

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**  
**NÚCLEO DE EDUCAÇÃO A DISTÂNCIA**  
**Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data**

**Gabriel Machado de Souza**

**SISTEMA DE RECOMENDAÇÃO DE MÚSICA**

Belo Horizonte  
2020

**Gabriel Machado de Souza**

## **SISTEMA DE RECOMENDAÇÃO DE MÚSICA**

Trabalho de Conclusão de Curso apresentado  
ao Curso de Especialização em Ciência de  
Dados e Big Data como requisito parcial à  
obtenção do título de especialista.

Belo Horizonte  
2020

## SUMÁRIO

1. Introdução.....	4
1.1. Contextualização .....	4
1.2. O Problema Proposto.....	4
2. Coleta de Dados .....	5
3. Processamento e Tratamento de Dados .....	7
4. Análise e Exploração dos Dados .....	8
5. Código Fonte e Linguagem de Programação .....	11
6. Apresentação dos Resultados .....	17
7. Links .....	21
REFERÊNCIAS.....	22

## **1. Introdução**

### **1.1. Contextualização**

A utilização de aplicativos de streaming de mídia não é nenhuma novidade e é utilizada diariamente por grande parte da população, tanto para conteúdo de vídeo quanto áudio. As plataformas de áudio como Spotify e Apple Music revolucionaram a forma como consumimos música, causando um grande impacto em produtoras que dominavam o mercado, assim como no formato de mídia, que era algo físico como o CD e DVD, tornando-se digital no decorrer dos últimos anos.

### **1.2. O Problema Proposto**

Os aplicativos de streaming de música buscam formas de atrair usuários, seja através de conteúdo exclusivo ou diferentes planos de assinatura, procuram ainda meios de interagir dinamicamente com os mesmos. Uma das táticas utilizadas por estes aplicativos é de sugerir bandas e/ou músicas novas para o usuário, baseado no seu gosto musical.

Através de análises do que cada usuário escuta, quais seus artistas e gêneros musicais preferidos é possível gerar recomendações assertivas para cada um dos seus usuários, incentivando a descoberta e exploração de novas músicas ou artistas disponíveis no aplicativo. Desta forma, o usuário interage mais e por mais tempo com o aplicativo, elevando os índices de satisfação e utilização perante seus concorrentes, fidelizando seus usuários.

O dataset utilizado neste projeto está disponível publicamente e possui um grande número de registros sobre usuários e quais músicas os mesmos estão escutando. As informações deste dataset foram obtidas através do Twitter, onde os próprios usuários compartilharam na rede social quais as músicas que estavam escutando, possuindo cerca de um milhão de registros.

Utilizando-se desta base dados, o projeto irá realizar a análise destes registros e elaborar um sistema de recomendação de músicas para os usuários, baseado em todo o histórico extraído do dataset.

## 2. Coleta de Dados

O dataset utilizado neste trabalho está disponível publicamente, no site da universidade austríaca Johannes Kepler University Linz. O mesmo tem sua origem em um artigo apresentado durante a International Society for Music Information Retrieval (ISMIR), em novembro de 2013, na cidade de Curitiba no Brasil. O dataset pode ser encontrado no seguinte link: <http://www.cp.jku.at/datasets/MMTD/>.

Os responsáveis por criar este dataset extraíram todos os seus dados da rede social Twitter entre setembro de 2011 e abril de 2013, utilizando API de streaming disponibilizada pela rede social, mais detalhes sobre a API podem ser obtidos no seguinte link: <https://developer.twitter.com/en/docs/api-reference-index>. No presente trabalho de conclusão, optei por usar três arquivos deste dataset, de um total de sete arquivos de dados, visando o objetivo do mesmo. Os arquivos utilizados estão detalhados a seguir.

