Music Recommendation System

By: Sandeep Yerra and Pranav Sukumaran

Mars Is a Cold Place
The 15th Planet
2:54
3:49

Project Summary

01 Goal

Building a comprehensive recommendation system for music recommendation

02 Dataset

Million Song Dataset + Musixmatch includes listening history, song metadata, lyrics, artist information.

03 Methods

Matrix Factorization Algorithm - ALS

Content-based- TFIDF, LDA, Word2Vec

04 Tools

Built using python, pyspark, Amazon EMR, S3 and Streamlit

Mars Is a Cold Place
The 15th Planet

2:54 3:49

Background

- History of music discovery (from radio to digital streaming)
- The role of recommendation systems in the digital era
- The challenge of choice overload in digital music platforms

Business Problem

- Need for effective music recommendation for user retention
- Enhancing user experience through personalized content
- Importance of tackling the 'cold start' problem in music recommendations

Mars Is a Cold Place
The 15th Planet

2:54

3:49

Million Song Dataset

musixmatch

Dataset from Echo Nest

Train_triplets.txt (3.0GB)

- 1,019,318 unique users
- 384,546 unique MSD songs
- 48,373,586 (user, song, play count) triplets

Track_metadata.csv (300MB)

- 1,000,000 songs metadata (artist_familiarity, title, songId etc.)

musiXmatch Dataset

Lyrics.csv (2.0GB)

- 19,045,332 words
- 237,662 unique songs

artist_similarity.csv

taste_profile_song_to_tracks

Mars Is a Cold Place
The 15th Planet
2:54
3:49

Dataset Information

Data Consolidation → Extract Metadata → Table Joins → Save Datasets → Model Training

(Lyrics, MSD, User Listening History)
Mapping MSD id to Musixmatch id

(Artist, IDs, Artist Similarity, Familiarity, Lyrics)
Extract from SQLite DB

Joining dataframes to create datasets for model training

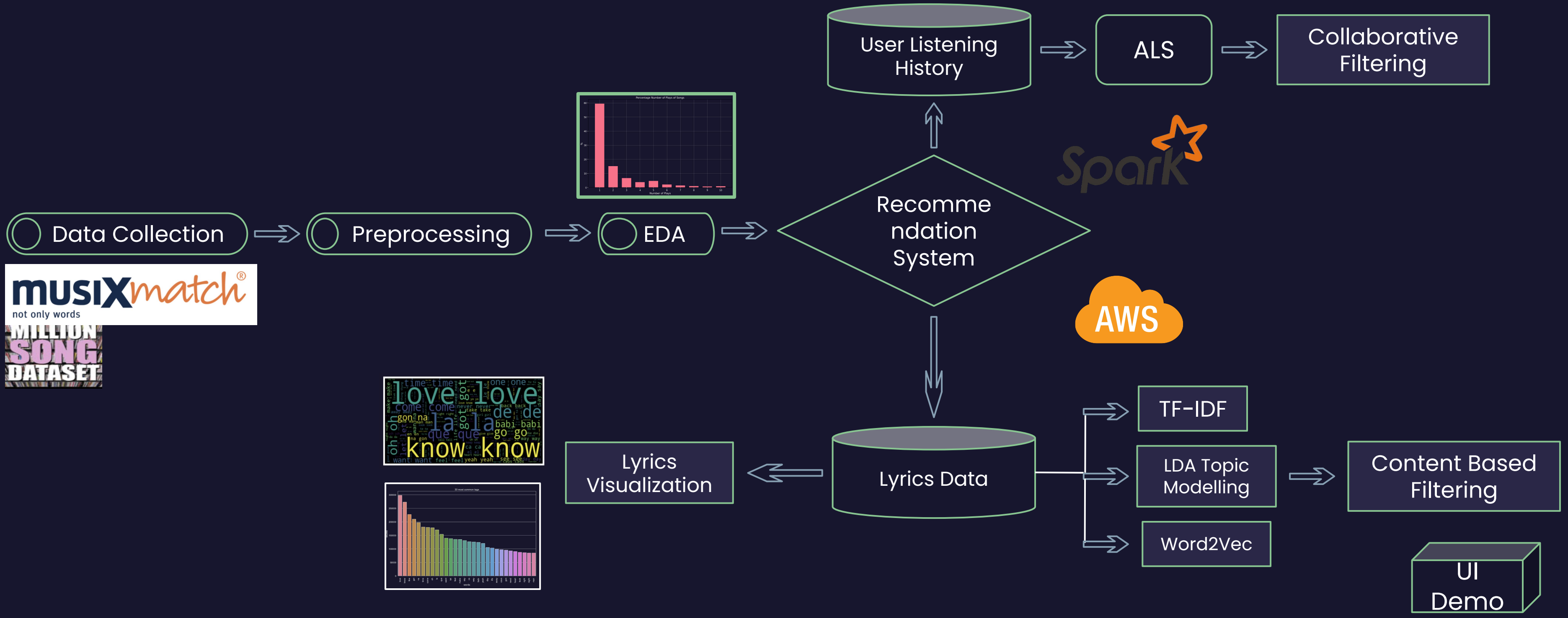
Saving datasets as parquet for efficient loading

