THE MILLION SONG RECOMMENDATION SYSTEM

GROUP 1

- JIE LU
- CHAOYING BAO
- YIHAO LI
- TRAM NGOC LE

Executive Summary

We aim to build a useable and comprehensive **recommendation system** for music recommendation.

Based on the **Million Song Dataset** and **musiXmatch dataset**, which include user listening history, song metadata, artist, artist similarity, and lyrics.

Compared and combined **popularity based, collaborative filtering based,** and **content-based methods** to build a recommending strategy for different scenarios and different users

The whole system was built utilizing **python** and **pyspark** on **Google Cloud Platform.**

Business Problem

- An increasing number of online companies are utilizing recommendation systems to increase user interaction and enrich business potential.
- The potential benefits of a state of art recommender system:
  - ✓ Improve **user retention**
  - ✓ Improve **user engagement**
  - ✓ Understand changing trend of the **customers' tastes**
- We want to focus on the streaming music industry and develop an industry level music recommendation system under different scenarios and for different users.

Data Description

The Echo Nest, 2011

Size: 280GB (Subset)
1,000,000 unique tracks ID
Song Metadata, Artist similarity, Artist tags
SQLite, Text Files
Main Dataset Link

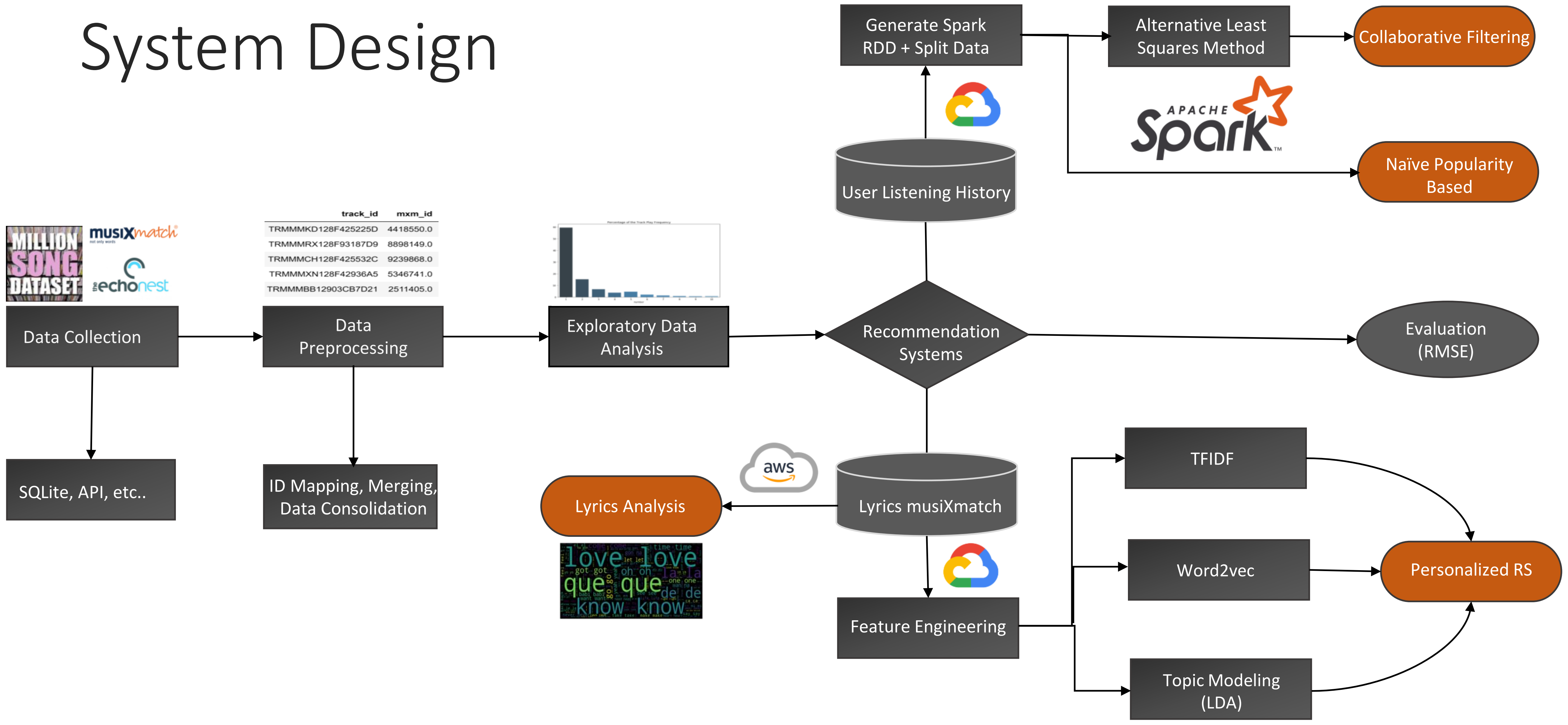
musiXmatch

Size: 70MB
779K matches between of musiXmatch ID & MSD ID
210,519 BOW for training & 27,143 BOW for testing
SQLite & Text Files
Complementary Dataset Link

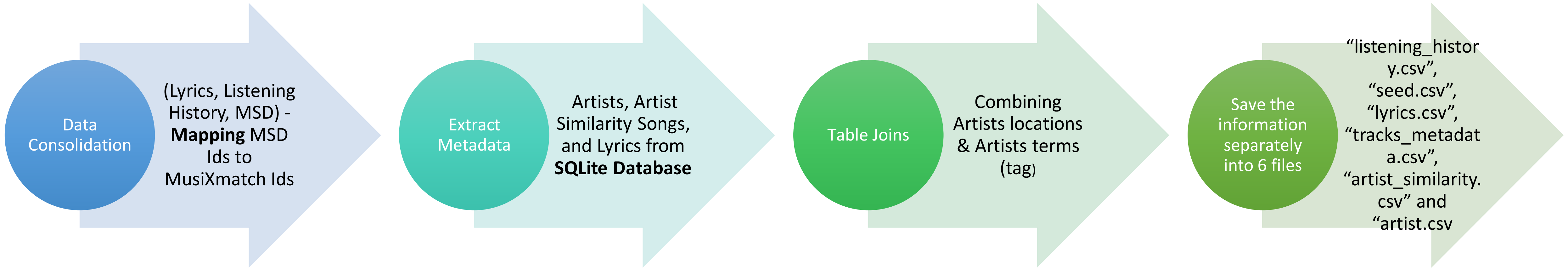
Taste Profile

Size: 488MB
48M user-song-play count triplets
1M unique users
380K unique songs
Tab-delimited
Complementary Dataset Link

System Design



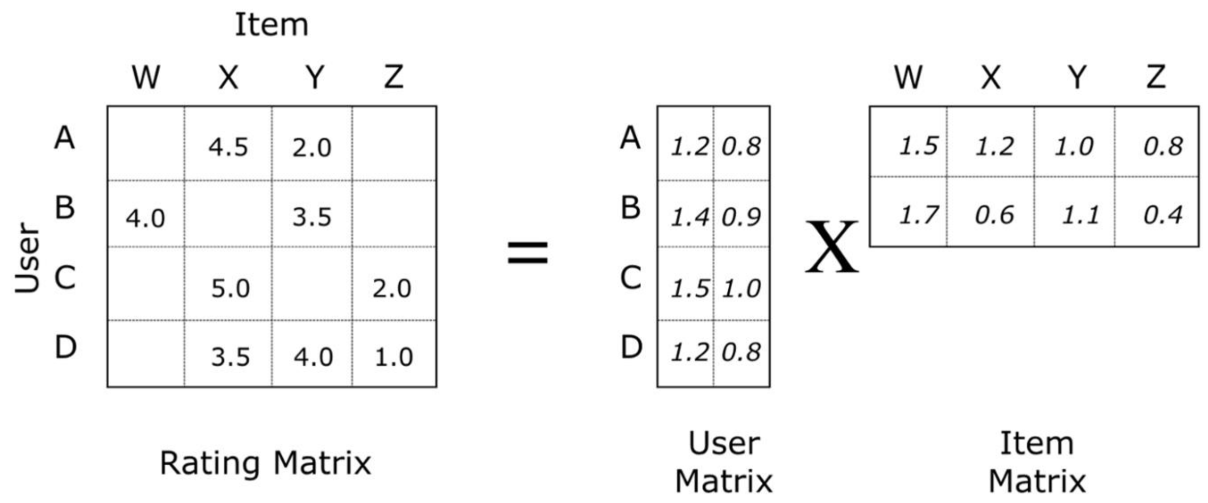
	track_id	mxm_id
	TRMMMKD128F425225D	4418550.0
	TRMMMRX128F93187D9	8898149.0
	TRMMMCH128F425532C	9239868.0
	TRMMMXN128F42936A5	5346741.0
	TRMMMBB12903CB7D21	2511405.0