### a. Artists.txt

Neste arquivo estão as informações de artistas no total de 523.479 registros, com as seguintes colunas:

Coluna	Descrição	Tipo
artist_id	Identificador do artista	Inteiro
artist_mbid	Identificador do artista na enciclopédia de música Music Brainz ( <a href="https://musicbrainz.org">https://musicbrainz.org</a> )	Texto
artist_name	Nome do artista	Texto

### b. Track.txt

Arquivo contendo as informações sobre as músicas e o artista de cada uma delas, total de 134.199 registros. Colunas do arquivo:

Coluna	Descrição	Tipo
track_id	Identificador da música	Inteiro
track_title	Título/nome da música	Texto
track_artistId	Identificador do artista (Referência ao arquivo artists.txt)	Inteiro

### c. Tweet.txt

Arquivo com todos os tweets a serem analisados no trabalho de conclusão, com um total 1.090.726 registros.

Coluna	Descrição	Tipo
tweet_id	Identificador do tweet	Inteiro
tweet_tweetId	Identificador do registro no Twitter	Inteiro
tweet_userId	Identificador do usuário	Inteiro
tweet_artistId	Identificador do artista (Referência ao arquivo artists.txt)	Inteiro
tweet_trackId	Identificador da música (Referência ao arquivo track.txt)	Inteiro
tweet_datetime	Data e hora do tweet	Data
tweet_weekday	Dia semana do tweet	Inteiro
tweet_longitude	Longitude da origem do tweet	Numérico
tweet_latitude	Latitude da origem do tweet	Numérico

### 3. Processamento e Tratamento de Dados

Os dados disponibilizados neste dataset já possuem uma excelente qualidade de informações, já sendo tratados casos como registros sem dados. Todos os registros contidos no arquivo `tweet.txt` possuem todas as suas colunas informadas com dados válidos, assim como todos os registros que fazem referência, tanto para o arquivo de artistas quanto de músicas, estão íntegros fazendo a referência correta entre os arquivos.

Algumas colunas dos arquivos não foram utilizadas para nenhum processamento neste trabalho, portanto os dataframes possuem a exclusão dessas colunas. No dataframe do arquivo artistas, a coluna `artist_mbid` foi removida.

Para o dataframe de tweets, as seguintes colunas foram removidas: `tweet_weekday`, `tweet_longitude`, `tweet_latitude`, `tweet_datetime`. Neste mesmo dataframe foi necessário criar uma nova coluna, um contador iniciado com o valor 1, para todos os registros do dataframe. A nova coluna criada, chamada de `tweet_count`, será utilizada no processamento das informações, para auxiliar na contagem e agrupamento dos registros realizado nas etapas seguintes.

Uma função auxiliar foi criada para obter todas as músicas (campo `tweet_trackId`) que um usuário específico publicou, realizando a busca no arquivo de tweets. A função, chamada `get_all_tracks_by_user` retorna uma lista, com todos os ID's de músicas encontrados para o usuário específico, removendo ainda as duplicações, caso existam.

A função `get_all_users_by_track` possui como parâmetro o ID de uma determinada música, realizando a busca no dataset de tweets por todos os usuários que publicaram um registro sobre a música. A função retorna uma lista com todos os ID de usuário encontrados, removendo duplicações nos casos onde o usuário publicou mais de uma vez estar escutando a música filtrada.

#### 4. Análise e Exploração dos Dados

O objetivo deste trabalho é conseguir sugerir músicas para um determinado usuário, de acordo com seu histórico de reproduções de músicas. Para alcançar este objetivo, um dos primeiros passos é compreender os dados com os quais estamos trabalhando. Os dados dos três arquivos que compõem este dataset são carregados para dataframes, da biblioteca pandas do Python, para serem processados e analisados.

A abordagem de recomendação que será desenvolvida neste trabalho é uma abordagem híbrida, utilizando como base a recomendação colaborativa, realizada através da semelhança entre perfis de usuário. A figura 1 exemplifica este conceito de recomendação, onde é gerada a recomendação baseado no que outros usuários com gostos similares ao usuário alvo também escutaram.