Mars Is a Cold Place
The 15th Planet

2:54

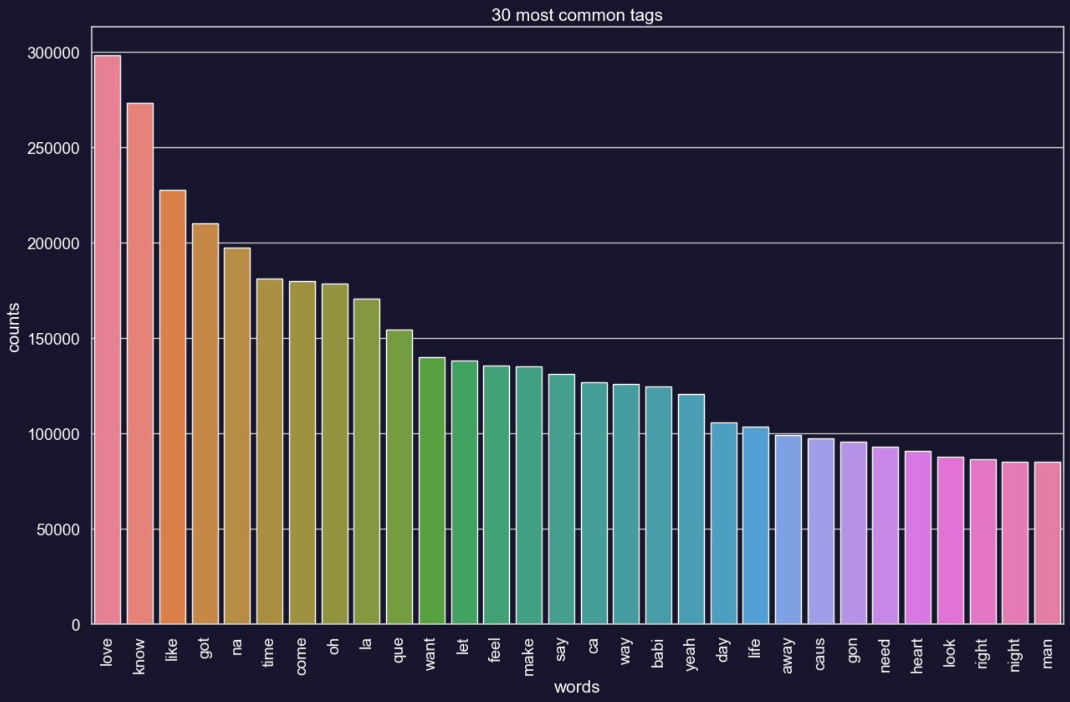
3:49

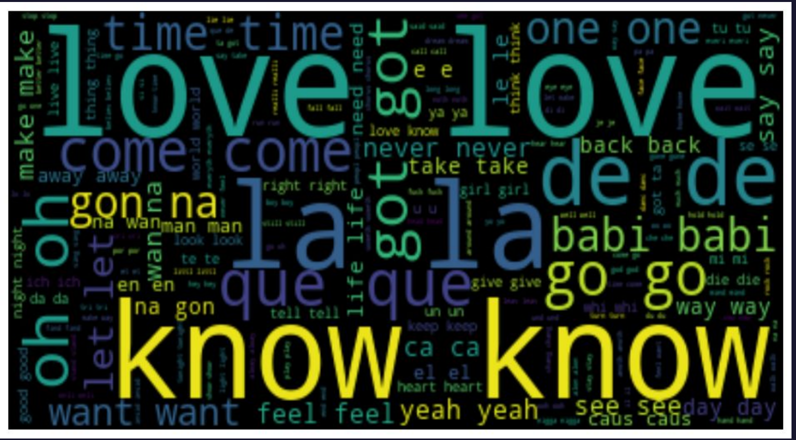
Project Design



Mars Is a Cold Place
The 15th Planet
2:54
3:49

EDA - All Time Lyrics



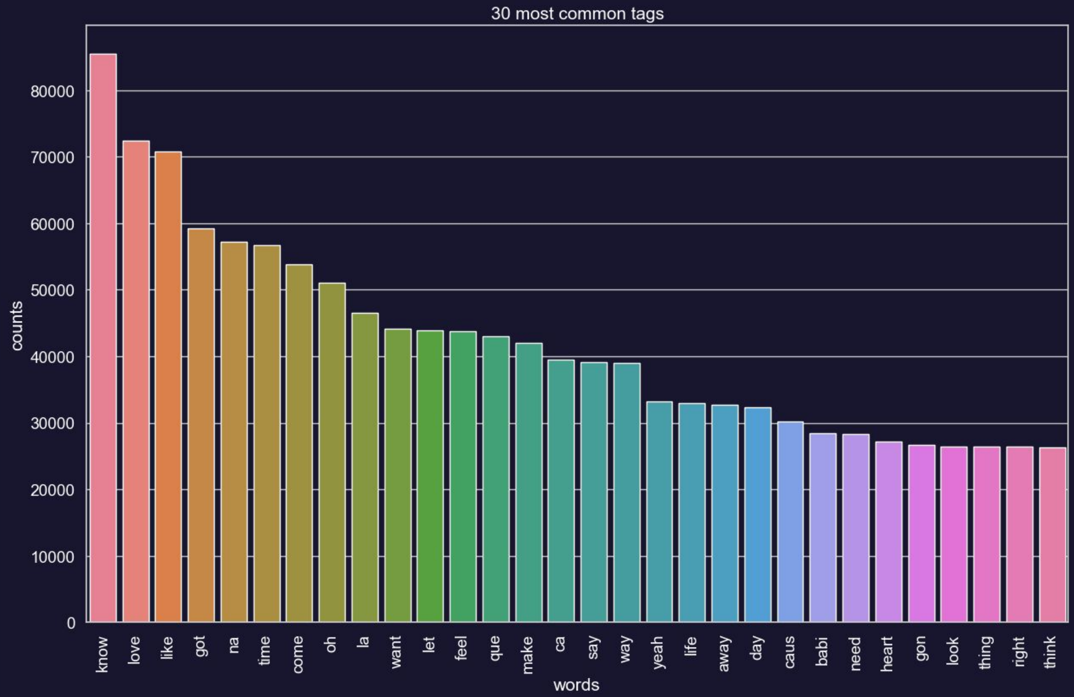


Mars Is a Cold Place
The 15th Planet

2:54

3:49

EDA – 2000's Lyrics



Mars Is a Cold Place
The 15th Planet

2:54

3:49

Content Based Filtering

TFIDF

TF-IDF:

$$TF\text{-}IDF(\text{word}) = TF(\text{word}) \times IDF(\text{word})$$

Importance-Weighted Features: TF-IDF highlights unique characteristics of songs, emphasizing distinctive lyrics or metadata in recommendations.

