaning**

In order to make the lyrics meaningful to train our models, we needed to clean the data. The data with the retrieved lyrics can be seen in Figure 9. The output string includes insignificant and unwanted words and expressions such as “[Verse]”, newline characters (\n), and many punctuations. First, we removed the punctuations and non-alphanumeric words using the RegEx library named “re” from Python while turning all string values into lower case [16]. The result of the task can be seen in Figure 10. For example, the word “baby’s” turned into “baby” since the apostrof gets separated as a different word during lemmatization, we wanted to remove these before as well.

lyrics
Oh my baby's comin' home tomorrow\nAin't that ...
My name is Lolita ...\nAnd er...\nI'm not supp...
ACT II\n\nMusic is heard, gay and bright. The ...
I'd like to sup with my baby tonight\nRefill t...
[Verse]\nMy Papa was a hobo when they delivere...

Figure 9: Sample dataset with lyrics extracted using a third party API.

lyrics
oh my baby comin home tomorrow ain that good n...
my name is lolita and er not supposed to play ...
act ii music is heard gay and bright the curta...
like to sup with my baby tonight refill the cu...
verse my papa was hobo when they delivered me ...

Figure 10: Sample dataset with lyrics extracted using a third party API after punctuations are removed.

Then, we extracted stopwords in English vocabulary from the NLTK [17] library that included words that do not add to the meaning of texts such as “then”, “is”, “the”, “where”, etc. However, we discovered that lyrical data also had extra stopwords such as “verse”, “chorus” and “intro”. Therefore, we created a common list of stopwords to be removed from the dataset. Lastly, we lemmatized the words in the lyrics. By that means, we grouped the inflected forms of a word into the word's dictionary form so they can be analyzed together. For example, after lemmatizing , the words “connect, connected, connects” are turned into “connect” and grouped together with other “connect” words in data. We used a lemmatization model from the NLTK library for this purpose but the effects of this task are not so visible on the data. With this, we concluded the cleaning operation on our data. The cleaned version of the lyrical data can be found in Figure 11.

lyrics
oh baby comin home tomorrow good news man news...
name lolita er supposed play boy moi mon coeur...
act ii music heard gay bright curtain rise mus...
like sup baby tonight refill cup baby tonight ...
papa hobo delivered doctor cause pay fee going...

Figure 11: Sample dataset with lyrics after cleaning is complete.

```
['oh', 'baby', 'comin', 'home', 'tomorrow', 'good', 'news', 'man', 'news', 'baby',
['name', 'lolita', 'er', 'supposed', 'play', 'boy', 'moi', 'mon', 'coeur', 'est',
['act', 'ii', 'music', 'heard', 'gay', 'bright', 'curtain', 'rise', 'music', 'fade
['like', 'sup', 'baby', 'tonight', 'refill', 'cup', 'baby', 'tonight', 'like', 'su
['papa', 'hobo', 'delivered', 'doctor', 'cause', 'pay', 'fee', 'going', 'got', 'ba
['long', 'shot', 'view', 'cul', 'de', 'sac', 'seen', 'look', 'peaceful', 'quiet',
['oh', 'yeah', 'everybody', 'oh', 'oh', 'oh', 'oh', 'aye', 'aye', 'aye', 'aye', 'c
['pad', 'heat', 'sullivan', 'street', 'last', 'hipster', 'lay', 'dyin', 'wearin',
['gave', 'u', 'glamorous', 'career', 'talent', 'big', 'star', 'year', 'public', 'p
['knocking', 'laboratory', 'door', 'much', 'busy', 'would', 'answer', 'igor', 'exc
['dryer', 'telegram', 'came', 'private', 'john', 'miller', 'shot', 'vietnam', 'tea
['got', 'key', 'highway', 'yeah', 'billed', 'bound', 'go', 'got', 'ta', 'keep', 'v
['year', 'ago', 'place', 'called', 'land', 'point', 'everything', 'land', 'point',
['wan', 'na', 'know', 'good', 'baby', 'know', 'must', 'good', 'baby', 'cause', 'su
['yeah', 'seem', 'civilized', 'mama', 'tryin', 'run', 'life', 'daddy', 'tryin', 'p
['multitude', 'people', 'cry', 'try', 'understand', 'book', 'read', 'one', 'must',
['stand', 'time', 'checking', 'take', 'one', 'instrumental']
```