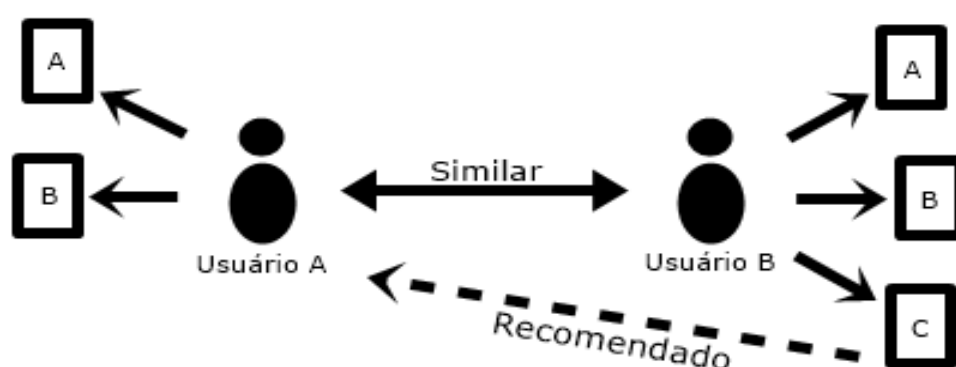


Figura 1 Exemplo de recomendação colaborativa

Como nossos dados são limitados para tentar traçar o perfil de usuário e como não possuímos nenhuma informação de avaliação e/ou nota sobre as determinadas músicas fornecida pelos usuários, de acordo com o seu gosto musical optou-se por criar uma abordagem própria para gerar as recomendações. Com os dados que temos disponíveis, a estratégia de recomendação que iremos abordar será de:

- Buscar por todas as músicas que o usuário alvo já escutou;
- Buscar por todos os usuários que escutaram as mesmas músicas;
- Gerar a recomendação de músicas para o usuário alvo, baseado no que os usuários semelhantes já escutaram.



Os conjuntos de dados neste dataset são de um volume considerável, totalizando um total de 1.090.726 tweets (linhas) no arquivo tweet.txt, além de 118.368 usuários distintos e 67.754 músicas distintas no total a serem analisados. Para contornar um dos problemas encontrados nos sistemas de recomendação, o cold-start, onde não possuímos dados do usuário, optou-se por criar uma análise de quais são as músicas mais populares entre todos os usuários. Desta forma, quando a recomendação estiver sendo feita para um usuário novo da plataforma, serão recomendadas as músicas mais populares para o mesmo. A classe chamada pop\_songs é a responsável por fazer essa análise, contabilizando e agrupando todos os registros através do identificador da música, gerando um conjunto de dados com as músicas mais populares. Nos casos em que existem dados do usuário em nossa base de dados, foi criada a classe chamada similar\_songs para realizar o processamento das informações.

O sistema de recomendação de músicas possui algumas etapas de processamento, na etapa inicial são extraídas todas as músicas que o usuário já escutou, no nosso caso, todas as músicas que o usuário fez um tweet que escutou. Neste conjunto de dados são removidos os registros duplicados encontrados, esse processamento está na função `get_all_tracks_by_user`.

A segunda etapa de processamento realiza a busca por todos os outros usuários, que escutaram as mesmas músicas que o usuário ao qual iremos fazer a recomendação. O processamento e filtro destes dados é realizado pela função `get_all_users_by_track`, removendo as duplicidades de identificadores do usuário.

Na terceira etapa, o processamento é realizado a partir da lista de usuários similares, gerada na etapa anterior. Para cada um destes usuários, similares ao usuário alvo da recomendação, são extraídos todos os registros encontrados na base dados, com os tweets e identificadores de músicas que cada um deles publicou. Ao fim deste processo, obtemos uma nova base de dados com todos os registros de usuários similares ao usuário ao qual vamos fazer a recomendação de músicas.

A etapa final de processamento computa todos os registros filtrados neste novo conjunto de dados, removendo todas as músicas que o usuário alvo já escutou, evitando de recomendar músicas já escutadas pelo mesmo. O conjunto de dados é sumarizado e agrupado pelo identificador de música, gerando uma lista de músicas mais escutadas pelos usuários similares, além de realizar a limpeza de colunas que não estão sendo utilizadas neste momento. Os resultados são retornados em uma

lista, ordenada por ordem decrescente, com as músicas que mais foram contabilizadas primeiro, gerando a recomendação de músicas para o usuário escutar.

