Recommendation System

A deep learning based trust- and tag-aware recommender system

Paper Link

Hansin Patwa - I040
Maitri Shah - I057

About

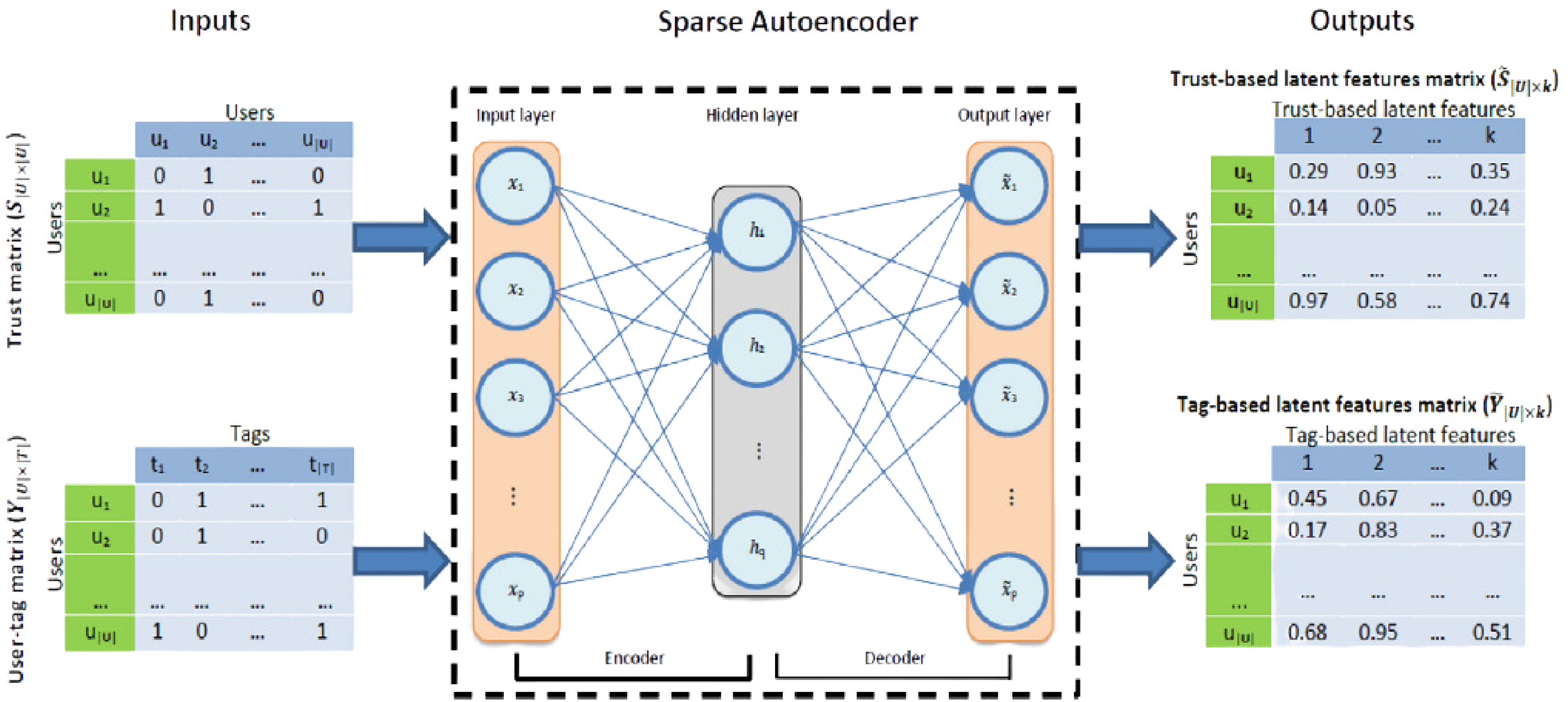
Recommender Systems and Collaborative Filtering: Recommender systems are used in various applications to help users find relevant items. Collaborative filtering, a popular technique, uses past user ratings to predict future interests. However, the lack of sufficient ratings can lead to reduced performance.

Utilizing Additional Resources: To improve recommendation accuracy, the paper suggests incorporating additional resources such as trust relationships and tag information. These resources can provide valuable insights into user preferences.

About

Sparse Data and Computational Complexity: Trust and tag data often suffer from sparsity since users might not provide enough information. Additionally, these resources tend to have large dimensions, leading to increased computational complexity in similarity calculations.

Proposed Solution - Deep Neural Networks: The paper proposes a new recommendation model that leverages deep neural networks to model trust relationships and tag information. Specifically, it employs a sparse autoencoder to extract latent features from user-user trust relationships and user-tag matrices.


Inputs
Sparse Autoencoder
Outputs
Trust matrix ($S_{|U|\times|U|}$)
Users
u_1 u_2 ... $u_{|U|}$
u_1 0 1 ... 0
u_2 1 0 ... 1
... ...
$u_{|U|}$ 0 1 ... 0
User-tag matrix ($Y_{|U|\times|T|}$)
Tags
t_1 t_2 ... $t_{|T|}$
u_1 0 1 ... 1
u_2 0 1 ... 0
... ...
$u_{|U|}$ 1 0 ... 1
Input layer
Hidden layer
Output layer
x_1 x_2 x_3 x_p
h_1 h_2 h_q
$\tilde{x}_1$ $\tilde{x}_2$ $\tilde{x}_3$ $\tilde{x}_p$
Encoder
Decoder
Trust-based latent features matrix ($\tilde{S}_{|U|\times k}$)
Trust-based latent features
1 2 ... k
u_1 0.29 0.93 ... 0.35
u_2 0.14 0.05 ... 0.24
... ...
$u_{|U|}$ 0.97 0.58 ... 0.74
Tag-based latent features matrix ($\tilde{Y}_{|U|\times k}$)
Tag-based latent features
1 2 ... k
u_1 0.45 0.67 ... 0.09
u_2 0.17 0.83 ... 0.37
... ...
$u_{|U|}$ 0.68 0.95 ... 0.51

About

Calculating Similarity and Prediction: The extracted latent features are then used to calculate similarity values between users. These similarity values help form the nearest neighbors of a target user, allowing the system to predict unseen items for that user.