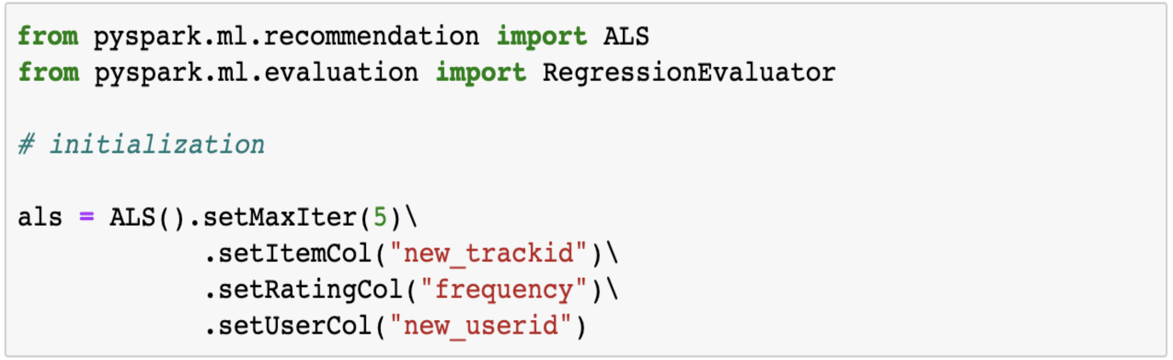

Data Consolidation
(Lyrics, Listening History, MSD) - **Mapping** MSD Ids to MusiXmatch Ids
Extract Metadata
Artists, Artist Similarity Songs, and Lyrics from **SQLite Database**
Table Joins
Combining Artists locations & Artists terms (tag)
Save the information separately into 6 files
“listening_history.csv”, “seed.csv”, “lyrics.csv”, “tracks_metadata.csv”, “artist_similarity.csv” and “artist.csv

Data Preprocessing

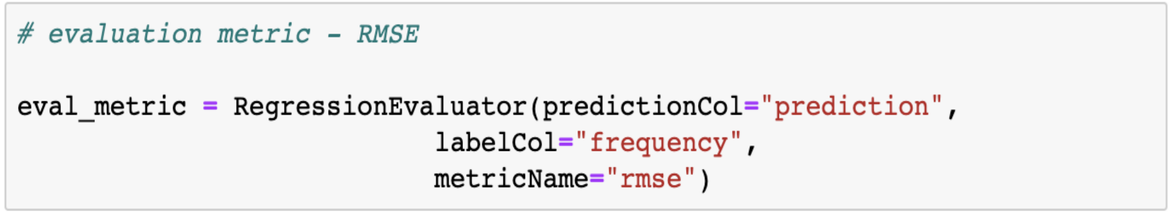
Algorithm Development | Collaborative Filtering

- Popularity Based
  - ❖ TOP 10 frequently listened songs
- ALS

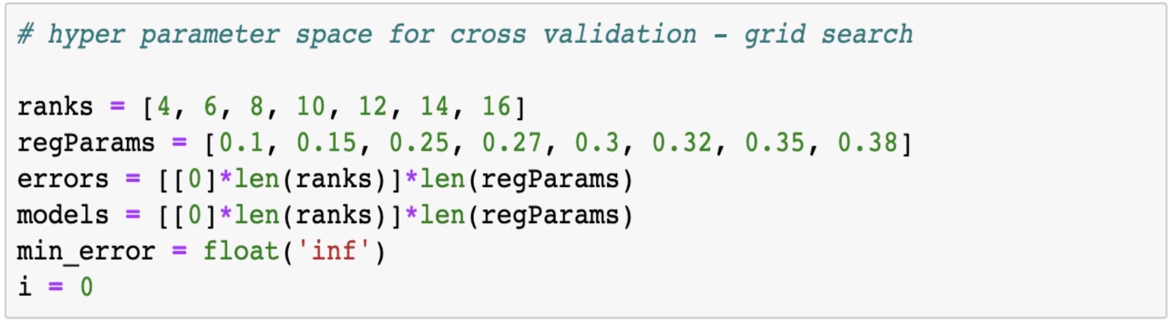




```
from pyspark.ml.recommendation import ALS
from pyspark.ml.evaluation import RegressionEvaluator

# initialization

als = ALS().setMaxIter(5)\
            .setItemCol("new_trackid")\
            .setRatingCol("frequency")\
            .setUserCol("new_userid")
```

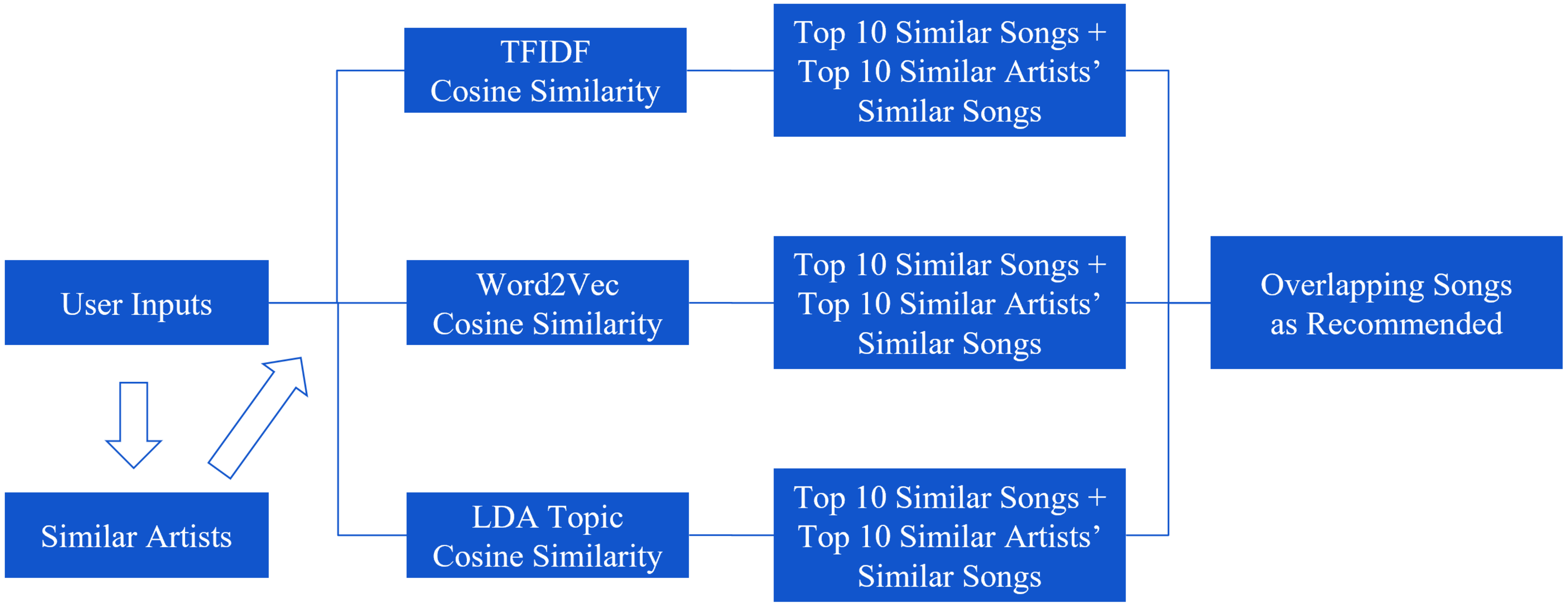


```
# evaluation metric - RMSE

eval_metric = RegressionEvaluator(predictionCol="prediction",
                                  labelCol="frequency",
                                  metricName="rmse")
```



```
# hyper parameter space for cross validation - grid search

ranks = [4, 6, 8, 10, 12, 14, 16]
regParams = [0.1, 0.15, 0.25, 0.27, 0.3, 0.32, 0.35, 0.38]
errors = [[0]*len(ranks)]*len(regParams)
models = [[0]*len(ranks)]*len(regParams)
min_error = float('inf')
i = 0
```