Figure 12: Snippet of lemmatized words of the sample dataset for the Naive Bayes model.

```
print(lyric_stopwords)
```

```
, 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up','down', 'in', 'out', 'on', 'off', 'c
```

Figure 13: Some stop words that were eliminated from the sample dataset.

- **Text Vectorization**

In Machine Learning, data needs to be adjusted for the models that will be used. The models we will use need the string data to be represented with numerical values which is referred to as **text**

vectorization. However, there are different ways of achieving this goal. For our models we chose 3 different ways: BoW with TF-IDF, Word2Vec, and GloVe. The BoW with TF-IDF method will be used to train our Multinomial Naive Bayes model whereas all three techniques will be used to train the RNN-LSTM model. Detailed explanation about each technique is given in the following subsections.

- **Bag of Words with TF-IDF**

After cleaning the data, we need to turn our strings into a "bag of words" matrix for the Multinomial Naive Bayes model. Figure 14 shows the matrix where each column represents a word and each row represents a song. The entries in the matrix represent the number of occurrences of a word in a song. For example, the word "03" occured 3 times in the song "38". On the other hand, the word "02" did not occur in the song with the row number "38". This matrix is achieved by using the CountVectorizer in the NLTK library.

On the other hand, there are multiple ways to count the occurrence of a word in documents. One way is called Binomial which gives the value 1 if the word exists and 0 if it does not exist in a document by disregarding the frequency of that word. The Multinomial approach shows the frequency of that word in each document which is the approach used by the CountVectorizer. In our project, we wanted to choose another approach which is called TF-IDF.

	00	000	001	02	03	06	...	zumo	zuper	zuperman	zw1tch	ørpheus	静的
0	0	0	0	0	0	0	...	0	0	0	0	0	0
1	0	0	0	0	0	0	...	0	0	0	0	0	0
2	0	0	0	0	0	0	...	0	0	0	0	0	0
3	0	0	0	0	0	0	...	0	0	0	0	0	0
4	0	0	0	0	0	0	...	0	0	0	0	0	0
5	0	0	0	0	0	0	...	0	0	0	0	0	0
6	0	0	0	0	0	0	...	0	0	0	0	0	0
7	0	0	0	0	0	0	...	0	0	0	0	0	0
8	0	0	0	0	0	0	...	0	0	0	0	0	0
9	0	0	0	0	0	0	...	0	0	0	0	0	0
10	0	0	0	0	0	0	...	0	0	0	0	0	0
11	0	0	0	0	0	0	...	0	0	0	0	0	0
12	0	0	0	0	0	0	...	0	0	0	0	0	0
13	0	0	0	0	0	0	...	0	0	0	0	0	0
14	0	0	0	0	0	0	...	0	0	0	0	0	0
15	0	0	0	0	0	0	...	0	0	0	0	0	0
16	0	0	0	0	0	0	...	0	0	0	0	0	0
17	0	0	0	0	0	0	...	0	0	0	0	0	0
18	0	1	1	0	1	1	...	0	0	0	0	0	0
19	0	0	0	0	0	0	...	0	0	0	0	0	0
20	0	0	0	0	0	0	...	0	0	0	0	0	0
21	0	0	0	0	0	0	...	0	0	0	0	0	0
22	0	0	0	0	0	0	...	0	0	0	0	0	0
23	0	0	0	0	0	0	...	0	0	0	0	0	0
24	0	0	0	0	0	0	...	0	0	0	0	0	0
25	0	0	0	0	0	0	...	0	0	0	0	0	0
26	0	0	0	0	0	0	...	0	0	0	0	0	0
27	2	0	0	0	1	2	...	0	0	0	0	0	0
28	0	0	0	0	0	0	...	0	0	0	0	0	0
29	0	0	0	0	0	0	...	0	0	0	0	0	0
30	0	0	0	0	0	0	...	0	0	0	0	0	0
31	0	0	0	0	0	0	...	0	0	0	0	0	0
32	0	0	0	0	0	0	...	0	0	0	0	0	0
33	0	0	0	0	0	0	...	0	0	0	0	0	0
34	0	0	0	0	0	0	...	0	0	0	0	0	0
35	0	0	0	0	0	0	...	0	0	0	0	0	0
36	0	0	0	0	0	0	...	0	0	0	0	0	0
37	0	0	0	0	0	0	...	0	0	0	0	0	0
38	2	0	0	1	3	1	...	1	1	1	1	1	1
39	0	0	0	0	0	0	...	0	0	0	0	0	0