## 5. Código Fonte e Linguagem de Programação

A linguagem escolhida para ser desenvolvido o presente trabalho foi o Python, por ser uma das linguagens mais populares no universo de Big Data, além da sua facilidade e extensibilidade, podendo ser utilizada em diversas plataformas disponíveis no mercado. Em conjunto, foi optado por trabalhar no ambiente do Google Colaboratory, vulgo Colab, que é uma plataforma na nuvem e gratuita, sendo possível rodar códigos Python, assim como criar Colab notebooks, bastante similar ao Jupyter notebook.

O recurso da linguagem Python que mais foi utilizado para o sistema de recomendação foi a biblioteca Pandas e sua estrutura de dataframes. Os dataframes são estruturas bidimensionais de dados, como em uma planilha, com diversos recursos para tratamento, sumarização e análise de dados. A seguir está todo o código fonte utilizado durante o desenvolvimento do trabalho, no formato Python.

```
import pandas as pd
```

```
!curl --remote-name \
  -H 'Accept: application/vnd.github.v3.raw' \
  --location
https://raw.githubusercontent.com/gkbelo/tcc_code/master/mmttd/artists.txt
```

```
!curl --remote-name \
  -H 'Accept: application/vnd.github.v3.raw' \
  --location
https://raw.githubusercontent.com/gkbelo/tcc_code/master/mmttd/track.txt
```

```
!curl --remote-name \
  -H 'Accept: application/vnd.github.v3.raw' \
  --location
https://raw.githubusercontent.com/gkbelo/tcc_code/master/mmttd/tweet.txt
```

```
def read_artists():
    cols = ['artist_id', 'artist_mbid', 'artist_name']
```

```

df = pd.read_csv('artists.txt', sep='\t', names=cols)
df = df.drop(['artist_mbid'], axis=1)
return df

```

```

def read_tracks():
    cols = ['track_id', 'track_title', 'track_artistId']
    df = pd.read_csv('track.txt', sep='\t', names=cols)
    return df

```

```

def read_tweets():
    cols = ['tweet_id', 'tweet_tweetId', 'tweet_userId', 'tweet_artistId', 'tweet_trackId',
'tweet_datetime',
            'tweet_weekday', 'tweet_longitude', 'tweet_latitude']
    df = pd.read_csv('tweet.txt', sep='\t', names=cols)
    df = df.drop(['tweet_weekday', 'tweet_longitude', 'tweet_latitude', 'tweet_datetime'],
axis=1)
    df['tweet_count'] = 1
    return df

```

```

def get_all_tracks_by_user(tweet_df, user_id):
    ids = tweet_df.drop(['tweet_id', 'tweet_tweetId', 'tweet_artistId', 'tweet_count'],
axis=1)
    ids = ids.loc[ids['tweet_userId'] == user_id]
    ids.drop_duplicates(subset="tweet_trackId",
                        keep = False,
                        inplace = True)
    return ids['tweet_trackId'].tolist()

```

```

def get_all_users_by_track(tweet_df, track_id):
    ids = tweet_df.drop(['tweet_id', 'tweet_tweetId', 'tweet_artistId', 'tweet_count'],
axis=1)
    ids = ids.loc[ids['tweet_trackId'] == track_id]
    ids.drop_duplicates(subset="tweet_userId",
                        keep = False,

```

```

        inplace = True)
    return ids['tweet_userId'].tolist()

class pop_songs:
    def __init__(self, tweets_df, tracks_df, artists_df):
        self.tweets_df = tweets_df
        self.tracks_df = tracks_df
        self.artists_df = artists_df

    def top_songs(self, topn=10):
        # Create the dataset with the most popular songs
        top_df = self.tweets_df.groupby('tweet_trackId')['tweet_count'].sum().sort_values(ascending=False).reset_index()
        top_df = top_df.rename(columns={"tweet_trackId": "track_id"})
        top_df = pd.merge(top_df, self.tracks_df, left_on='track_id', right_on='track_id',
                           how='left')
        top_df = pd.merge(top_df, self.artists_df, left_on='track_artistId',
                           right_on='artist_id', how='left')
        top_df = top_df.drop(['artist_id', 'track_artistId', 'tweet_count', 'track_id'], axis=1)
        return top_df.head(topn)

    def print_top5(self):
        print('-- Top 5 songs --')
        print(self.top_songs(5))
        print('--')

class similar_songs:
    def __init__(self, tweets_df, tracks_df, artists_df, user_tracks):
        self.tweets_df = tweets_df
        self.tracks_df = tracks_df
        self.artists_df = artists_df