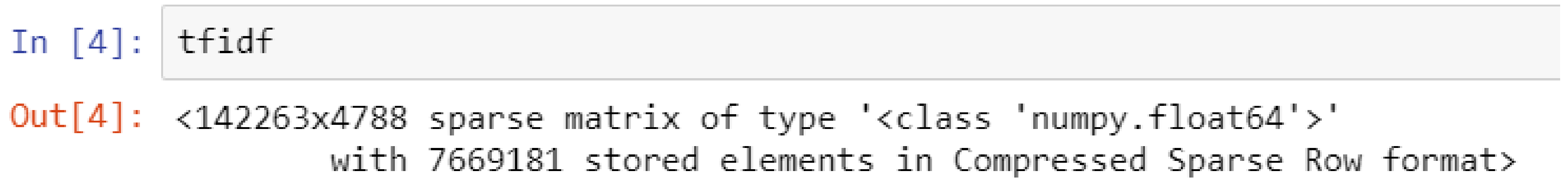
Algorithm Development | Content-Based Filtering


User Inputs
Similar Artists
TFIDF Cosine Similarity
Word2Vec Cosine Similarity
LDA Topic Cosine Similarity
Top 10 Similar Songs + Top 10 Similar Artists' Similar Songs
Top 10 Similar Songs + Top 10 Similar Artists' Similar Songs
Top 10 Similar Songs + Top 10 Similar Artists' Similar Songs
Overlapping Songs as Recommended

Algorithm Development | Content-Based Filtering

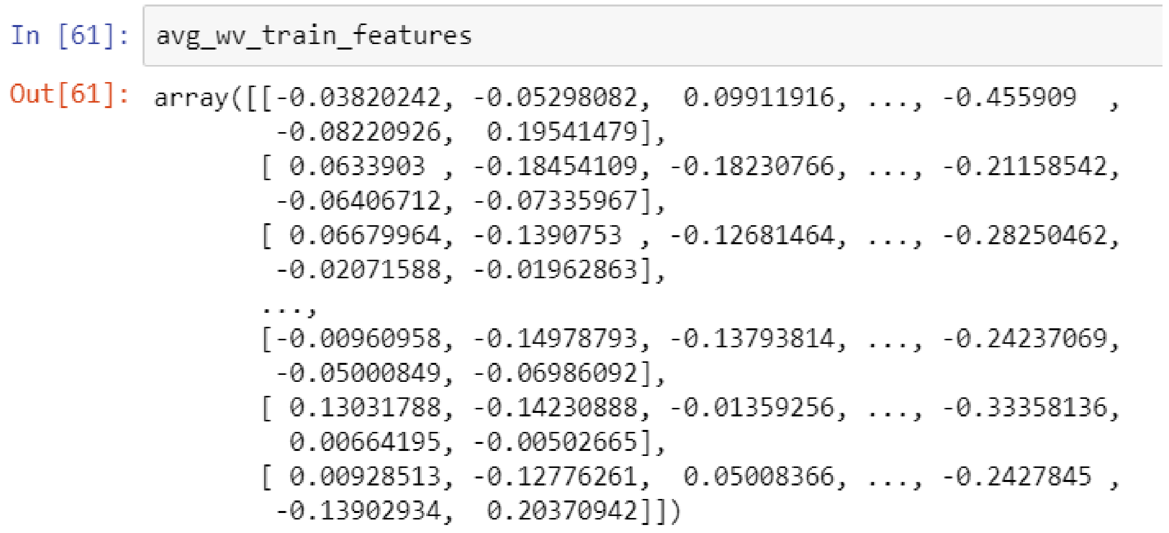
Features Creation

Method 1: TFIDF for Lyrics



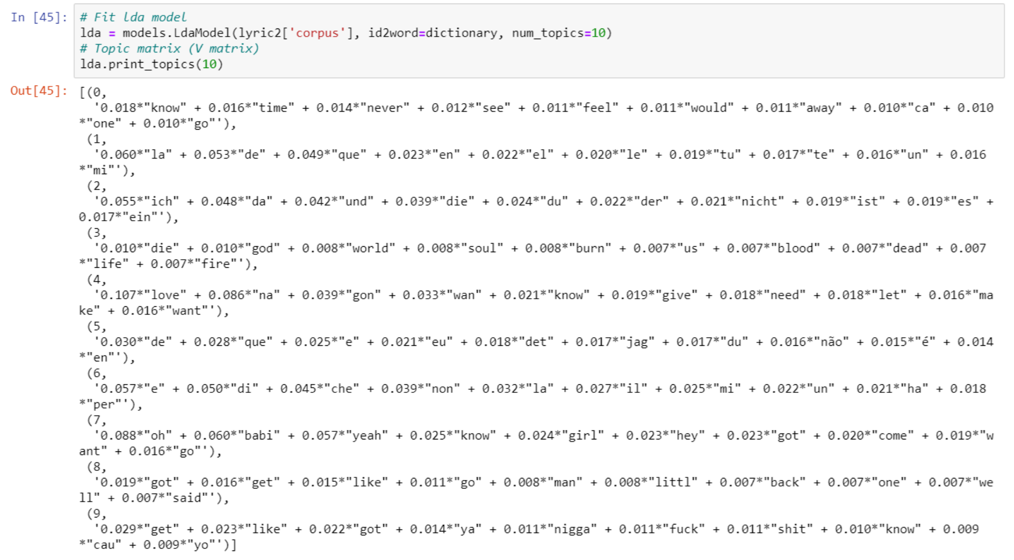
```
In [4]:  tfidf

Out[4]:  <142263x4788 sparse matrix of type '<class 'numpy.float64'>'
                with 7669181 stored elements in Compressed Sparse Row format>
```

Method 2: Word2vec for Lyrics



```
In [61]:  avg_wv_train_features

Out[61]:  array([[-0.03820242, -0.05298082,  0.09911916, ..., -0.455909   ,
                  -0.08220926,  0.19541479],
                 [ 0.0633903 , -0.18454109, -0.18230766, ..., -0.21158542,
                  -0.06406712, -0.07335967],
                 [ 0.06679964, -0.1390753 , -0.12681464, ..., -0.28250462,
                  -0.02071588, -0.01962863],
                 ...,
                 [-0.00960958, -0.14978793, -0.13793814, ..., -0.24237069,
                  -0.05000849, -0.06986092],
                 [ 0.13031788, -0.14230888, -0.01359256, ..., -0.33358136,
                   0.00664195, -0.00502665],
                 [ 0.00928513, -0.12776261,  0.05008366, ..., -0.2427845 ,
                  -0.13902934,  0.20370942]])
```