Benefits of the Proposed Method: The proposed method addresses data sparsity issues and reduces the computational complexity of recommender systems. This is achieved by using latent features with smaller dimensions compared to the original data.

Recommendation

In the recommendation process of the proposed method, latent features of users are utilized to measure similarity values between users. The paper discusses three different scenarios for providing recommendations based on these latent features:

1. **Trust-Based Recommendations - Using Cosine Similarity**
2. **Tag-Based Recommendations - Using Cosine Similarity**
3. **Combined Recommendations - Using the F-beta Score**

After obtaining the similarity values based on these scenarios, the nearest neighbors of the target user are identified. The ratings of unseen items are then predicted using these similarity values.

Dataset

Last.fm dataset contains 1892 users who are interconnected in a social network based on friendship relationships1. The users can tag music tracks and artists, and a list of relevant music tracks is provided for each user. In this dataset, the artists are considered as the items to be recommended to the target user.

Delicious dataset contains 1867 users who can tag and share their personal bookmarks (URLs)2. Moreover, the users are interconnected in a social network based on mutual fan relationships and each user has bookmarks and tag assignments. In this dataset, the URLs are considered as the items to be recommended to the target user.

We use the social information of the Last.fm and Delicious datasets as the trust relationships between users.

References

Research Paper	Research Gap	Solution Proposed	Results
SSL-SVD: Semisupervised learning-based sparse trust recommendation, (2020).	Cold Start Challenge and Data Sparsity Challenge	-Mining sparse trust relationships between users. -Utilizing Transductive Support Vector Machine (SVM) to combine trust factors. -Incorporating sparse trust information into the SVD++ recommendation model.	**For all Users** MAE: 0.756 RMSE: 0.989 **For Cold Start Users** MAE: 0.802 RMSE: 1.052
A multistep priority-based ranking for top-N recommendation using social and tag information, (2021)	Existing top-N recommendation methods primarily rely on user-item rating data and struggles to fully capture user preferences, interactions, and handle the cold-start problem.	-Leverages user-item ratings, social network connections, and item tags, while implementing a multistep priority-based ranking system.	**LAST.FM Dataset - HIT Rate @ Top-N recommendation** HR@5 - 0.1286 HR@10 - 0.1565 HR@15 - 0.1758
Deep auto encoder model with convolutional text networks for video recommendation, (2019)	Collaborative filtering algorithms face sparsity and cold start limitations in recommender systems.	-Utilizes user behavior data, attributes for latent feature mining. -Employs preprocessing, embedding, convolutional layers, and autoencoders. Final prediction uses multilayer perception.	**MovieLens Dataset** MAE: 0.715+- 0.003 RMSE: 0.914+-0.002

Final Evaluation

Dataset

Last.fm dataset contains 1892 users who are interconnected in a social network based on friendship relationships1. The users can tag music tracks and artists, and a list of relevant music tracks is provided for each user. In this dataset, the artists are considered as the items to be recommended to the target user.

We use the social information of the Last.fm as the trust relationships between users.

Exploratory Data Analysis - 1

Various statistics on our data files:

Friends

Shape: (25434, 2)

Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25434 entries, 0 to 25433
Data columns (total 2 columns):

#	Column	Non-Null Count	Dtype
0	userID	25434 non-null	int64
1	friendID	25434 non-null	int64

dtypes: int64(2)
memory usage: 397.5 KB
None

Unique:
userID 1892
friendID 1892
dtype: int64

No duplicates: True

Tags

Shape: (11946, 2)

Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11946 entries, 0 to 11945
Data columns (total 2 columns):

#	Column	Non-Null Count	Dtype
0	tagID	11946 non-null	int64
1	tagValue	11946 non-null	object

dtypes: int64(1), object(1)
memory usage: 186.8+ KB
None

Unique:
tagID 11946
tagValue 11946
dtype: int64

No duplicates: True

User Tagged Artists (Date)

Shape: (186479, 6)

Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 186479 entries, 0 to 186478
Data columns (total 6 columns):

#	Column	Non-Null Count	Dtype
0	userID	186479 non-null	int64
1	artistID	186479 non-null	int64
2	tagID	186479 non-null	int64
3	day	186479 non-null	int64
4	month	186479 non-null	int64
5	year	186479 non-null	int64

dtypes: int64(6)
memory usage: 8.5 MB
None

Unique:
userID 1892
artistID 12523
tagID 9749
day 4
month 12
year 10
dtype: int64

No duplicates: True

Exploratory Data Analysis - 2

- 17,632 unique artist ID and name
- 1,892 unique user ID
- 92,834 total data points (artist/user pairs)

```
***Artist Plays***

Shape: (92834, 2)

Info:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 92834 entries, 2800 to 88660
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   userID    92834 non-null  int64
 1   artistID  92834 non-null  int64
dtypes: int64(2)
memory usage: 2.1 MB
None

Unique:
userID       1892
artistID    17632
dtype: int64

No duplicates: True
```

Exploratory Data Analysis - 3

```
[ ]  # Britney Spears is the most played at 2.4 million plays
     artist_rank.head()
```

name	totalUniqueUsers	totalArtistPlays	avgUserPlays
Britney Spears	522	2393140	4584.559387
Depeche Mode	282	1301308	4614.567376
Lady Gaga	611	1291387	2113.563011
Christina Aguilera	407	1058405	2600.503686
Paramore	399	963449	2414.659148

Per the total artist plays column we see an average of almost 4,000 plays per artist with a max of 2.4 million (for Britney Spears).