Figure 14: Matrix for bag of words approach.

Term Frequency- Inverse Document Frequency(TF-IDF) is a statistical measurement that reflects how important a word is for a document by looking at occurences of words in a document and among other documents. TF is the measurement of how frequent that term is in a document, IDF is the measurement of how much information that word provides by looking at its occurrences among other documents. For example, if a word is common in every document, it provides less information for the classification tasks. These measurements together are especially used for data preparation processes to figure out the important and less important words in a document. Therefore, at the last step before training our Naive Bayes model, we implemented a TF-IDF matrix. Formulas of TF and IDF measurements are given below and Figure 15 is the TF-IDF matrix we acquired. In the matrix, the first column is a tuple with the first tuple being the document id and the second being the word id. The second column represents the TF-IDF value of each word for each document which was retrieved by multiplying TF and IDF values.

$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}},$$

Formula 1: TF formula where f value represents the raw count of term t in document d.

$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

Formula 2: IDF formula where N is the number of documents in the corpus divided by the number of documents where term t appears.

```
  (0, 9380)     0.02297587208802962
  (0, 2909)     0.024627513878450018
  (0, 699)      0.01862525109142657
  (0, 3626)     0.020485405282931194
  (0, 3988)     0.01271493607995438
  (0, 10135)    0.01713994665495053
  (0, 1729)     0.015354937278893574
  (0, 6897)     0.03725050218285314
  (0, 6413)     0.013094593695331858
  (0, 4103)     0.01436657787037478
  (0, 13)       0.013917852561797693
  (0, 5325)     0.011365446476863056
  (0, 3266)     0.017845404083992
  (0, 6556)     0.01713994665495053
  (0, 3796)     0.019497045874412398
  (0, 5517)     0.015354937278893574
  (0, 8753)     0.01862525109142657
  (0, 8037)     0.02873315574074956
  (0, 5740)     0.04325276496987658
  (0, 8994)     0.016495914480900674
  (0, 9587)     0.035690808167984
  (0, 8945)     0.06487914745481488
  (0, 6051)     0.019497045874412398
  (0, 10171)    0.04309973361112434
  (0, 10216)    0.014844272690480274
  :     :
  (39, 7098)    0.024608287516157356
  (39, 3341)    0.003841553787271429
```

Figure 15: TF-IDF matrix for the Naive Bayes model.