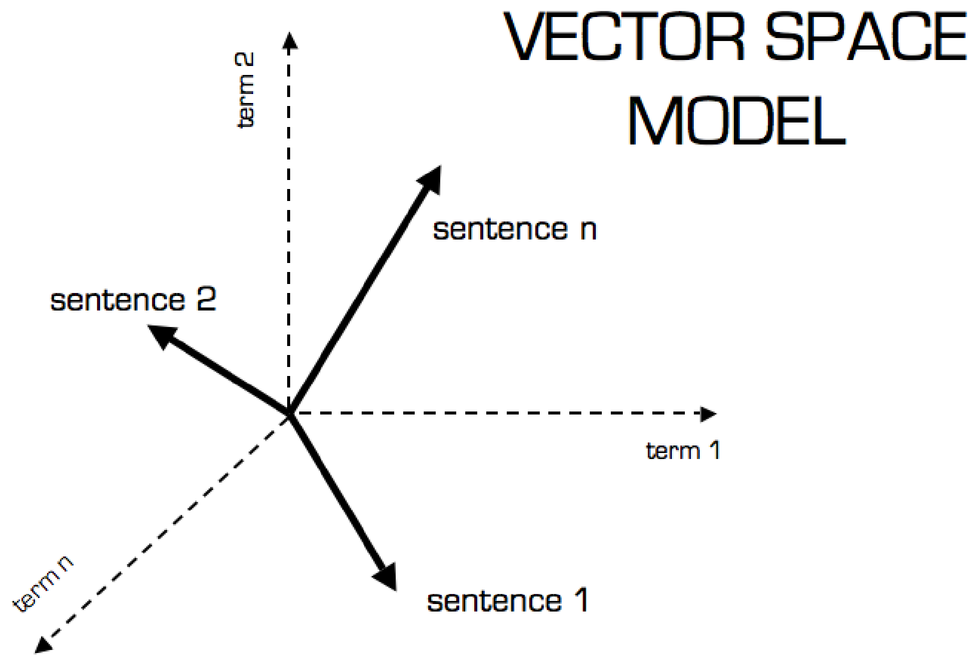
Method 3: LDA for Lyrics



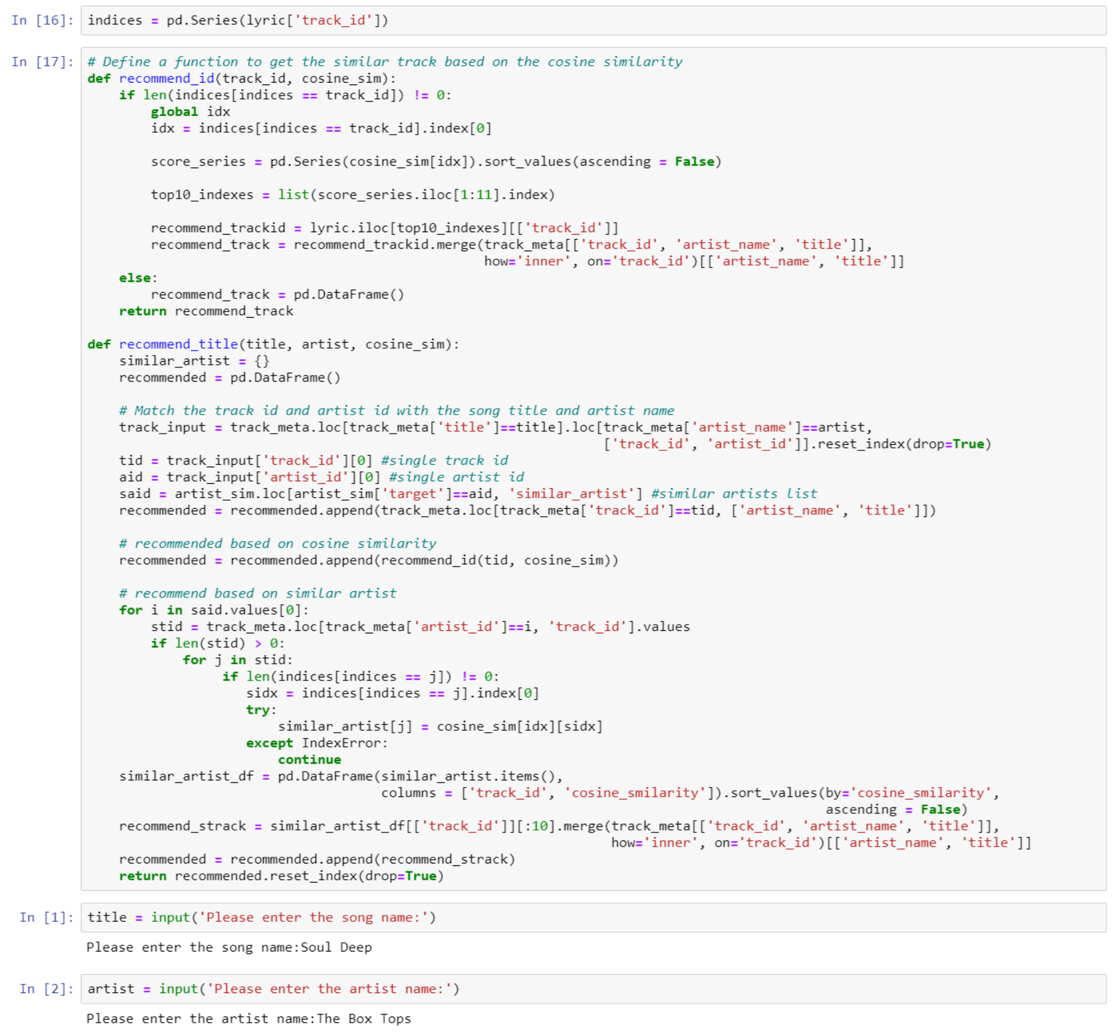
```
In [45]:  # Fit lda model
          lda = models.LdaModel(lyric2['corpus'], id2word=dictionary, num_topics=10)
          # Topic matrix (V matrix)
          lda.print_topics(10)

Out[45]:  [(0,
            '0.018*"know" + 0.016*"time" + 0.014*"never" + 0.012*"see" + 0.011*"feel" + 0.011*"would" + 0.011*"away" + 0.010*"ca" + 0.010
          *"one" + 0.010*"go"'),
           (1,
            '0.060*"la" + 0.053*"de" + 0.049*"que" + 0.023*"en" + 0.022*"el" + 0.020*"le" + 0.019*"tu" + 0.017*"te" + 0.016*"un" + 0.016
          *"mi"'),
           (2,
            '0.055*"ich" + 0.048*"da" + 0.042*"und" + 0.039*"die" + 0.024*"du" + 0.022*"der" + 0.021*"nicht" + 0.019*"ist" + 0.019*"es" +
          0.017*"ein"'),
           (3,
            '0.010*"die" + 0.010*"god" + 0.008*"world" + 0.008*"soul" + 0.008*"burn" + 0.007*"us" + 0.007*"blood" + 0.007*"dead" + 0.007
          *"life" + 0.007*"fire"'),
           (4,
            '0.107*"love" + 0.086*"na" + 0.039*"gon" + 0.033*"wan" + 0.021*"know" + 0.019*"give" + 0.018*"need" + 0.018*"let" + 0.016*"ma
          ke" + 0.016*"want"'),
           (5,
            '0.030*"de" + 0.028*"que" + 0.025*"e" + 0.021*"eu" + 0.018*"det" + 0.017*"jag" + 0.017*"du" + 0.016*"não" + 0.015*"é" + 0.014
          *"en"'),
           (6,
            '0.057*"e" + 0.050*"di" + 0.045*"che" + 0.039*"non" + 0.032*"la" + 0.027*"il" + 0.025*"mi" + 0.022*"un" + 0.021*"ha" + 0.018
          *"per"'),
           (7,
            '0.088*"oh" + 0.060*"babi" + 0.057*"yeah" + 0.025*"know" + 0.024*"girl" + 0.023*"hey" + 0.023*"got" + 0.020*"come" + 0.019*"w
          ant" + 0.016*"go"'),
           (8,
            '0.019*"got" + 0.016*"get" + 0.015*"like" + 0.011*"go" + 0.008*"man" + 0.008*"littl" + 0.007*"back" + 0.007*"one" + 0.007*"we
          ll" + 0.007*"said"'),
           (9,
            '0.029*"get" + 0.023*"like" + 0.022*"got" + 0.014*"ya" + 0.011*"nigga" + 0.011*"fuck" + 0.011*"shit" + 0.010*"know" + 0.009
          *"cau" + 0.009*"yo"')]
```