Filtering Noise: Reduces the impact of frequently occurring, less significant words or features, focusing on unique aspects that differentiate songs.

Word2Vec

Family of model architectures and optimizations that can be used to learn word embeddings from large datasets

Word2Vec: Creates vector representations of songs where vectors capture semantic relationships, enabling recommendations based on song similarity.

Context-Based Learning: Learns song relationships from user playlists, leveraging context to recommend songs with similar thematic content.

LDA Topic

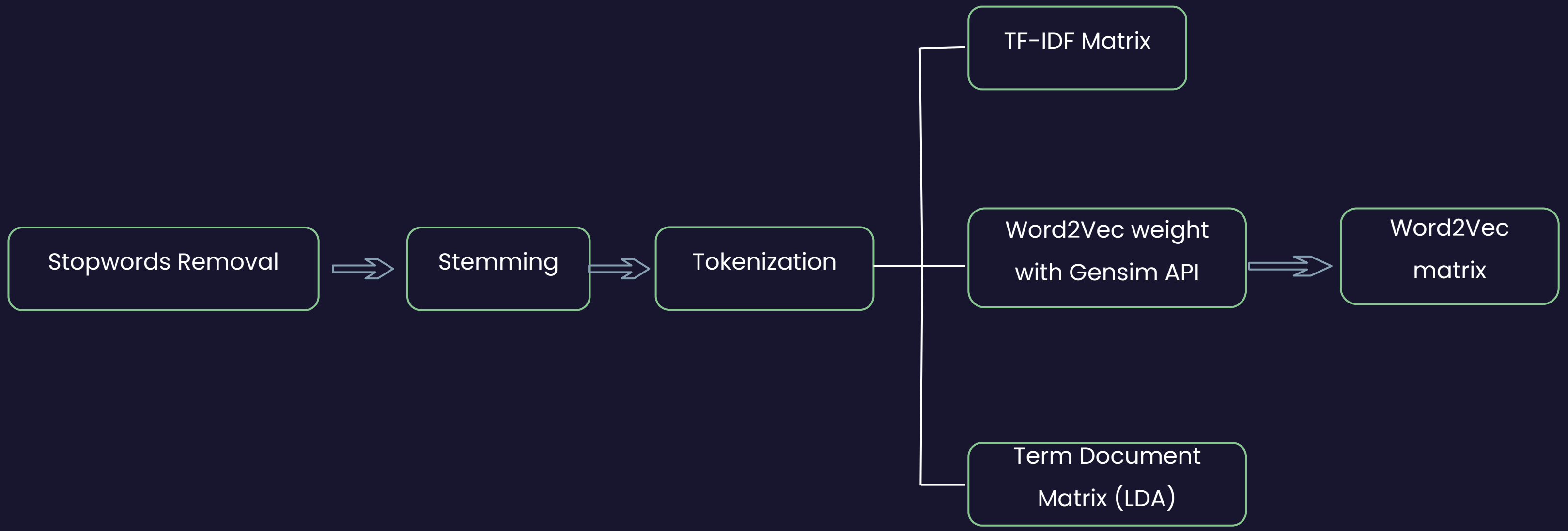
Latent Dirichlet Allocation
Topic modeling technique to extract topics from a given corpus.