- **Word2Vec**

We will use Word2Vec for word embedding. Using Word2Vec we are able to transform words into vectors that can denote meaning. The technique utilizes the context in which the words appear and the Distributional Hypothesis [12] to derive meaning through their distributions. Assume we are examining the sentence “it was the best of times, it was the worst of times”. Iterating through each word and creating a spreadsheet for the neighbors, that is the preceding and the following words, of each word we get:

	START ___ was	it ___ the	was ___ best	the ___ of	best ___ times	of ___ it	times ___ was	was ___ worst	worst ___ times	of ___ END
it	1	0	0	0	0	0	1	0	0	0
was	0	2	0	0	0	0	0	0	0	0
the	0	0	1	0	0	0	0	1	0	0
best	0	0	0	1	0	0	0	0	0	0
of	0	0	0	0	1	0	0	0	1	0
times	0	0	0	1	0	0	0	0	0	1
worst	0	0	0	1	0	0	0	0	0	0

Figure 16: The neighbours matrix of the sentence [17].

Then the vector of the words “best” and “worst” would be [0, 0, 0, 1, 0, 0, 0, 0, 0, 0] and [0, 0, 0, 1, 0, 0, 0, 0, 0, 0] respectively. Due to the two words being used in the exact same context the resulting vectors are the same and according to the model have the same meaning. Of course, contrary to this result, the two words normally do not have the same meaning. Therefore, more data is needed to provide more context to the words and by the application of Distributional Hypothesis it is assumed that the distributional properties of these items in large enough samples basically denote the meanings associated to the items.

Applying this technique, the cosine similarity of the resulting vectors is used to determine the similarity and the relation of words. For example, assume the associated vectors for the words “king”, “queen”, “aunt” are V_{king} , V_{queen} and V_{aunt} respectively. Then let $V_x = (V_{king} - V_{queen}) + V_{aunt}$. In such a case we can see that the similarity (and the differemce) between “king” and “queen” are that they refer to the same position but one is a male noun whilst the other is a female noun. Therefore, searching for the closest vector to V_x should give us the vector of “uncle”, the male counterpart of “aunt”. Such properties of the vectors allow us to feed our model with a matrix that not only represents the words in the song but also the meaning of the lyrics.

For the purposes of this project using a previously trained Word2Vec model’s output makes more sense as it allows us to experiment with different outputs to be able to test and search for the most-suitable for our models. The python library Gensim features an implementation of Word2Vec which can either be used directly almost like a dictionary. As an example following is the word “cheese” represented as a vector with 300 data using word-to-vector algorithm:

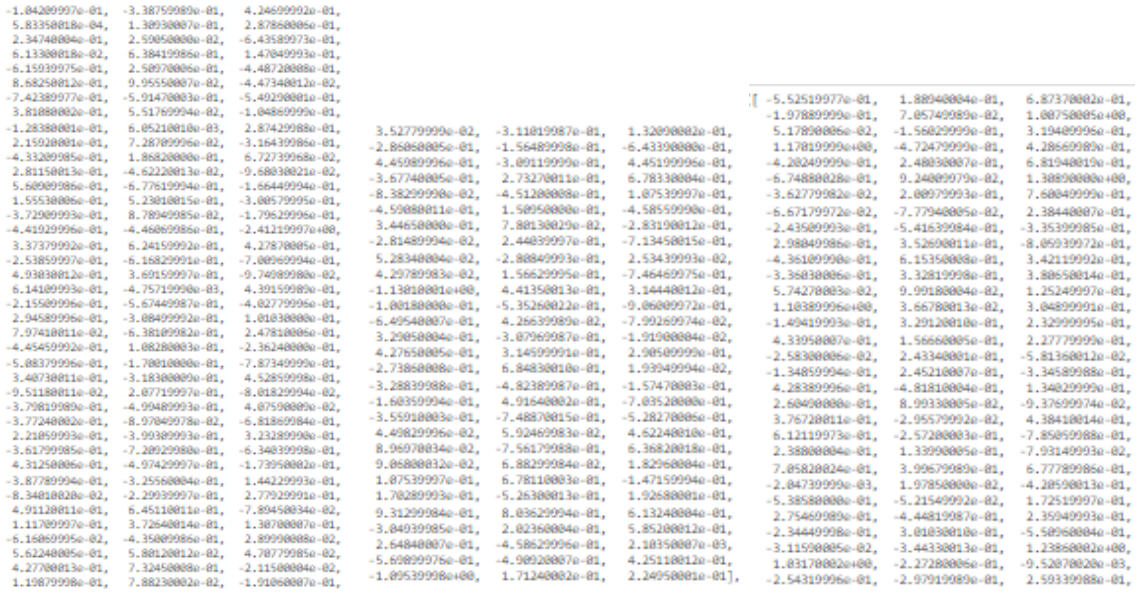


Figure 17: The distribution vector of “cheese”.