Algorithm Development | Content-Based Filtering

Cosine Similarity Calculation



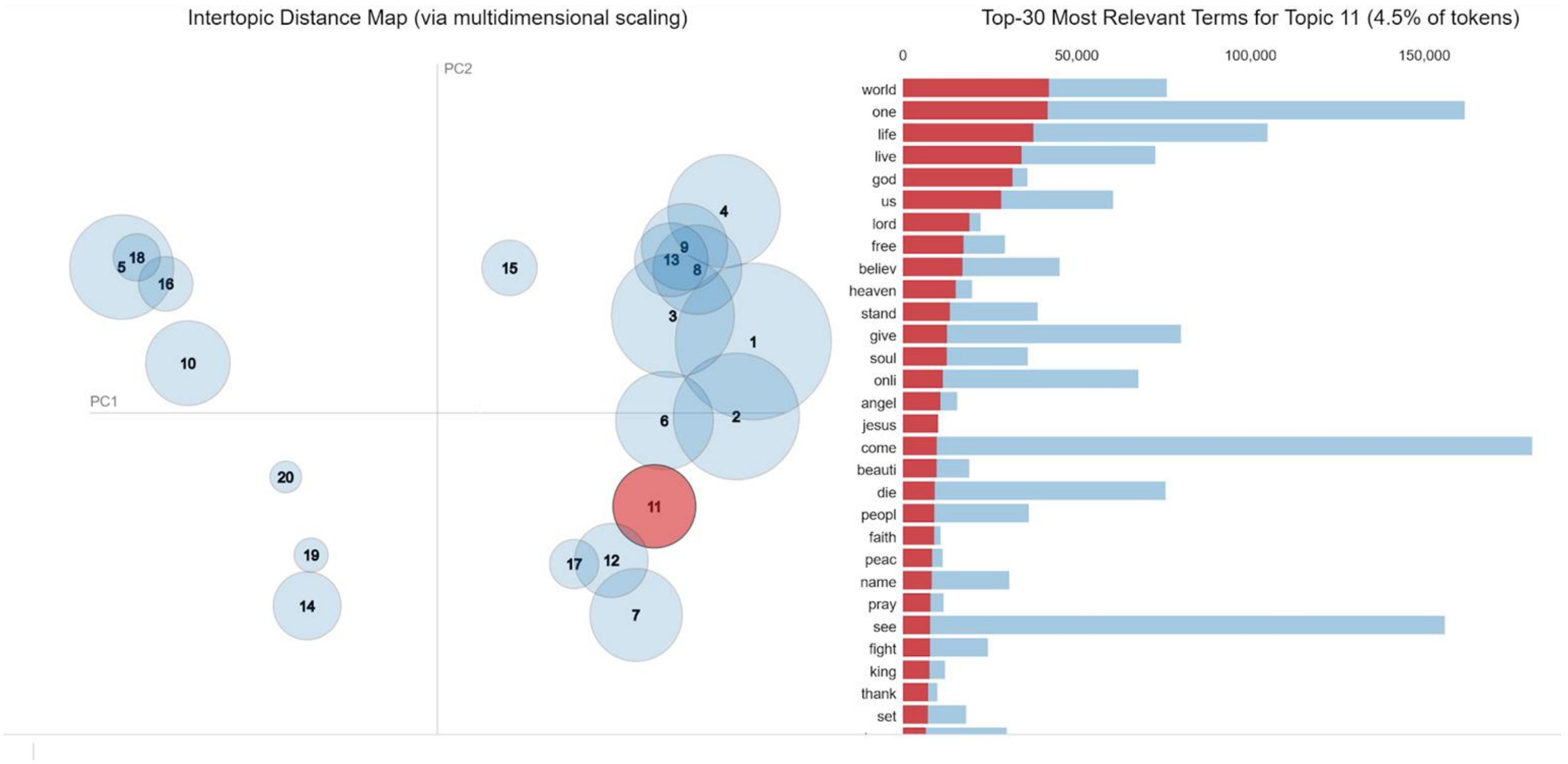
System Building



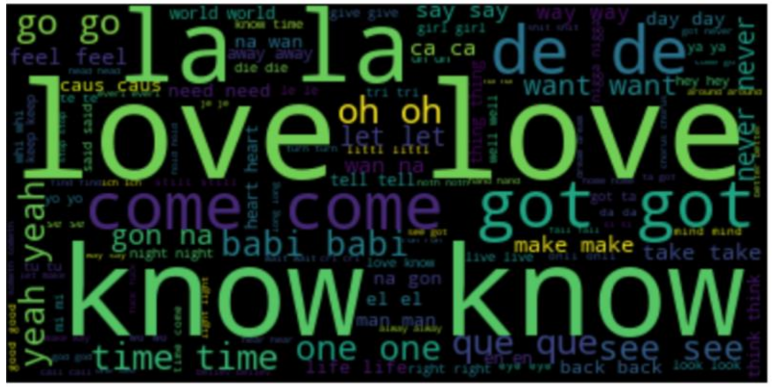
```
In [16]: indices = pd.Series(lyric['track_id'])

In [17]: # Define a function to get the similar track based on the cosine similarity
         def recommend_id(track_id, cosine_sim):
             if len(indices[indices == track_id]) != 0:
                 global idx
                 idx = indices[indices == track_id].index[0]

                 score_series = pd.Series(cosine_sim[idx]).sort_values(ascending = False)

                 top10_indexes = list(score_series.iloc[1:11].index)

                 recommend_trackid = lyric.iloc[top10_indexes][['track_id']]
                 recommend_track = recommend_trackid.merge(track_meta[['track_id', 'artist_name', 'title']],
                                                           how='inner', on='track_id')[['artist_name', 'title']]
             else:
                 recommend_track = pd.DataFrame()
             return recommend_track

         def recommend_title(title, artist, cosine_sim):
             similar_artist = {}
             recommended = pd.DataFrame()

             # Match the track id and artist id with the song title and artist name
             track_input = track_meta.loc[track_meta['title']==title].loc[track_meta['artist_name']==artist,
                                                                          ['track_id', 'artist_id']].reset_index(drop=True)
             tid = track_input['track_id'][0] #single track id
             aid = track_input['artist_id'][0] #single artist id
             said = artist_sim.loc[artist_sim['target']==aid, 'similar_artist'] #similar artists list
             recommended = recommended.append(track_meta.loc[track_meta['track_id']==tid, ['artist_name', 'title']])

             # recommended based on cosine similarity
             recommended = recommended.append(recommend_id(tid, cosine_sim))

             # recommend based on similar artist
             for i in said.values[0]:
                 stid = track_meta.loc[track_meta['artist_id']==i, 'track_id'].values
                 if len(stid) > 0:
                     for j in stid:
                          if len(indices[indices == j]) != 0:
                             sidx = indices[indices == j].index[0]
                             try:
                                 similar_artist[j] = cosine_sim[idx][sidx]
                             except IndexError:
                                 continue
             similar_artist_df = pd.DataFrame(similar_artist.items(),
                                              columns = ['track_id', 'cosine_smilarity']).sort_values(by='cosine_smilarity',
                                                                                                        ascending = False)
             recommend_strack = similar_artist_df[['track_id']][:10].merge(track_meta[['track_id', 'artist_name', 'title']],
                                                                           how='inner', on='track_id')[['artist_name', 'title']]
             recommended = recommended.append(recommend_strack)
             return recommended.reset_index(drop=True)

In [1]: title = input('Please enter the song name:')

        Please enter the song name:Soul Deep

In [2]: artist = input('Please enter the artist name:')

        Please enter the artist name:The Box Tops
```