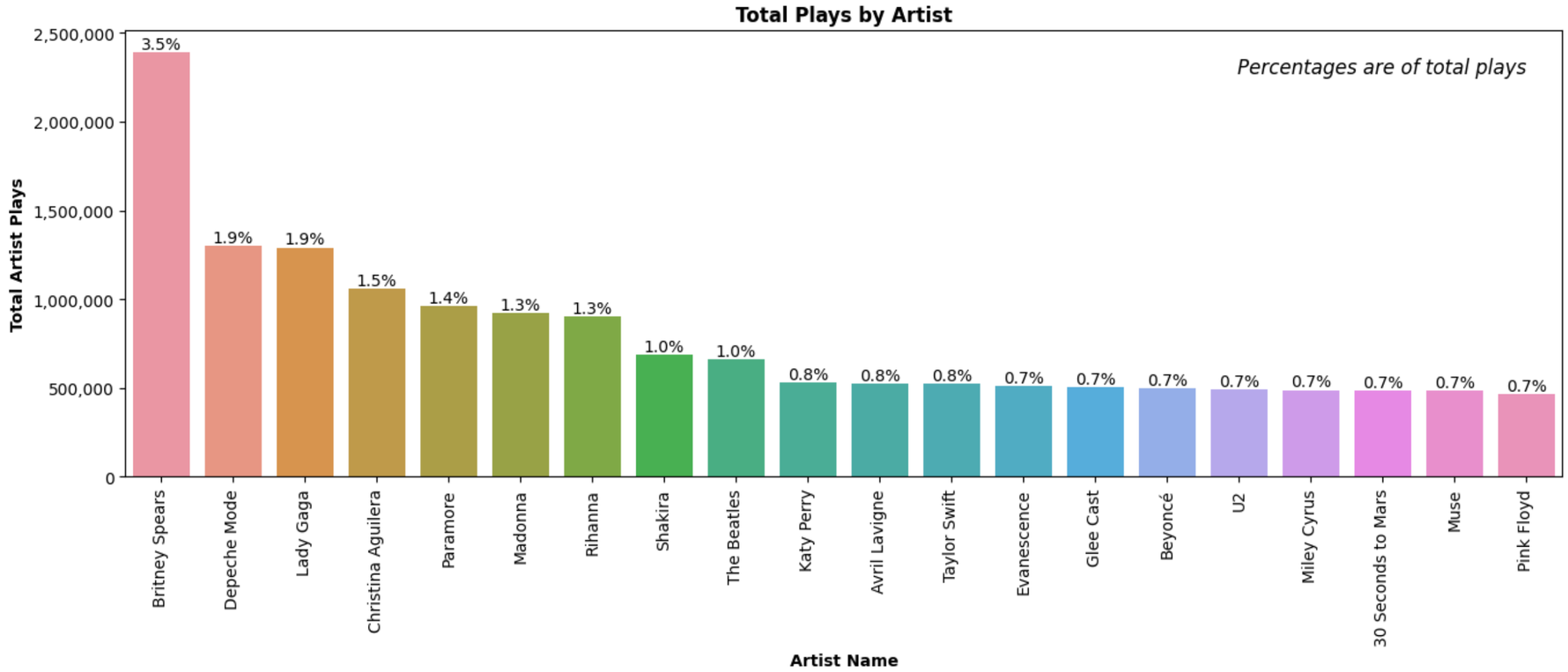
Exploratory Data Analysis - 4

```
user_rank.head()
```

userID	totalUniqueArtists	totalUserPlays
757	50	480039
2000	50	468409
1418	50	416349
1642	50	388251
1094	50	379125