Thematic Grouping: LDA discovers latent topics within song lyrics or metadata, grouping songs into thematic clusters for recommendation.

Diverse Recommendations: Offers a variety of songs by recommending from different thematic topics, catering to varied user interests.

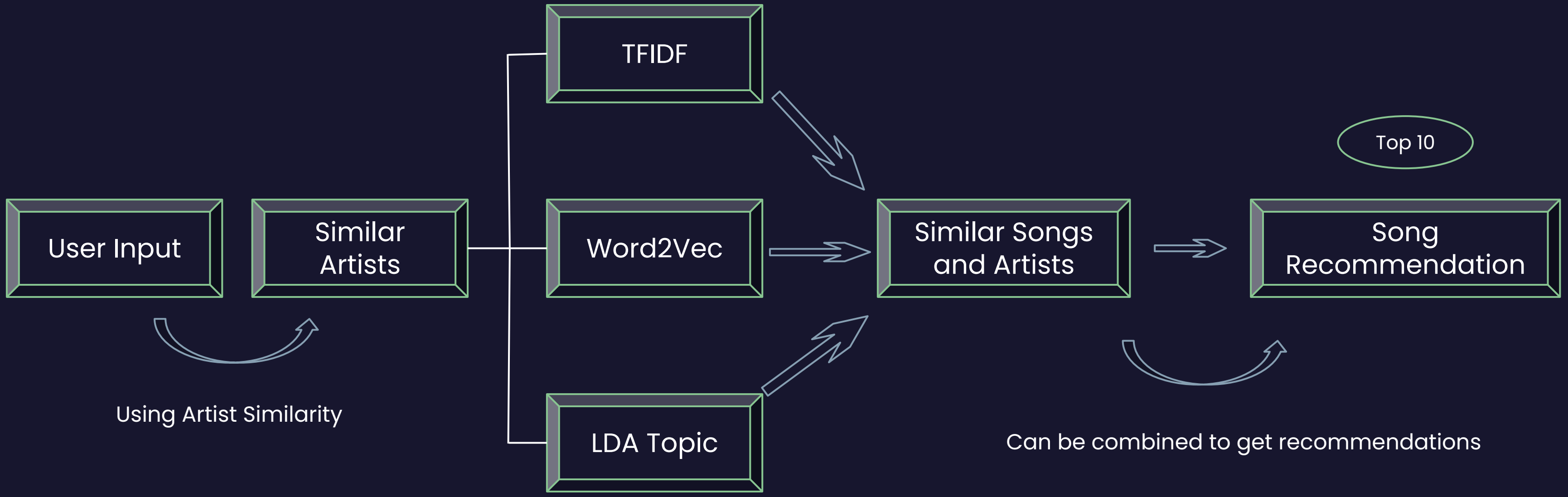
Mars Is a Cold Place
The 15th Planet
2:54
3:49

Data Preprocessing and Preparation



Mars Is a Cold Place
The 15th Planet
2:54
3:49

Algorithm -Content Based Filtering



Mars Is a Cold Place
The 15th Planet

2:54

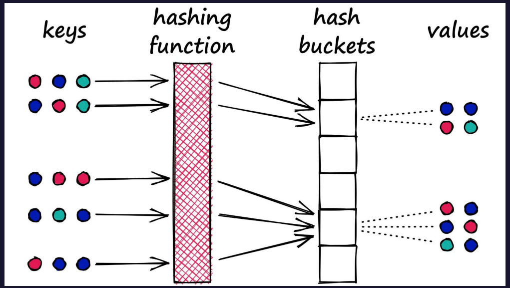
3:49

Locality-sensitive hashing

- Why LSH over Cosine Similarity?