Lyrics Analysis

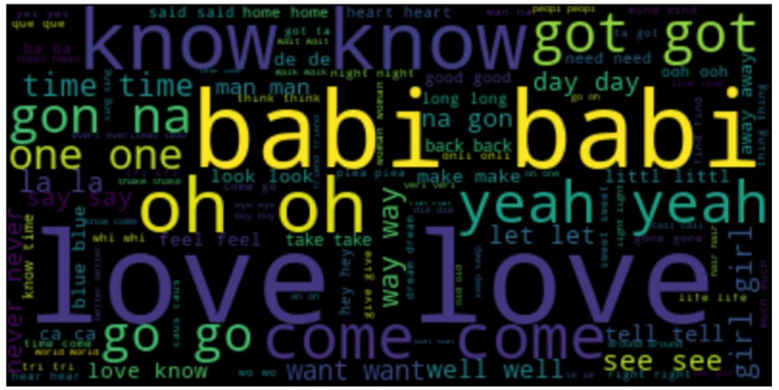
Topic Modeling (Cluster n=20)



Word Cloud Visualization



2000s

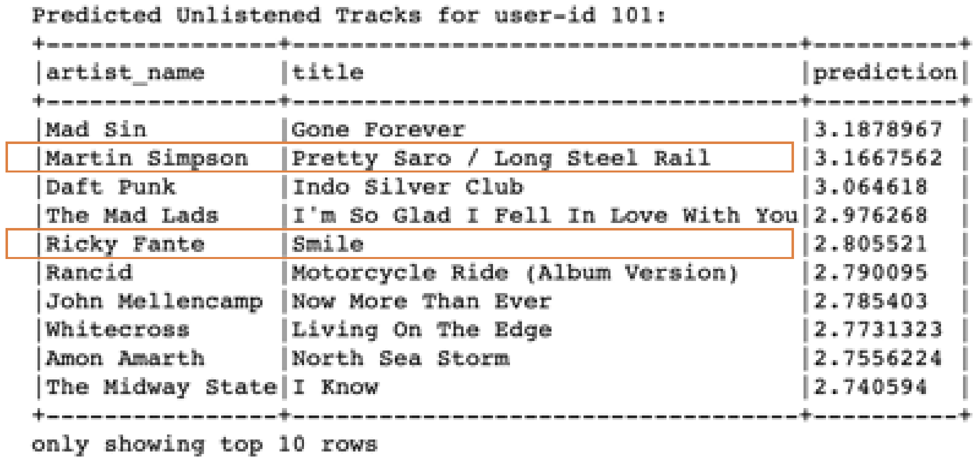


60s

Results | ALS

Basic Recommendation

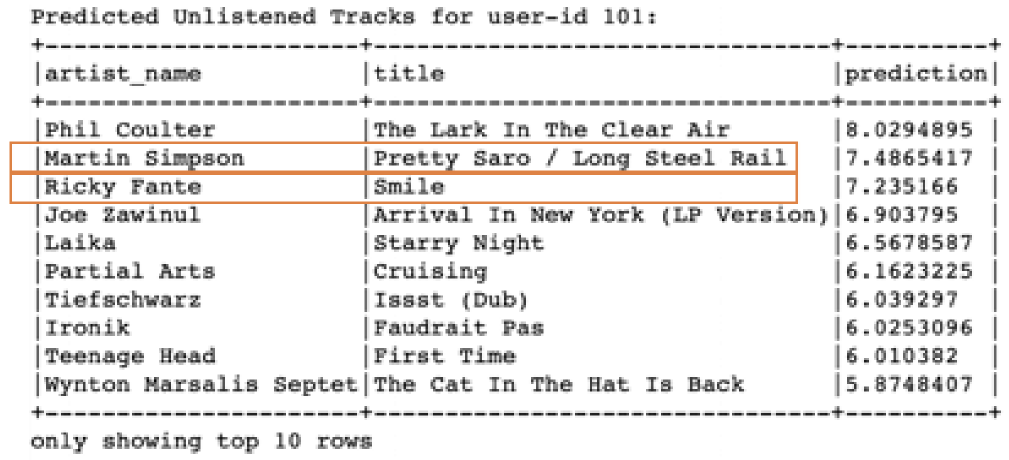
- **RMSE on test: 6.24**
- **Average frequency: 3**
- **RMSE with average frequency: 6.23**
- **Result for user-id 101:**



```
Predicted Unlistened Tracks for user-id 101:
+----------------+----------------------------------+----------+
|artist_name     |title                             |prediction|
+----------------+----------------------------------+----------+
|Mad Sin         |Gone Forever                      |3.1878967 |
|Martin Simpson  |Pretty Saro / Long Steel Rail     |3.1667562 |
|Daft Punk       |Indo Silver Club                  |3.064618  |
|The Mad Lads    |I'm So Glad I Fell In Love With You|2.976268 |
|Ricky Fante     |Smile                             |2.805521  |
|Rancid          |Motorcycle Ride (Album Version)   |2.790095  |
|John Mellencamp |Now More Than Ever                |2.785403  |
|Whitecross      |Living On The Edge                |2.7731323 |
|Amon Amarth     |North Sea Storm                   |2.7556224 |
|The Midway State|I Know                            |2.740594  |
+----------------+----------------------------------+----------+
only showing top 10 rows
```

Recommendation for tracks listened >=2

- **RMSE on test: 9.21**
- **Average frequency: 6**
- **RMSE with average frequency: 9.23**
- **Result for user-id 101**:

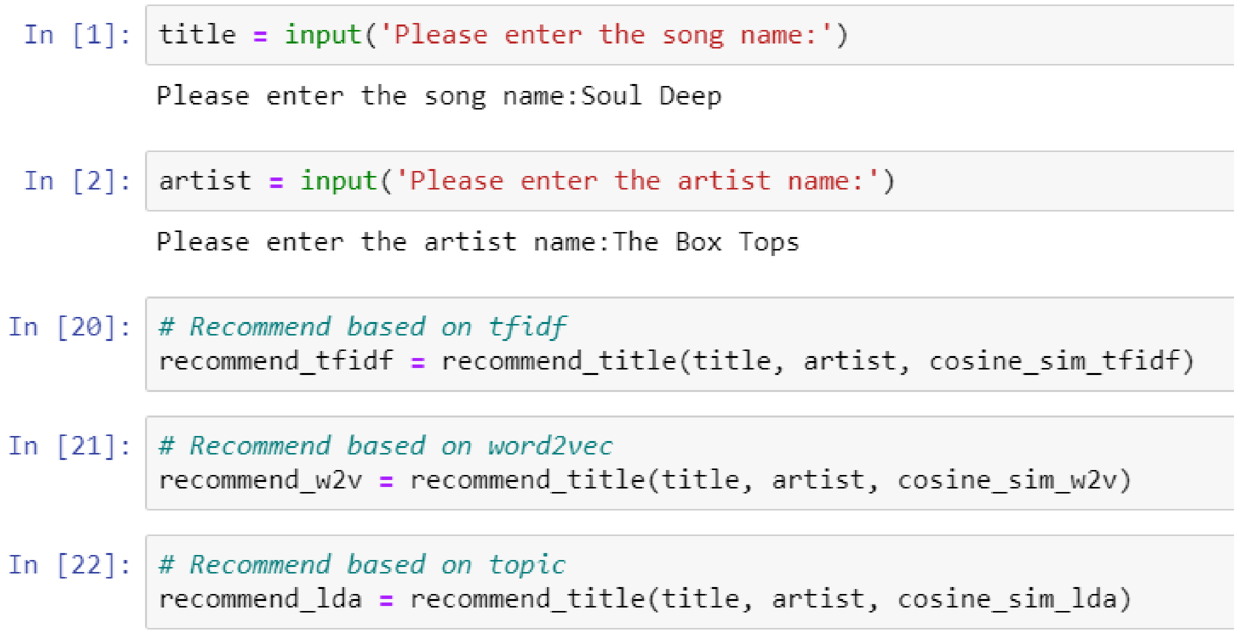


```
Predicted Unlistened Tracks for user-id 101:
+----------------------+--------------------------------+----------+
|artist_name           |title                           |prediction|
+----------------------+--------------------------------+----------+
|Phil Coulter          |The Lark In The Clear Air       |8.0294895 |
|Martin Simpson        |Pretty Saro / Long Steel Rail   |7.4865417 |
|Ricky Fante           |Smile                           |7.235166  |
|Joe Zawinul           |Arrival In New York (LP Version)|6.903795  |
|Laika                 |Starry Night                    |6.5678587 |
|Partial Arts          |Cruising                        |6.1623225 |
|Tiefschwarz           |Issst (Dub)                     |6.039297  |
|Ironik                |Faudrait Pas                    |6.0253096 |
|Teenage Head          |First Time                      |6.010382  |
|Wynton Marsalis Septet|The Cat In The Hat Is Back      |5.8748407 |
+----------------------+--------------------------------+----------+
only showing top 10 rows
```