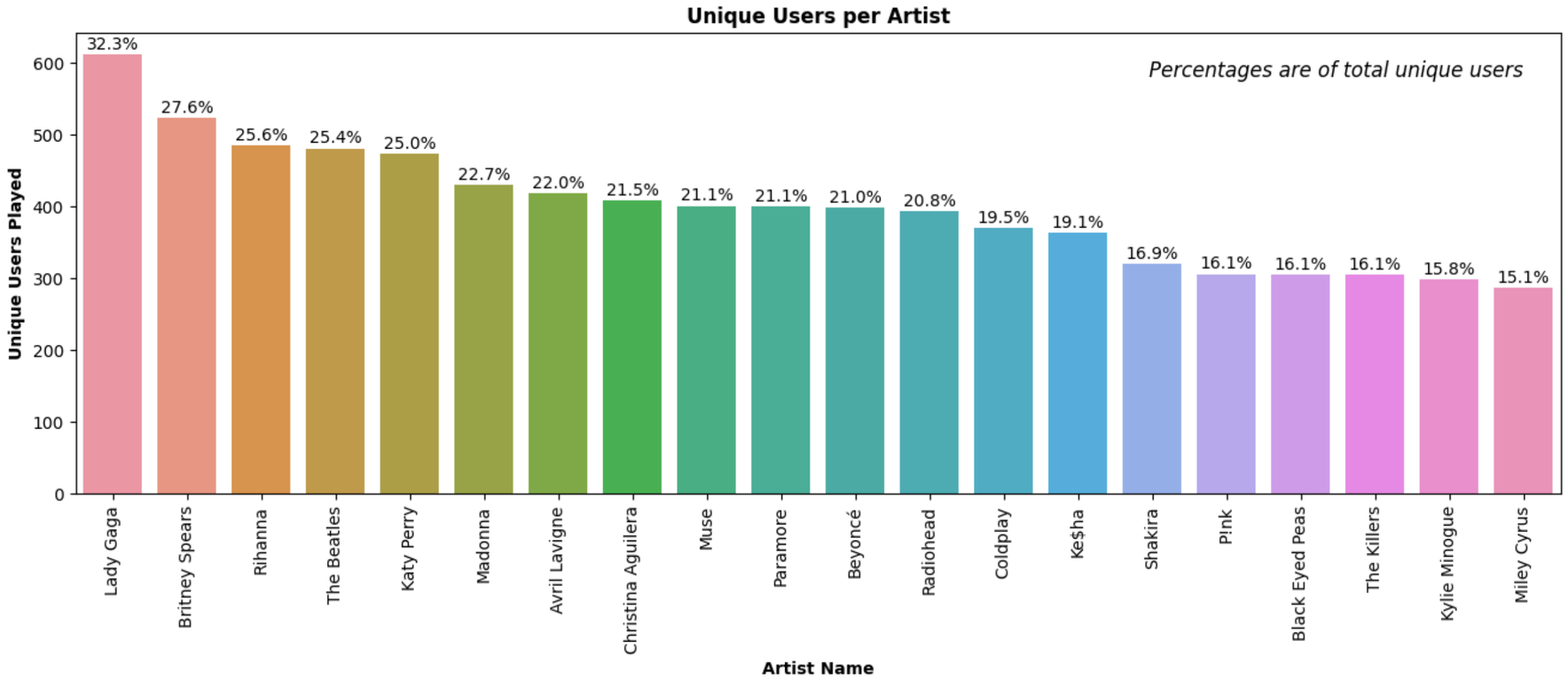
Here we have our top users in terms of total play count. We also see an apparent cap of 50 artists per user.

Exploratory Data Analysis - 5



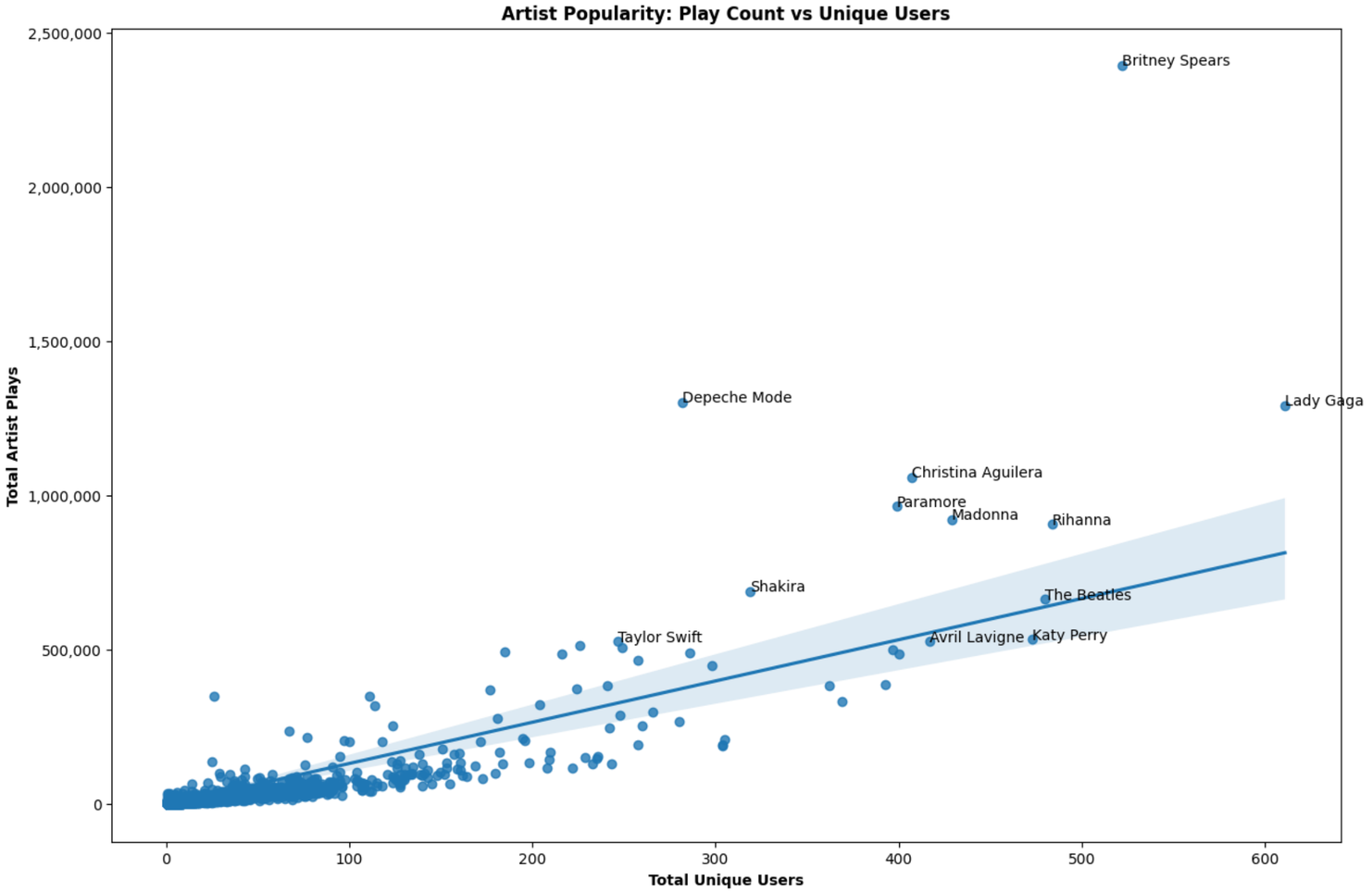
With 2.4 million plays out of a aggregate play count of 69 million in the dataset, Britney Spears received 3.5% of the plays in the dataset.