```

```

self.user_tracks = user_tracks

def recommend(self, ignore_tracks=[], topn=10, ignore_user_id=0):
    users_all = self.tweets_df.drop(['tweet_id', 'tweet_tweetId', 'tweet_artistId'], axis=1)
    users_filtered_df = pd.DataFrame()

    for idx_t in range(len(self.user_tracks)):
        # get all users that tweeted the same songs that the target user
        users_id_by_track = get_all_users_by_track(tweets, self.user_tracks[idx_t])
        if ignore_user_id > 0:
            users_id_by_track.remove(ignore_user_id)

        for idx_u in range(len(users_id_by_track)):
            # get all the tracks for each user
            tmp = users_all.loc[users_all['tweet_userId'] == users_id_by_track[idx_u]]
            users_filtered_df = pd.concat([tmp, users_filtered_df], ignore_index=True)

    recom_df = users_filtered_df[~users_filtered_df['tweet_trackId'].isin(ignore_tracks)] \
        .sort_values('tweet_count', ascending=False)
    recom_df = recom_df.groupby('tweet_trackId')['tweet_count'].sum().sort_values(ascending=False) \
        .reset_index()
    recom_df = recom_df.rename(columns={"tweet_trackId": "track_id"})
    recom_df = pd.merge(recom_df, self.tracks_df, left_on='track_id',
        right_on='track_id', how='left')
    recom_df = pd.merge(recom_df, self.artists_df, left_on='track_artistId',
        right_on='artist_id', how='left')
    recom_df = recom_df.drop(['artist_id', 'track_artistId', 'tweet_count', 'track_id'],
        axis=1)
    return recom_df.head(topn)

```

"""Read the dataset files"""

```

artists = read_artists()
tracks = read_tracks()
tweets = read_tweets()

"""Exploring the dataset"""

print('Tweets numbers')
print('# lines: ' + str(tweets.shape[0]))

total_users = read_tweets()
total_users.drop_duplicates(subset="tweet_userId",
                           keep = False,
                           inplace = True)
print('Total of unique users: ' + str(total_users.shape[0]))

total_tracks = read_tweets()
total_tracks.drop_duplicates(subset="tweet_trackId",
                            keep = False,
                            inplace = True)
print('Total of unique songs: ' + str(total_tracks.shape[0]))

"""** Popularity model **
Simple analysis with Top 5 songs
"""

pop_model = pop_songs(tweets, tracks, artists)
pop_model.print_top5()

"""Get the user id to start the process"""

# example users
# 265101134 (14 tweets)
# 58937384 (854 tweets)

```

```

# 92235951 (75 tweets)
# 250253081 (2 tweets)
#
# new user
# 43254
#
test_user = 58937384
#
tot_user_rows = read_tweets()
tot_user_rows = tot_user_rows.loc[tot_user_rows['tweet_userId'] == test_user]
print('-- User tweets count --')
print(str(tot_user_rows.shape[0]))

"""** Recommendation **
Check the data for the target user
"""

# All songs tweeted by the target user
user_tracks = get_all_tracks_by_user(tweets, test_user)

# It's a new user then recommend the Top Songs
if len(user_tracks) == 0:
    print('-- Popular song recommendation --')
    print(pop_model.top_songs(topn=10))
else:
    # If not a new user then recommend similar songs
    sim_song = similar_songs(tweets, tracks, artists, user_tracks)
    print('-- Song recommendation --')
    print(sim_song.recommend(ignore_tracks=user_tracks, ignore_user_id=test_user))

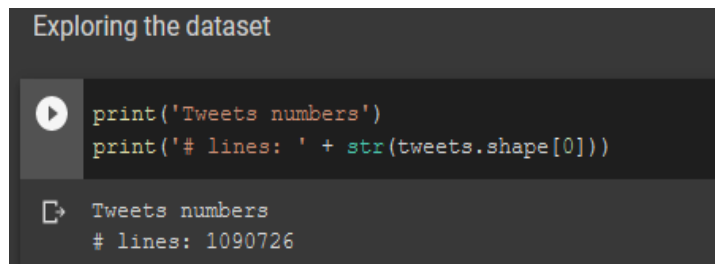
```