Conclusion: Many songs have only been listened once. Although the second model has a higher RMSE on test, it behaves relatively better when compared to average frequency. Since those tracks are listened more, we infer those songs can better represent users' tastes. The circled items might be of highest recommendation quality.

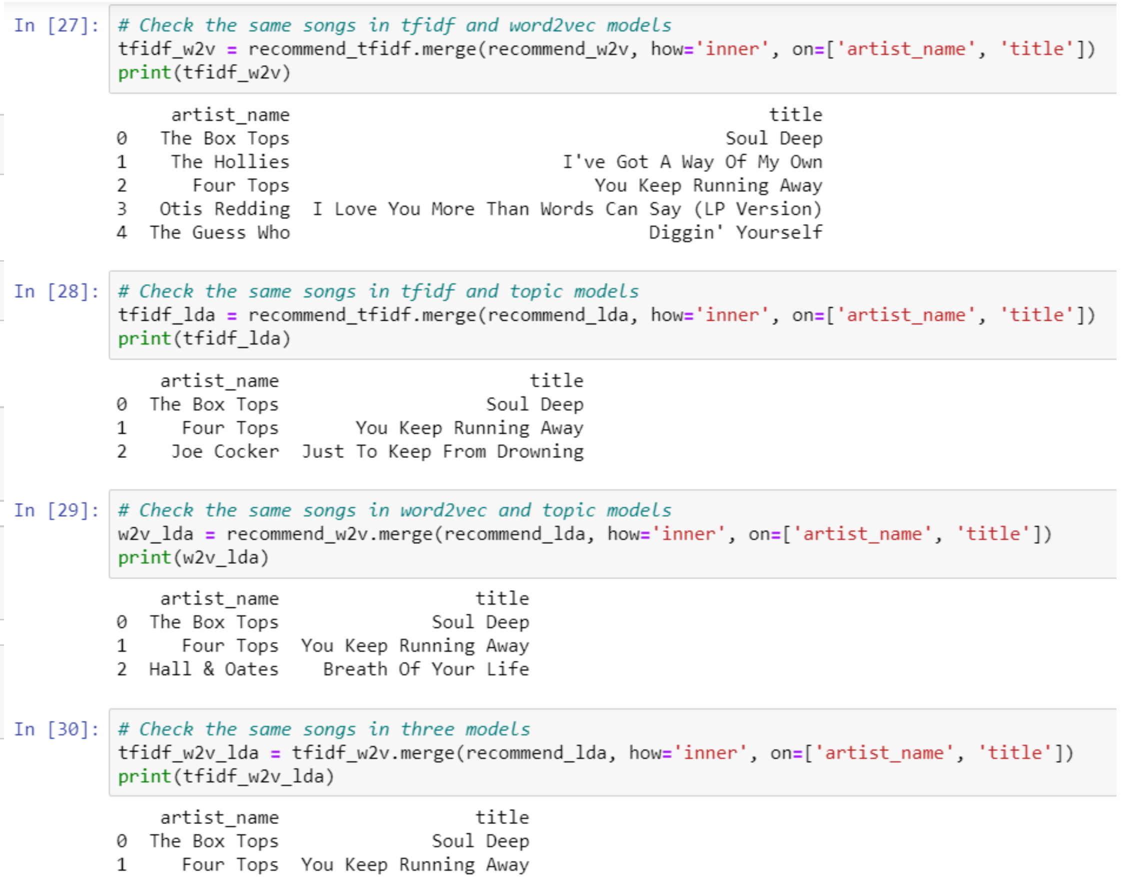
Results | Content-Based Filtering

Simulation of A New User



```
In [1]:  title = input('Please enter the song name:')

         Please enter the song name:Soul Deep

In [2]:  artist = input('Please enter the artist name:')

         Please enter the artist name:The Box Tops

In [20]: # Recommend based on tfidf
         recommend_tfidf = recommend_title(title, artist, cosine_sim_tfidf)

In [21]: # Recommend based on word2vec
         recommend_w2v = recommend_title(title, artist, cosine_sim_w2v)

In [22]: # Recommend based on topic
         recommend_lda = recommend_title(title, artist, cosine_sim_lda)
```

Recommendation



```
In [27]: # Check the same songs in tfidf and word2vec models
         tfidf_w2v = recommend_tfidf.merge(recommend_w2v, how='inner', on=['artist_name', 'title'])
         print(tfidf_w2v)

              artist_name                                        title
         0   The Box Tops                                    Soul Deep
         1    The Hollies                        I've Got A Way Of My Own
         2      Four Tops                          You Keep Running Away
         3   Otis Redding  I Love You More Than Words Can Say (LP Version)
         4  The Guess Who                               Diggin' Yourself

In [28]: # Check the same songs in tfidf and topic models
         tfidf_lda = recommend_tfidf.merge(recommend_lda, how='inner', on=['artist_name', 'title'])
         print(tfidf_lda)

             artist_name                         title
         0  The Box Tops                     Soul Deep
         1     Four Tops         You Keep Running Away
         2    Joe Cocker  Just To Keep From Drowning

In [29]: # Check the same songs in word2vec and topic models
         w2v_lda = recommend_w2v.merge(recommend_lda, how='inner', on=['artist_name', 'title'])
         print(w2v_lda)

             artist_name                  title
         0  The Box Tops              Soul Deep
         1     Four Tops  You Keep Running Away
         2  Hall & Oates    Breath Of Your Life

In [30]: # Check the same songs in three models
         tfidf_w2v_lda = tfidf_w2v.merge(recommend_lda, how='inner', on=['artist_name', 'title'])
         print(tfidf_w2v_lda)

             artist_name                  title
         0  The Box Tops              Soul Deep
         1     Four Tops  You Keep Running Away
```

Results | Content-Based Filtering

Lyrics

Darlin' I don't know much
I know I love you so much
A lot depends on your touch
My love is a river running soul deep
A way down inside me it's a soul deep
Too big to hide, can't be denied
Love is a river running soul deep

I worked myself to euphoria
Just to show I adore ya
There's nothing I wouldn't do for ya
Cause my love is a river running soul deep
A way down inside me it's a soul deep
Too big to hide, can't be denied
Love is a river running soul deep

All I ever, ever hoped to be
Depends on your love for me
If you believe me, if you should leave me
I'd be nothing but a jilted male
I know darned well, I could tell, but

I don't know much
I know I love you so much
A lot depends on your touch
My love is a river running soul deep
A way down inside me it's a soul deep
Too big to hide, can't be denied
Love is a river running soul deep
My love is a river running soul deep
A way down inside me it's a soul deep
My love is a river running soul deep
A way down inside me it's a soul deep
My love is a river running soul deep
A way down inside me it's a soul deep

Source: LyricFind

Songwriters: Per Gessle

Soul Deep lyrics © Kobalt Music Publishing Ltd.