Exploratory Data Analysis - 6



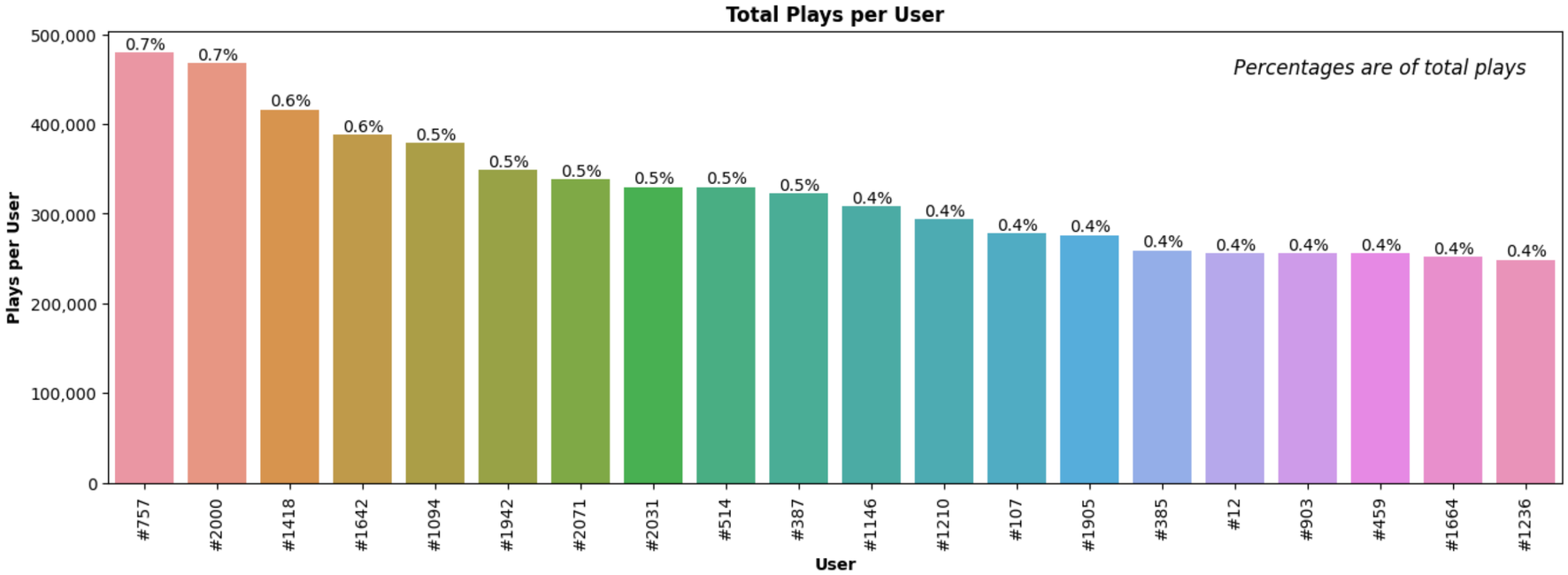
Of the total 1,892 users in this dataset, 611 of them listened to Lady Gaga at least one time. That is 32.3% of all users!