Cosine Similarity ⇒ Computationally Expensive

Especially for large datasets



⇩

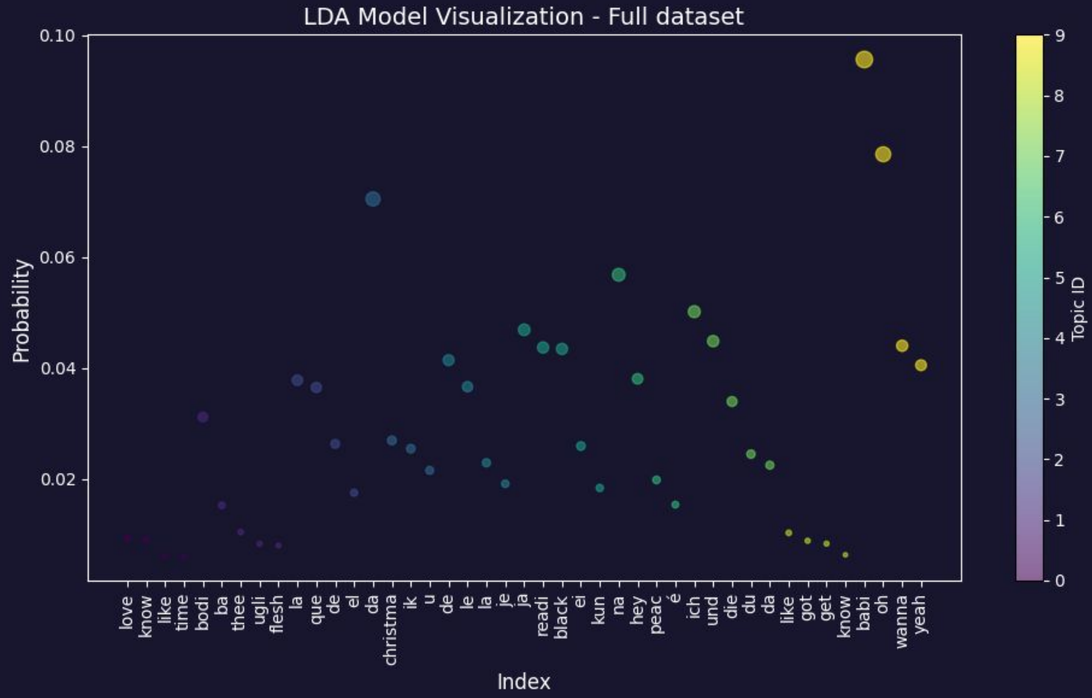
Efficient approximation of cosine similarity: Maximize Collisions

- Reducing complexity through hashing
- Scales effectively with size of data

LDA Topics

Terms and Term weights

```
TOPIC 0
        0.3565810976616964
love    0.009388950954965048
know    0.008998458253338474
like    0.006294540987858848
time    0.006195170707496157
==========
TOPIC 1
bodi    0.03118721445172204
ba      0.015277980639994724
thee    0.010457111314668276
ugli    0.008354062493226232
flesh   0.008030665819900917
==========
TOPIC 2
        0.2930046576761982
la      0.037817126179108485
que     0.03650581621240135
de      0.02634319802219863
el      0.01753992912318418
==========
TOPIC 3
        0.09703995646107175
da      0.07046059899790494
christma        0.026966616142479517
ik      0.025477535676983257
u       0.02157393882680142
```



Mars Is a Cold Place
The 15th Planet

2:54

3:49

Content Based Filtering Results

Input Song

Super Cat – "Trash and Ready"

Reggae

TFIDF

Word2Vec

LDA Topic

Recommended Songs

1. Ward 21 – "Never Sell Out"
2. Sizzla – "Sound The Trumpet"
3. T.O.K. – "Guardian Angel"
4. Fantan Mojah – "Feel Di Pain"
5. Cocoa Tea – "A Business"
6. Pinchers – "Hold Me"
7. Mavado – "House Cleaning"
8. T.O.K. – "Gal You Lead"
9. Mr. Vegas – "Deh Pon The Scene (Album Version)"
10. Shabba Ranks – "Hood Top"

All models recommends the same 10 songs for “Trash and Ready” - Super Cat

Reggae

Mars Is a Cold Place

The 15th Planet

2:54

3:49

Collaborative Filtering - ALS

Matrix Factorization:

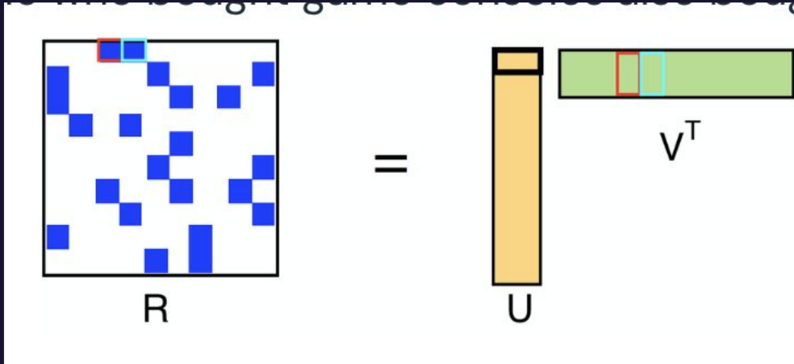
- ALS decomposes the user-item interaction matrix into two lower-dimensional matrices: one for users and one for items.
- These matrices capture latent factors that represent user preferences and item characteristics.

Alternating Optimization:

- ALS optimizes these matrices alternately:
    - Fix one matrix (e.g., user matrix) and optimize the other (e.g., item matrix).
    - Then, fix the optimized matrix and update the other.
    - This process alternates until convergence.

Objective Function:

- The optimization minimizes the least squares error between the observed user-item interactions and the predicted interactions based on matrix multiplication.
- ALS aims to find the best-fitting user and item matrices that minimize this error.



Algorithm -Collaborative Filtering

Matrix Factorization using ALS

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

# Initializing ALS learner
als = ALS()

# Setting the parameters for the method
als.setMaxIter(5)\
    .setSeed(seed)\
    .setItemCol("new_songId")\
    .setRatingCol("Plays")\
    .setUserCol("new_userId")
```

Approximate user-item interaction matrix (user-song plays) with the product of two lower-dimensional matrices, representing latent factors for users and items (songs).

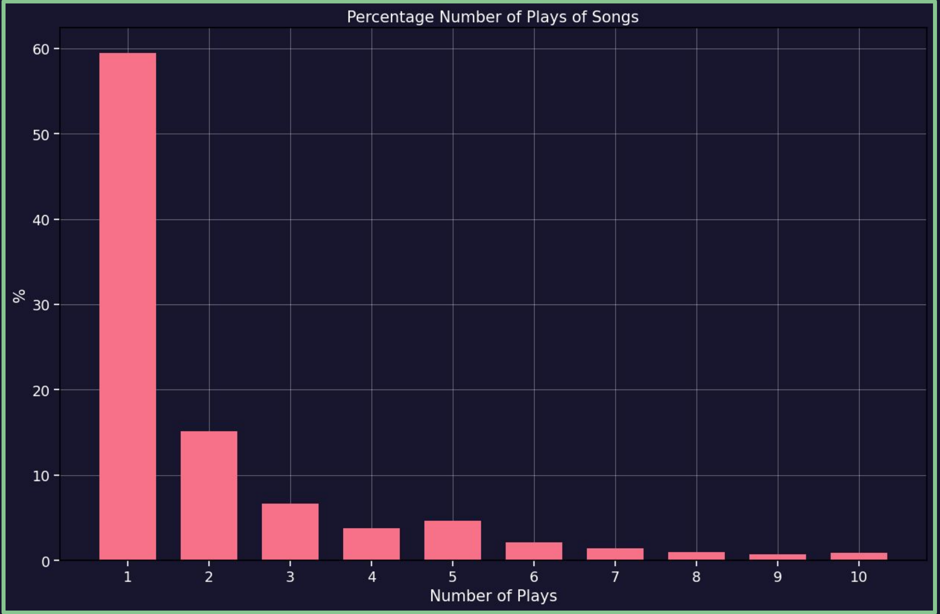
Matrix Factorization: The goal is to find two matrices U (user matrix of size $m \times k$) and V (item matrix of size $n \times k$) such that their product approximates R. Here, k is the rank, representing the number of latent factors.

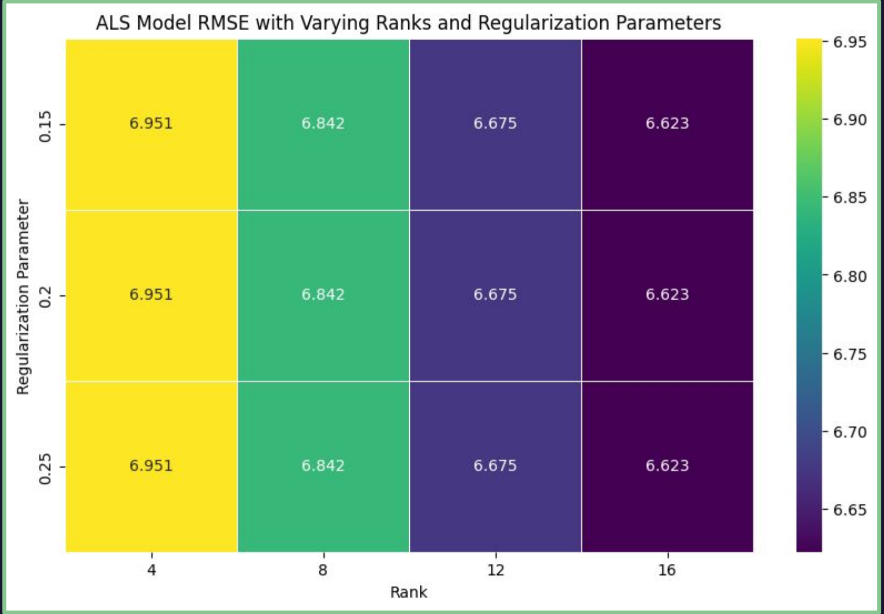
$$R \approx U \times V^T$$

The optimization aims to minimize the difference between R and the product $U \times V^T$, measured using the RMSE.

Mars Is a Cold Place
The 15th Planet
2:54
3:49

Algorithm -Collaborative Filtering





Mars Is a Cold Place
The 15th Planet
2:54
3:49

Algorithm -Collaborative Filtering

Results