Lyrics

You keep running away
Though I beg you not to leave
But still you won't stay
Darlin' you keep running away
Tear my heart apart every step of the way

You're here today and gone tomorrow
Leavin' this heart of mine in sorrow
Now you come around every now and then
Long enough to hurt me, and then you're gone again

Darlin' you, you keep running away
Oh, I begged you not to leave, you never stay
Now you, you keep running away
Leavin' me here to face another lonely day

To you all of this is just a game
But each time you came here, I feel the pain
But I've got so much love for you
I keep wanting you, no matter what you do

All I want to do is take care of you
Everything I have in my life, I'll share with you
This soul of mine has been possessed by you
Darling my heart has been obsessed with you
Just look at me, I'm not the man, I used to be

I used to be proud, I used to be strong
But all of that's changed girl, since you come along
Your lovin' sweetness is my weakness
Though I need you, dear, I just can't keep you near

Running away, running away, running away
Running away, running away, running away

Each time you go, the hurt comes callin'
My days become nights, darlin'
My nights become so much longer
You're in my life, you're in my heart

But I can't get you, get you into my arms
Darlin' you, you keep runnin' away
Darlin' you, you just keep runnin' away

Source: LyricFind

Songwriters: Jr. / Brian Holland / Edward Holland / Edward / Jr. Holland / Lamont Dozier / Lamont Herbert Dozier

You Keep Running Away lyrics © Sony/ATV Music Publishing LLC

Conclusion

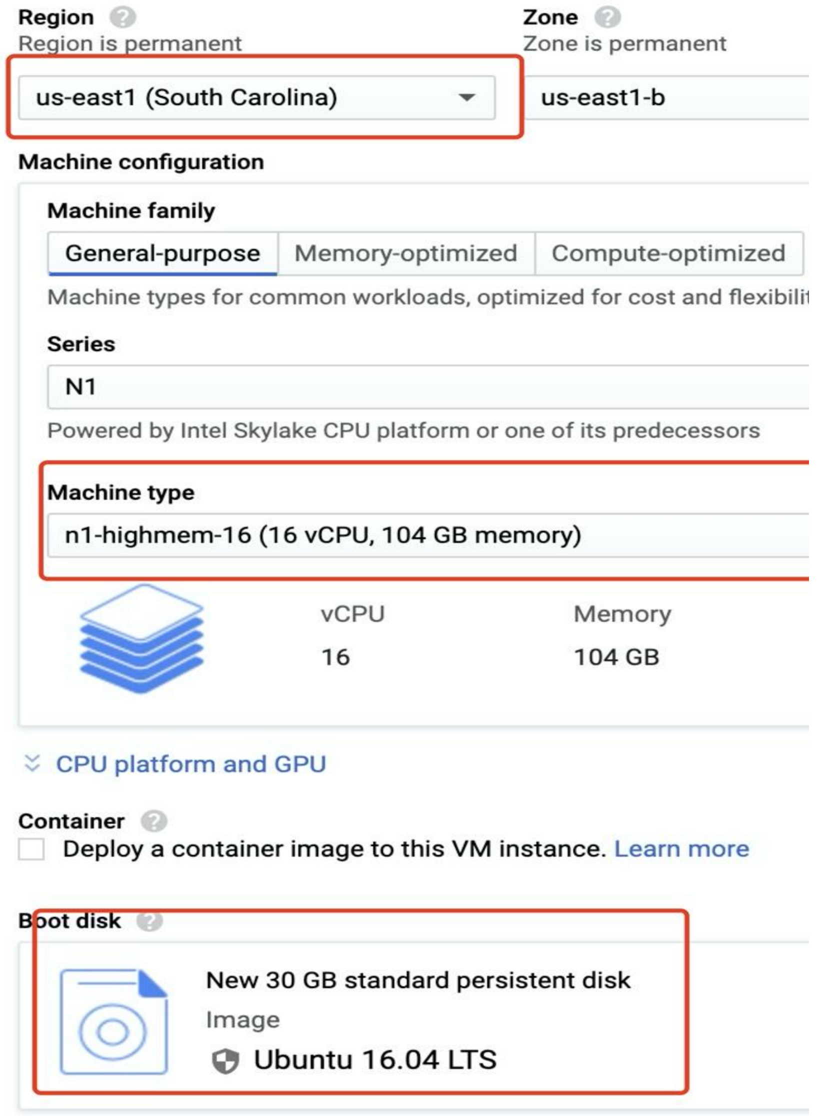
- ✓ Built a recommender systems using a dataset with 1,019,318 unique users and 384,546 unique songs.
- ✓ ALS algorithm for our collaborative filtering
  - For old users with enough listening history to generate personalized recommendations.
- ✓ Content-based recommender: combined artist similarity and lyric similarity (LDA, Word2vec and TF-IDF modeling)
  - For new users with only one or a few search and listening history.
  - Similar songs for the current song will be recommended.
- ✓ Considering that Spotify has about 2 million monthly active users, our project is close to the monthly magnitude of the industry-level.

Lessons Learned

- Cloud memory
- Spark configuration and data types

```
conf = (conf.set('spark.executor.memory', '30G')
             .set('spark.driver.memory', '30G')
             .set('spark.driver.maxResultSize', '30G'))
```

```
# Cache those to save memory

train_df = train_df.cache()
valid_df = valid_df.cache()
test_df = test_df.cache()
```



Region
Region is permanent
us-east1 (South Carolina)

Zone
Zone is permanent
us-east1-b

Machine configuration

Machine family
General-purpose | Memory-optimized | Compute-optimized
Machine types for common workloads, optimized for cost and flexibili

Series
N1
Powered by Intel Skylake CPU platform or one of its predecessors

Machine type
n1-highmem-16 (16 vCPU, 104 GB memory)

vCPU	Memory
16	104 GB

CPU platform and GPU

Container
Deploy a container image to this VM instance. Learn more

Boot disk
New 30 GB standard persistent disk
Image
Ubuntu 16.04 LTS

Further Steps

Hybrid system

More dimensions of recommendation is always better

Google naturally combined plenty of recommendation strategies in its wide and deep recommendation system with neural networks and ensemble methods.

New Ideas

User2vec

Graph algorithms

Content-based filtering using music audio

References

1. A Beginner's Guide to Word2Vec and Neural Word Embeddings. (n.d.). Retrieved from https://pathmind.com/wiki/word2vec
2. Content-based Filtering. (2012, January 24). Retrieved from http://recommender-systems.org/content-based-filtering/
3. Karantyagi. (n.d.). karantyagi/Restaurant-Recommendations-with-Yelp. Retrieved from https://github.com/karantyagi/Restaurant-Recommendations-with-Yelp
4. Li, S. (2018, June 1). Topic Modeling and Latent Dirichlet Allocation (LDA) in Python. Retrieved from https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24
5. MODELING. (n.d.). Retrieved from https://xindizhao19931.wixsite.com/spotify2/modeling
6. Welcome! (n.d.). Retrieved from http://millionsongdataset.com/

Thank You

for making us become great data scientists

From Jie Lu, Li Yihao, Chaoying Bao, Tram Le