Lyrics of a song can be denoted as an NxM matrix where N stands for the number of words in the song and M stands for the length of the vector that denotes a word. Size of the vector M can be set to different values that increase the details we can learn about the word.

- **Global Vector (GloVe)**

Just like its predecessor Word2Vec, GloVe is another word representation technique. GloVe creates contextual word embeddings to be used as inputs for the classification models. The difference between Word2Vec and GloVe is that GloVe is an improved version of Word2Vec. The Word2Vec method is limited to the local information of the words in a sentence. On the other hand, GloVe recognizes both the local and the global information of the words in the vectorization process. GloVe is constructed on the idea of a co-occurrence matrix that includes the information of the words that co-occur which supports the production of word matrices that include the global statistics of words [6].

Assume that there are two documents with the following sentences:

Document 1: All that glitters is not gold.
Document 2: All is well that ends well.

The co-occurrence matrix that is constructed from these documents can be seen below.

*	<START>	all	that	glitters	is	not	gold	well	ends	<END>
<START>	0	2	0	0	0	0	0	0	0	0
all	2	0	1	0	1	0	0	0	0	0
that	0	1	0	1	0	0	0	1	1	0
glitters	0	0	1	0	1	0	0	0	0	0
is	0	1	0	1	0	1	0	1	0	0
not	0	0	0	0	1	0	1	0	0	0
gold	0	0	0	0	0	1	0	0	0	1
well	0	0	1	0	1	0	0	0	1	1
ends	0	0	1	0	0	0	0	1	0	0
<END>	0	0	0	0	0	0	1	1	0	0

Figure 18: The co-occurrence matrix of Document 1 and 2 [7].

Looking at the word “well”, we can see that this word has co-occurred with the words “that”, “is”, and “well” once and additionally, it appeared on the end of the document once.

Pennington et al. used the co-occurrence statistics of words from different documents to produce the vector representations of the words which they called GloVe and they also provided some different versions of pre-trained GloVe word vectors in different word representation dimensions and with words from different sources [6]. In our project, we will use one of these pre-trained word embeddings as training our own embeddings will take a lot of time. In the first place, we decided to use the “glove.6B.50d.txt”, which is the embeddings file with the smallest number of word representation dimensions. While having less information compared to bigger dimension vectors, this file will be less complex to calculate. When we will feed our model, our input will be the vectors of the words on the lyrics which we gathered from the pre-trained GloVe word embeddings.

Below are some examples from the song “(Ain’t That) Good News” by Sam Cooke. This song starts with the line “Oh, my baby’s”. After the preprocessing, from Figure 7, it can be seen that this line becomes “oh baby …”. In order to put these words on the model, we need their word vector representations from GloVe embeddings file “glove.6B.50d.txt”.

```
print(embeddings_index.get("oh"))
```

```
[-0.070292  1.6078    0.64854  -0.4591   -0.16151  -1.093     0.61743
  0.048792 -0.47594   1.2585   -0.52256   0.96757  -0.70143   0.31107
  0.13962   0.72396   0.21441  -0.019466  0.40694   0.94655  -0.89237
  0.30974   1.8434    0.54281   0.60901  -1.867    -1.9405    0.71482
 -0.090765 -1.5403    1.287     0.79188  -0.069779  1.3083    0.54165
 -0.94769   0.90328   0.18304   0.87004   0.46736  -0.32235   0.69321
 -0.25275  -0.17076   0.52085   0.30456  -0.47081  -0.64507   0.49646
  0.71087 ]
```