Exploratory Data Analysis - 7



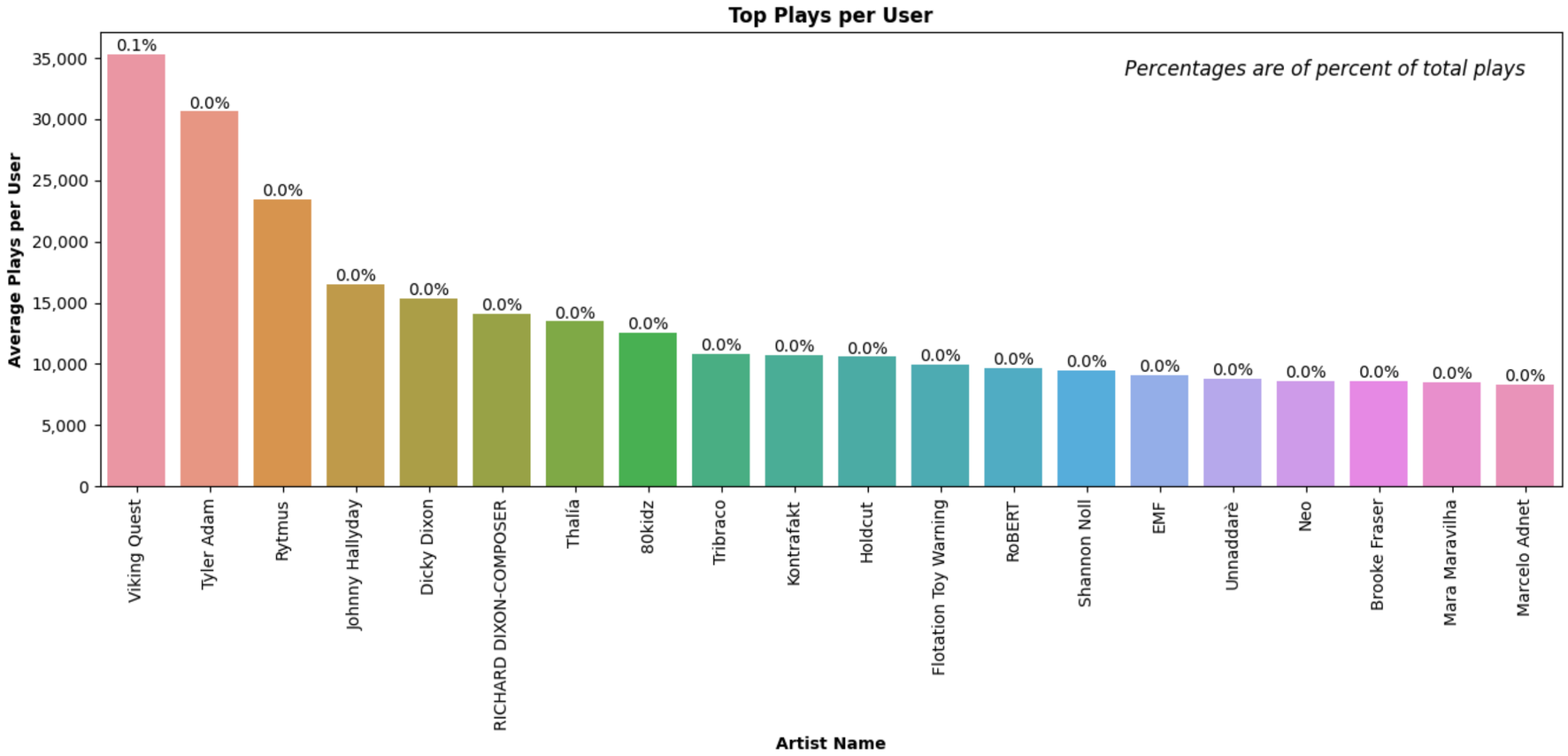
We see general positive correlation between number of artist plays and number of unique users. We see Britney Spears with the most aggregate plays and Lady Gaga with the most unique listeners.

Exploratory Data Analysis - 8



User #757 accounts for almost 1% of all plays in the dataset with almost 500,000 plays.

Exploratory Data Analysis - 9

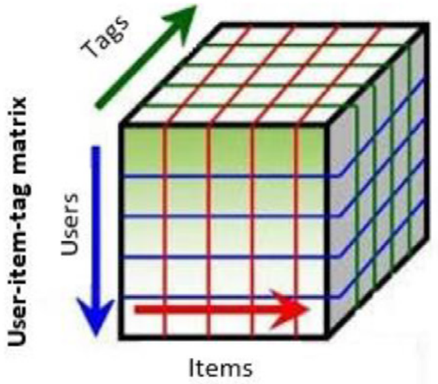


Viking Quest has a single unique user which played them 35,000 times.

Data Preparation

Trust matrix (S)

Users	u_1	u_2	...	$u_{\|U\|}$
u_1	0	1	...	0
u_2	1	0	...	1
...	...	...	...	...
$u_{\|U\|}$	0	1	...	0



User-item matrix (R)

Users	i_1	i_2	...	$i_{\|I\|}$
u_1	1	0	...	1
u_2	1	0	...	0
...	...	...	...	...
$u_{\|U\|}$	0	1	...	1

User-tag matrix (Y)

Users	t_1	t_2	...	$t_{\|T\|}$
u_1	0	1	...	1
u_2	0	1	...	0
...	...	...	...	...
$u_{\|U\|}$	1	0	...	1

Data Preparation

Trust Relationship Matrix, S: Represents whether a trust relationship exists between users u and v.

- S[u, v] = 1 if there is a trust relationship between users u and v.
- S[u, v] = 0 if there is no trust relationship between users u and v.

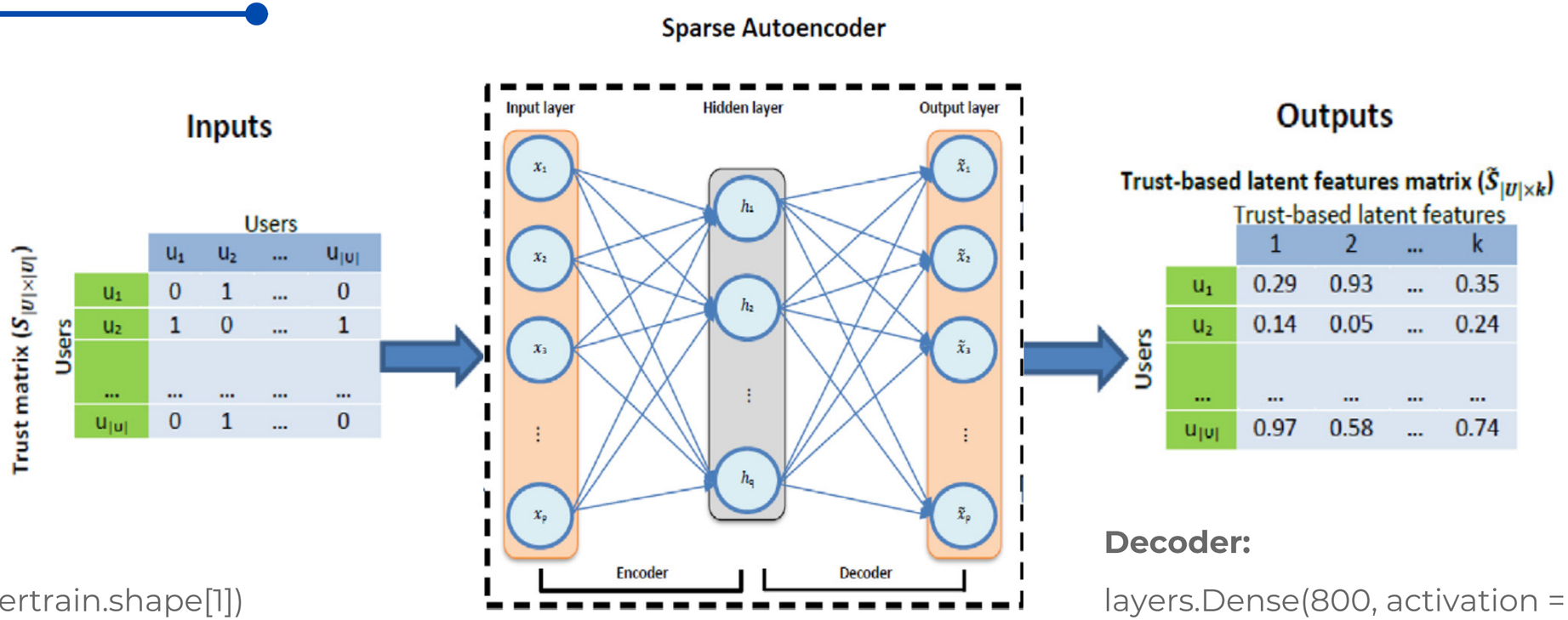
User-Tag Matrix, Y: Represents whether a tag relationship exists between user u and tag T.