## 6. Apresentação dos Resultados

O Python mostrou-se uma linguagem de programação bastante eficaz no processamento de grandes volumes de dados, como neste dataset de mais 1 milhão de registros, onde em poucos segundos conseguimos extrair informações e gerar insights para nosso objetivo. A seguir, iremos detalhar alguns pontos da execução, os resultados que o sistema de recomendação gerou em alguns testes realizados, assim como o tempo de processamento do código em questão.

A alternativa proposta para burlar problemas de cold-start, em nosso sistema de recomendação, foi de criar uma recomendação das músicas mais populares dentro de todos os registros do dataset. O total de registros no dataset está detalhado na figura 2.



```

Exploring the dataset

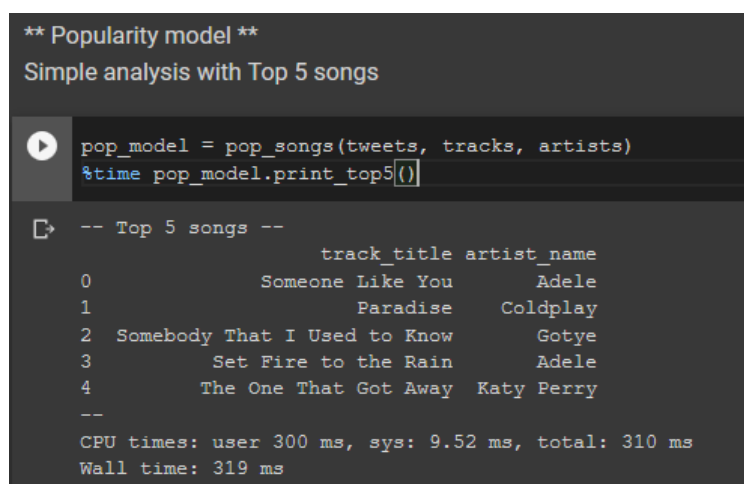
print('Tweets numbers')
print('# lines: ' + str(tweets.shape[0]))

Tweets numbers
# lines: 1090726

```

Figura 2 Total de registros

O resultado de quais são as músicas mais populares no dataset está na figura 3, levando menos de 1 segundo para fazer a contagem de todos os registros, retornando os 5 registros mais populares. Nos casos onde não temos nenhum dado do usuário em nossa base de dados, as músicas mais populares serão as recomendações apresentadas como sugestão para o mesmo.



```

** Popularity model **
Simple analysis with Top 5 songs

pop_model = pop_songs(tweets, tracks, artists)
%time pop_model.print_top5()

-- Top 5 songs --
      track_title  artist_name
0    Someone Like You      Adele
1         Paradise    Coldplay
2  Somebody That I Used to Know    Gotye
3    Set Fire to the Rain      Adele
4    The One That Got Away  Katy Perry
--
CPU times: user 300 ms, sys: 9.52 ms, total: 310 ms
Wall time: 319 ms