Figure 19: GloVe word vector of the word “oh”, obtained from the word embeddings file “glove.6B.50d.txt”.

```
print(embeddings_index.get("baby"))
```

```
[ 0.54936   0.22994  -0.035731 -0.91432   0.70442   1.3736   -0.99369
 -0.50342   0.5793    0.34814   0.23851   0.54439   0.34322   0.57407
  1.3732    0.46358  -0.72877   0.28868   0.10006  -0.2302   -0.12893
  0.7033    0.39612   0.26045   0.26971  -1.3036   -0.93774   0.27053
  0.60701  -0.66894   1.9709    0.6796   -0.69439   1.038     0.51364
  0.23022   0.36456  -0.30902   1.1395   -1.1466   -0.78887   0.054432
 -0.069112 -0.24386   1.4049    0.091876  0.23131  -1.3028    0.3246
  0.10741 ]
```

Figure 20: GloVe word vector of the word “baby”, obtained from the word embeddings file “glove.6B.50d.txt”.

From the pre-trained GloVe word embeddings file “glove.6B.50d.txt”, we can obtain 50-dimensioned word vectors to be used as inputs for our classification models.

4. Remaining Tasks

Remaining tasks to finalise this report and research requires the listed points below to be completed. More problems can occur outside of what is currently envisioned. Those cases will be explored in the next report.

- The complete lyrics dataset will be preprocessed using the same scripts written for sample dataset.
- The RNN architecture and Multinomial Naive Bayes classifier will be formed.
- The models will be trained with the preprocessed dataset.
- Hyperparameter tuning will be performed to achieve best results on the RNN model.
- The results will be documented and analysed.

5. Division of Work

WP	Work Package	Members Involved
WP1	Background Research	All Members
WP2	Data Retrieval and Preprocessing	Cansu Moran, Elif Kurtay
WP3	Reports	All Members
WP4	Word2Vec	Atakan Dönmez
WP5	Global Vector	Öykü Hatipoğlu
WP6	Bag of Words with TF-IDF	Elif Kurtay, Gamze Güliter
WP7	Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM)	Atakan Dönmez, Öykü Hatipoğlu, Cansu Moran
WP8	Multinomial Naive Bayes	Elif Kurtay, Gamze Güliter
WP9	Hyperparameter Optimization for RNN	Cansu Moran
WP10	Result Analysis	All Members

Table 1: Work packages and members involved in each package

Table 1 displays our division of work into work packages and the responsible members for each package. **WP1** includes making a research on similar previous work and learning tools that will be used for the project. **WP2** includes data preprocessing and formation of the dataset into including the information required to start text vectorization. Some of these activities are the retrieval of song lyrics from AZLyrics, deleting non English or non verbal songs, and determining the period of the songs. **WP3** includes writing the project reports. **WP4**, **WP5, WP6** all belong to the text vectorization process where each word in string lyrical data is transformed into numerical representations in different ways. **WP7** and **WP8** are the two different models to be used for classification. The Deep Learning model will utilize Word2Vec, Global Vector(GloVe) and Bag-of-Words(BoW) with TF-IDF word vectorization techniques while Multinomial Naive Bayes will only utilize Bag-of-Words with TF-IDF. In **WP9**, hyperparameter optimization will be performed on RNN. The **WP10** package will include a detailed analysis of the results obtained from different models using different word vectorization techniques.

6. References

[1] Varnum MEW, Krems JA, Morris C, Wormley A, Grossmann I (2021) Why are song lyrics becoming simpler? a time series analysis of lyrical complexity in six decades of American popular music. PLoS ONE 16(1): e0244576. https://doi.org/10.1371/journal.pone.0244576.