```
Training: 1742149, Validation: 580227, Test: 579449
```

```
The best model was trained with regularization parameter 0.25
The best model was trained with rank 16
```

```
The average number of plays in the dataset is 3.0
The RMSE on the average set is 6.666914929507369
```

```
Songs user has listened to:

+---------------+--------+
|    artist_name|   title|
+---------------+--------+
|    OneRepublic|Secrets|
|Vampire Weekend|     Run|
|Vampire Weekend|Holiday|
+---------------+--------+
```

Song Recommendation

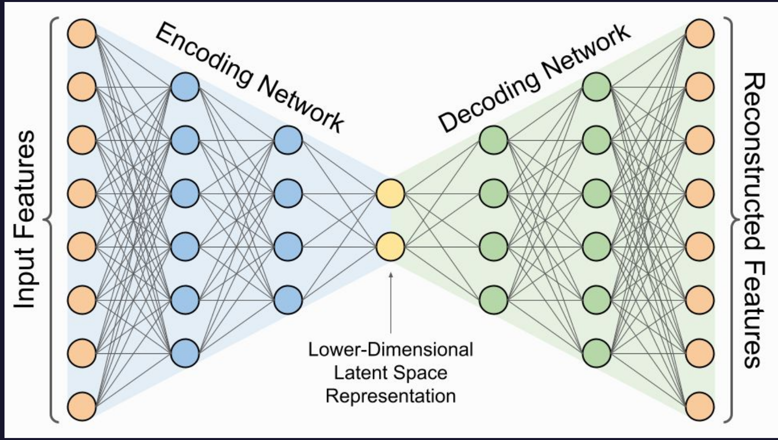
```
Predicted Songs:
+----------------+----------------------------------+-----------+
|artist_name     |title                             |prediction|
+----------------+----------------------------------+-----------+
|Van Halen       |Humans Being (Album Version)      |40.39946  |
|Scumbucket      |Call Me Anyone                    |29.412138 |
|The Ark         |Clamour For Glamour (Radio Edit)  |27.924892 |
|Les Nubians     |Unfaithful / Si Infidèle          |21.477009 |
|Willie Gonzalez |No Podrás Escapar De Mi (En Vivo)|20.079897 |
|Young Jeezy     |Keep It Movin                     |20.077602 |
|Desmond Dekker  |No Place Like Home                |19.63868  |
|Jay Reatard     |No Time                           |19.552528 |
|Super Cat       |Trash And Ready                   |19.505892 |
|Anthony Rother  |Back Home                         |19.38985  |
+----------------+----------------------------------+-----------+
```

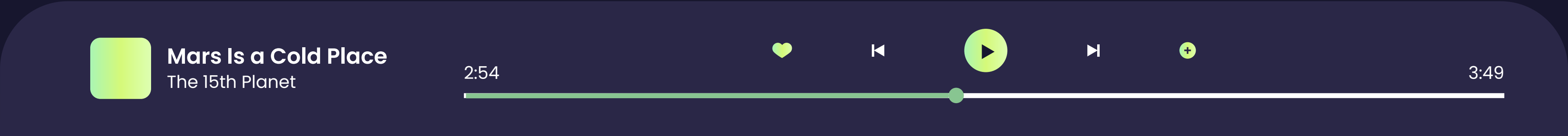
Mars Is a Cold Place
The 15th Planet
2:54
3:49

AutoEncoders for Collaborative Filtering

- Designed a feedforward autoencoder architecture. It typically consists of an encoder and a decoder.
- The encoder takes the user-item interaction matrix as input and maps it to a lower-dimensional latent space.
- The decoder reconstructs the original user-item interaction matrix from the latent space.
- Train the autoencoder using the user-item interaction matrix as both input and target.
- The objective is to minimize the reconstruction loss, between the input and output matrices.

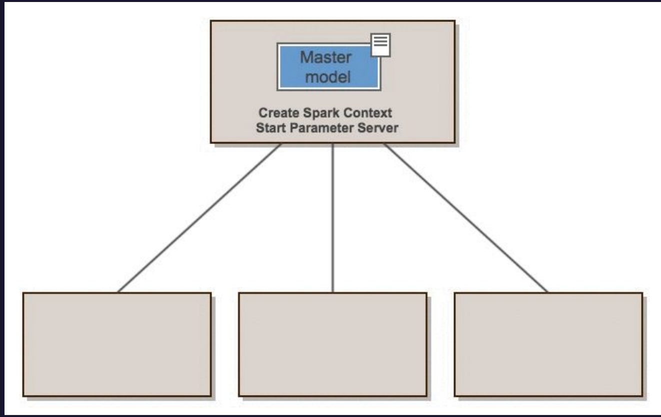
- Recommendation: To make recommendations, you can calculate user-item interaction scores in the latent space (e.g., dot product between user and item embeddings).
- Rank items based on their interaction scores and recommend the top-N items to users.





AutoEncoder with Elephas (10000 interactions)