```

Figura 3 Top 5 músicas mais populares

Para usuários em que já temos dados suficientes, em nossa base de dados, conseguimos gerar uma recomendação de músicas baseada em seu histórico e no histórico de outros usuários, que escutaram as mesmas músicas que o usuário alvo da recomendação. Na figura 4, temos um caso onde o usuário alvo da recomendação possui 854 registros (tweets) de músicas que ele escutou, em nossa base de dados. O processamento e resultado da recomendação para este usuário, durou cerca de 5 segundos no total, conforme a imagem.

```
[17] test_user = 58937384
#
tot_user_rows = read_tweets()
tot_user_rows = tot_user_rows.loc[tot_user_rows['tweet_userId'] == test_user]
print('-- User tweets count --')
print(str(tot_user_rows.shape[0]))

-- User tweets count --
854

** Recommendation **
Check the data for the target user

[18] # All songs tweeted by the target user
user_tracks = get_all_tracks_by_user(tweets, test_user)

[19] # It's a new user then recommend the Top Songs
if len(user_tracks) == 0:
    print('-- Popular song recommendation --')
    print(pop_model.top_songs(topn=10))
else:
    # If not a new user then recommend similar songs
    sim_song = similar_songs(tweets, tracks, artists, user_tracks)
    print('-- Song recommendation --')
    %time print(sim_song.recommend(ignore_tracks=user_tracks, ignore_user_id=test_user))

-- Song recommendation --
      track_title      artist_name
0  Somebody That I Used to Know      Gotye
1             Drive By          Train
2             Domino      Jessie J
3  We Take Care of Our Own  Bruce Springsteen
4      Charlie Brown      Coldplay
5      Next to Me      Emeli Sande
6  My Kind of Love      Emeli Sande
7  Silenced by the Night      Keane
8  Through the Night      Ren Harvieu
9      Black Heart      Stooshe
CPU times: user 5.22 s, sys: 94.3 ms, total: 5.32 s
Wall time: 5.32 s
```

Figura 4 Recomendação usuário ID 58937384

Outro exemplo do sistema recomendação sendo executado está na figura 5, desta vez para o usuário de identificador número 92235951. Para este caso, temos um total de 75 tweets encontrados com músicas que o usuário escutou, o processamento e

resultado da recomendação de músicas para este usuário durou cerca de 36 segundos.

```
[25] test_user = 92235951
#
tot_user_rows = read_tweets()
tot_user_rows = tot_user_rows.loc[tot_user_rows['tweet_userId'] == test_user]
print('-- User tweets count --')
print(str(tot_user_rows.shape[0]))

-- User tweets count --
75

** Recommendation **
Check the data for the target user

[26] # All songs tweeted by the target user
user_tracks = get_all_tracks_by_user(tweets, test_user)

[27] # It's a new user then recommend the Top Songs
if len(user_tracks) == 0:
    print('-- Popular song recommendation --')
    print(pop_model.top_songs(topn=10))
else:
    # If not a new user then recommend similar songs
    sim_song = similar_songs(tweets, tracks, artists, user_tracks)
    print('-- Song recommendation --')
    %time print(sim_song.recommend(ignore_tracks=user_tracks, ignore_user_id=test_user))

-- Song recommendation --
      track_title      artist_name
0  Somebody That I Used to Know      Gotye
1              Levels      Avicci
2    The One That Got Away    Katy Perry
3    Someone Like You      Adele
4    Call Me Maybe    Carly Rae Jepsen
5              Echt    Glasperlenspiel
6      Ma Cherie    The Beat Shakers
7    Heart Skips a Beat      Olly Murs
8    Jar of Hearts    Christina Perri
9      Ich bin ich    Glasperlenspiel
CPU times: user 35.1 s, sys: 900 ms, total: 36 s
Wall time: 36 s
```

Figura 5 Recomendação usuário ID 92235951

Com este sistema, podemos influenciar os usuários que utilizam uma plataforma de streaming de músicas a explorar mais músicas e engajá-los na utilização da mesma. De forma simples, foi possível realizar uma análise em um grande volume de dados, utilizando as informações do dataset para conseguir gerar novas recomendações para os usuários. O objetivo proposto neste trabalho, era de gerar recomendações de músicas para usuários, simulando como se fosse na utilização de uma plataforma de streaming e com a codificação criada, conseguimos gerar dois tipos diferentes de

recomendações, uma baseada na popularidade das músicas e outra através do histórico de “plays” disponível em nossa fonte de dados.

## 7. Links

Abaixo estão detalhados os links para os artefatos gerados neste trabalho.

Repositório com o código fonte e documento:

[https://github.com/gkbelo/tcc\\_code](https://github.com/gkbelo/tcc_code)

Vídeo de apresentação:

[https://youtu.be/EJBaQ\\_gr-Lw](https://youtu.be/EJBaQ_gr-Lw)

## REFERÊNCIAS

Hauger, D. and Schedl, M. and Košir, A. and Tkalčič, M. **The Million Musical Tweets Dataset - What We Can Learn From Microblogs**  
Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR 2013), Curitiba, Brazil, November 2013.