[2] McKay, Cory; Burgoyne, John A.; Hockman, Jason; Smith, Jordan BL; Vigliensoni, Gabriel; and Fujinaga; Ichiro. "Evaluating the genre classification performance of lyrical features relative to audio, symbolic and cultural features." ISMIR, pages 213–218, 2010.

[3] Tsaptsinos, Alexandros. "Music Genre Classification by Lyrics using a Hierarchical Attention Network" ICME, Stanford University. [Accessed in: 26-Nov-2021].

[4] Mayer, Rudolf; Rauber, Andreas. "Musical genre classification by ensembles of audio and lyrics features." In Proceedings of International Conference on Music Information Retrieval, pages 675–680, 2011.

[5] Mayer, Rudolf; Neumayer, Robert; and Rauber, Andreas. "Combination of audio and lyrics features for genre classification in digital audio collections." Proceedings of the 16th ACM international conference on Multimedia, pages 159–168. ACM, 2008.

[6] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[7] A. Jha, "Vectorization techniques in NLP [guide]," *Vectorization Techniques in NLP*, 20-Jul-2021.

[8] Saurabhk, "Word embedding: New age text vectorization in NLP," *Medium*, 16-Jan-2021. [Online]. Available: https://medium.com/swlh/word-embedding-new-age-text-vectorization-in-nlp-3a2db1db2f5b. [Accessed: 28-Nov-2021].

[9] "A Beginner's Guide to Bag of Words & TF-IDF." Pathmind, https://wiki.pathmind.com/bagofwords-tf-idf.

[10] E. Negi, "Spotify-data 1921-2020," *Kaggle*, 29-Aug-2020. [Online]. Available: https://www.kaggle.com/ektanegi/spotifydata-19212020. [Accessed: 30-Oct-2021].

[11] "Langdetect." *PyPI*, [Online]. Available: https://pypi.org/project/langdetect/. [Accessed: 30-Oct-2021].

[12] J. W. Miller, "Scraping song lyrics from genius.com 🎶," 21-Aug-2017. [Online]. Available: https://www.johnwmillr.com/scraping-genius-lyrics/. [Accessed: 27-Nov-2021].

[13] elmoiv, “Azapi,” *PyPI*, 12-Feb-2021. [Online]. Available: https://pypi.org/project/azapi/. [Accessed: 30-Oct-2021].

[14] Don Ralke, “Fascinating Rhythm,” *But You’ve Never Heard Gershwin with Bongos*. Warner Records Inc, 1960. [Sound recording]. Available: https://open.spotify.com/track/6NTceHSsE8bSWBE0hEVUUl?si=34fd786e3c174ce8.

[15] K. Tiffany, “You can now play with Spotify's recommendation algorithm in your browser,” *The Verge*, 05-Feb-2018. [Online]. Available: https://www.theverge.com/tldr/2018/2/5/16974194/spotify-recommendation-algorithm-playlist-hack-nelson. [Accessed: 28-Nov-2021].

[16] “Re - Regular Expression Operations.” Re - Regular Expression Operations - Python 3.10.0 Documentation, [Online]. Available: https://docs.python.org/3/library/re.html. [Accessed: 28-Nov-2021].

[17] NLTK, [Online]. Available: https://www.nltk.org/. [Accessed: 28-Nov-2021].

[18] Sahlgren, Magnus (2008). "The Distributional Hypothesis" *Rivista di Linguistica*. 20 (1): 33–53. [Online]. Available: https://neptune.ai/blog/vectorization-techniques-in-nlp-guide. [Accessed: 26-Nov-2021].

[19] A. Parrish, “Understanding word vectors: A tutorial for ‘Reading and writing electronic text," A class I teach at ITP. (python 2.7) code examples released under CC0 https://creativecommons.org/choose/zero/, other text released under CC by 4.0 Https://creativecommons.org/licenses/by/4.0/,” *Gist*. [Online]. Available: https://gist.github.com/aparrish/2f562e3737544cf29aaf1af30362f469. [Accessed: 28-Nov-2021].