```python
# Autoencoder Model
input_dim = df_final.select("features").first()[0].size  # Total size of user_id_encoded + track_id_encoded
input_layer = Input(shape=(input_dim,), name='input_layer')

# Encoder
encoded = Dense(128, activation='relu')(input_layer)
encoded = Dense(64, activation='relu')(encoded)

# Decoder
decoded = Dense(128, activation='relu')(encoded)
output_layer = Dense(input_dim, activation='sigmoid')(decoded)

# Compile Model
model = Model(inputs=input_layer, outputs=output_layer)
model.compile(optimizer=Adam(), loss='binary_crossentropy')

# Preparing RDD for Elephas
rdd = df_final.select("features").rdd.map(lambda row: (row.features.toArray(), row.features.toArray()))

# Elephas Model
spark_model = SparkModel(model, frequency='epoch', mode='synchronous')
spark_model.fit(rdd, epochs=5, batch_size=32, verbose=0, validation_split=0.1)

# Create the track_id to track_id_index mapping
track_id_mapping = create_track_id_mapping(df_transformed)
```



artist_name	title
Delorean	Soon
The Cinematic Orc...	That Home
Pixies	Break My Body
The Stranglers	Always The Sun
Poison	Ride The Wind

Mars Is a Cold Place
The 15th Planet
2:54
3:49

Limitations and Improvements

Evaluation of Content Based Filtering challenges

- **Lack of Ground Truth label**

Next Steps

- **Build a deep recommendation model using Neural Collaborative filtering and expand using GPU**
- **Build Hybrid Recommendation system using ColdStart**

Mars Is a Cold Place
The 15th Planet
2:54
3:49

References

ALS – Pyspark:
https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.recommendation.ALS.html

Kieran Tan Kah Wang. "Collaborative Filtering in PySpark: An Introduction to Collaborative Filtering and Implementation in PySpark Using Alternating Least Squares (ALS) Algorithm." Towards Data Science, October 9, 2020. https://towardsdatascience.com/collaborative-filtering-in-pyspark-52617dd91194.

"Million Song Dataset." Accessed [10/2023]. http://millionsongdataset.com/pages/getting-dataset/.

Najafabadi, Maryam Khanian et al. "Improving the Accuracy of Collaborative Filtering Recommendations Using Clustering and Association Rules Mining on Implicit Data." Advanced Informatics School (AIS), Universiti Teknologi Malaysia (UTM), Kuala Lumpur, Malaysia.

Fabio Aiolli: Preliminary Study on a recommender system for the Million Songs Dataset challenge. (2011)

Mars Is a Cold Place
The 15th Planet
2:54
3:49