- Y[u, t] = 1 if f there is at least one artist (i) associated with the user that has a positive tag value for the tag t .
- Y[u, t] = 0 if none.

User-item Matrix, R: Represents whether a relationship exists between user u and tag T.

- R[u, i] = 1 if f there is at least one tag (t) associated with the user that has a positive artist value for the artist i.
- R[u, i] = 0 if none.

Model 1: User - User



Encoder:

Input(userusertrain.shape[1])

layers.Dense(1000, activation = 'relu'),

layers.Dense(800, activation = 'relu'),

layers.Dense(400, activation='relu',

activity_regularizer=keras.regularizers.L1(0.0001))

Decoder:

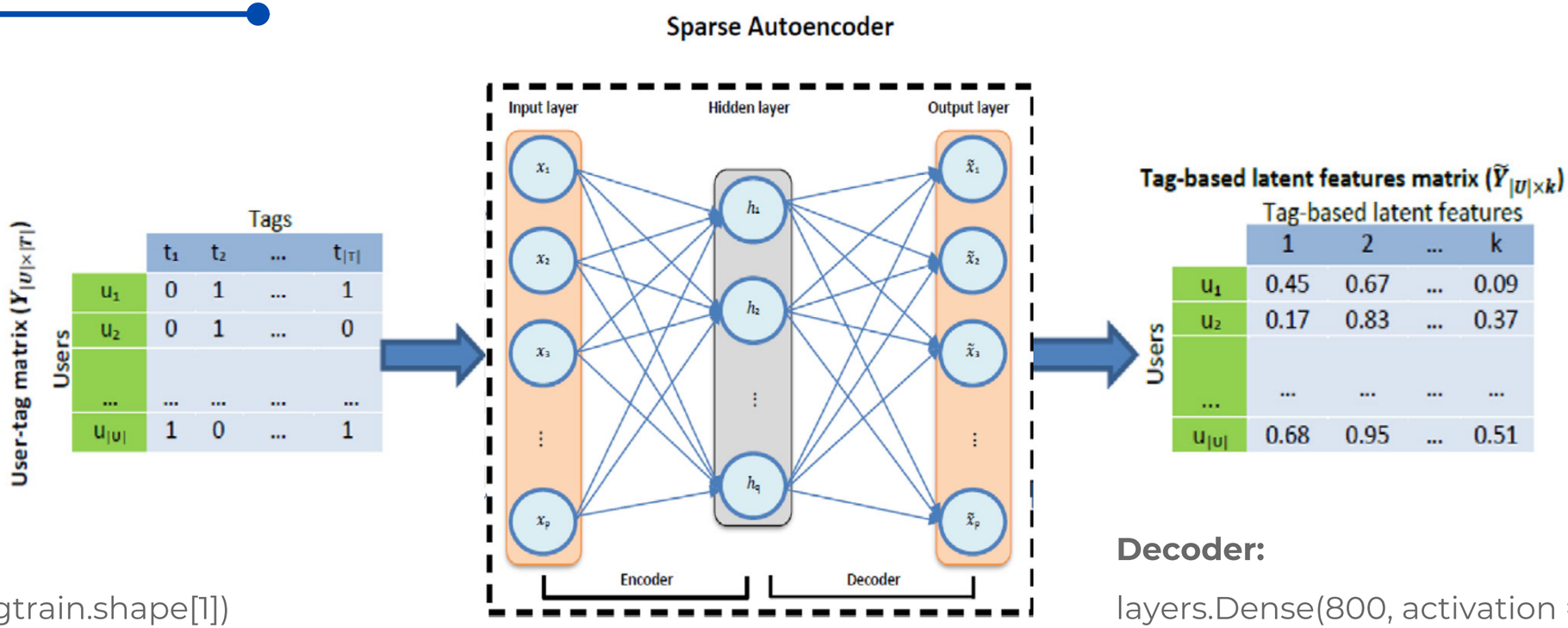
layers.Dense(800, activation = 'relu'),

layers.Dense(1000, activation = 'relu'),

layers.Dense(userusertrain.shape[1],

activation = 'relu')

Model 2: User - Tag



Encoder:

Input(usertagtrain.shape[1])

layers.Dense(1000, activation = 'relu'),

layers.Dense(800, activation = 'relu'),

layers.Dense(400, activation='relu',

activity_regularizer=keras.regularizers.L1(0.0001))

Decoder:

layers.Dense(800, activation = 'relu'),

layers.Dense(1000, activation = 'relu'),

layers.Dense(usertagtrain.shape[1],

activation = 'relu')

Latent Feature Extraction

Sk and **Yk** are the latent features of user u obtained based on the trust and tag vectors, respectively. **k** denotes the number of latent features which is usually much less than the number of users and tags. We choose k = 90.

This method is able to make representations of input data with lower-dimensional in comparison to the raw input data by obtaining the latent features using the sparse autoencoder. Thus, the proposed method alleviates the data sparsity problem and reduces the computational complexity of the recommendation process.

Recomendation

Latent features are utilized to measure similarity values between users.

Scenario 1:

Trust Simu,v represents the similarity value or score that quantifies the level of trust or similarity between users u and v. It measures how much users u and v are alike in terms of trust. Applied cosine similarity on the top 90 latent features extracted from User-User Autoencoder model.

$$Trust\_Sim_{u,v} = \frac{\sum_{i=1}^{k} S_{u,i} \times S_{v,i}}{\sqrt{\sum_{i=1}^{k} \tilde{S}_{u,i}^{2}} \sqrt{\sum_{i=1}^{k} \tilde{S}_{v,i}^{2}}}$$

```
[ ] trust_similarity_cosine = cosine_similarity(useruserk90)
    trust_similarity_cosine.shape
```

Recomendation

Latent features are utilized to measure similarity values between users.

Scenario 2:

Tag Simu,v represents the similarity between users u and v according to the latent features extracted from the user-tag matrix. This similarity metric is based on how the users' latent features align with each other, taking into account the number of features (k= 90) in their latent features vectors.

$$Tag\_Sim_{u,v} = \frac{\sum_{i=1}^{k} \tilde{Y}_{u,i} \times \tilde{Y}_{v,i}}{\sqrt{\sum_{i=1}^{k} \tilde{Y}_{u,i}^{2}} \sqrt{\sum_{i=1}^{k} \tilde{Y}_{v,i}^{2}}}$$

```
[ ]  tag_similarity_cosine = cosine_similarity(usertagk90)
     tag_similarity_cosine.shape
```

Recomendation

Latent features are utilized to measure similarity values between users.

Scenario 3:

Goal is to combine the similarity values obtained from both the latent features of trust relationships (Trust Simu,v) and the latent features of user-tag matrices (Tag Simu,v) to calculate a final similarity value.

- If Beta < 1 , it assigns more weight to Trust Simu,v
- If Beta > 1 , it assigns more weight to Tag Simu,v
- If Beta = 1 , both trust-based and tag-based similarity values have equal importance.

$$Trust\_Tag\_Sim_{u,v} = \frac{(1+\beta^2) \times (Trust\_Sim_{u,v} \times Tag\_Sim_{u,v})}{(\beta^2 \times Trust\_Sim_{u,v}) + Tag\_Sim_{u,v}}$$

```
def combine_similarity(trust_simu, tag_simu, beta):
  numerator = (beta**2 + 1) * np.multiply(trust_simu , tag_simu)
  denominator = (beta**2 * trust_simu) + tag_simu
  combined_similarity = np.divide(numerator ,denominator)
  return combined_similarity

beta_value = 1

result = combine_similarity(trust_similarity_cosine,
                            tag_similarity_cosine, beta_value)
print(result)
```

Prediction

After obtaining the similarity values based on one of the above scenarios, some users with the highest similarity values are determined as the nearest neighbors of the target user. Then, the rating of unseen items can be obtained as follows:

$$P_{u,i} = \frac{\sum_{v \in N_u} Sim_{u,v} \times R_{v,i}}{\sum_{v \in N_u} Sim_{u,v}}$$

```
loc_pred = np.argsort(r,axis=1)[:,-15:]
np.take_along_axis(r, loc_pred, axis=1)
```

```
array([[0.01429185, 0.01429185, 0.01429258, 0.01429528, 0.01429528,
        0.01429568, 0.01429568, 0.01429568, 0.01432292, 0.01432292,
        0.02856714, 0.02856998, 0.02857027, 0.02860906, 0.05715081],
       [0.0428525 , 0.04287032, 0.0428714 , 0.0428714 , 0.04287242,
        0.04287242, 0.04287242, 0.04290412, 0.04291394, 0.04291394,
        0.05713471, 0.05714663, 0.05716196, 0.05716845, 0.07147947],
       [0.01431359, 0.01431359, 0.01431359, 0.01431359, 0.01431359,
        0.01431359, 0.01431359, 0.01431359, 0.01431359, 0.01431359,
        0.01431359, 0.02856119, 0.02856315, 0.02856944, 0.0285771 ],
       [0.14290526, 0.1429545 , 0.14296367, 0.1572107 , 0.17147711,
        0.17148233, 0.1857133 , 0.1858115 , 0.2000614 , 0.21430098,
        0.2143196 , 0.21435646, 0.22864835, 0.24291891, 0.2572156 ],
       [0.01430129, 0.01430129, 0.01430129, 0.01430502, 0.01430532,
        0.01430532, 0.01430782, 0.0143119 , 0.01431431, 0.01434253,
        0.0285527 , 0.02856238, 0.02857268, 0.02858476, 0.04287352]])
```

TOP 15 RECOMENDATIONS FOR 5 USERS

Thank You