

Music Recommendation System Project

Introduction

The music industry witnessed a significant shift from selling physical CD's to a cloud-based streaming model, exemplified by services like Spotify. This transformation took over an era where online downloading was the primary way of listening to a wide range of music, often through platforms like The Pirate Bay and LimeWire. With the introduction of online streaming, the music industry has grown to be a massive industry worth \$20 Billion in 2023 and there is an increased focus on developing more complex song recommendation systems, which drive revenue through subscriptions.

Creating an effective recommendation system presents multiple challenges. A primary issue is the complexity of determining song similarity, as songs that appear similar might appeal to different audiences. This requires the identification of relevant data to suggest suitable songs for users accurately. Another challenge is in selecting the right song features for the learning algorithm, as appropriate feature selection can enhance the recommendation system significantly. To tackle these challenges, we utilized the Million Song Dataset (MSDS) and EchoNest user play history. Our approach centered on employing various algorithms, specifically focusing on the Alternating Least Squares (ALS) model for matrix factorization using user listening data, and the Word2Vec, TF-IDF, and Latent Dirichlet Allocation (LDA) models for predicting song recommendations based on lyrics data. These techniques allowed us to develop recommendation systems that can consider both user behavior and the thematic and semantic content of song lyrics.

The size of the dataset we worked with was substantial, requiring cloud computational resources. We utilized Amazon EMR (Elastic MapReduce) clusters and Amazon S3 storage for our data processing and analysis. This approach was essential as the size of the dataset far exceeded the memory capacities of our machines. Utilizing cloud computing resources allowed us to efficiently process the large dataset and train our models.

Dataset and Analysis

For our project, we primarily used the Million Song Dataset (MSD), developed by the Laboratory for the Recognition and Organization of Speech and Audio at Columbia University. This dataset offers a comprehensive collection of metadata and audio features for one million songs, encompassing a diverse range of genres, periods, and countries. We also incorporated additional data linked to the MSD songs, including lyrics from musixMatch, user-generated tags from Last.fm, and listening data from the EchoNest. These supplementary datasets provided richer context and depth to our analysis.

User Listening Data

Our project leverages a set of "triplets" data extracted from the Million Song Dataset, encompassing listening information from one million users. Each triplet consists of a User ID, Song ID, and the Number of Plays, forming the backbone of our user listening data. This comprehensive dataset allows us to analyze and understand user listening habits and preferences, forming the foundation of our collaborative filtering recommendation system.

Song Metadata

Alongside the user listening data, we utilize detailed metadata for each song in the dataset. This metadata includes vital information such as the song title, artist name, artist familiarity and hotness. The inclusion of these details provides a more comprehensive understanding of each song, enabling a more in depth analysis that goes beyond just play counts.

Lyrics

To look at the content of the songs, we incorporate lyrics data sourced from musixmatch. This dataset offers an extensive collection of song lyrics in a bag-of-words format. By analyzing these lyrics, we gain insights into the themes, moods, and stylistic elements of the songs. This lyrical analysis adds another dimension to our recommendation system, allowing for more nuanced and personalized song recommendations based on lyrical content.

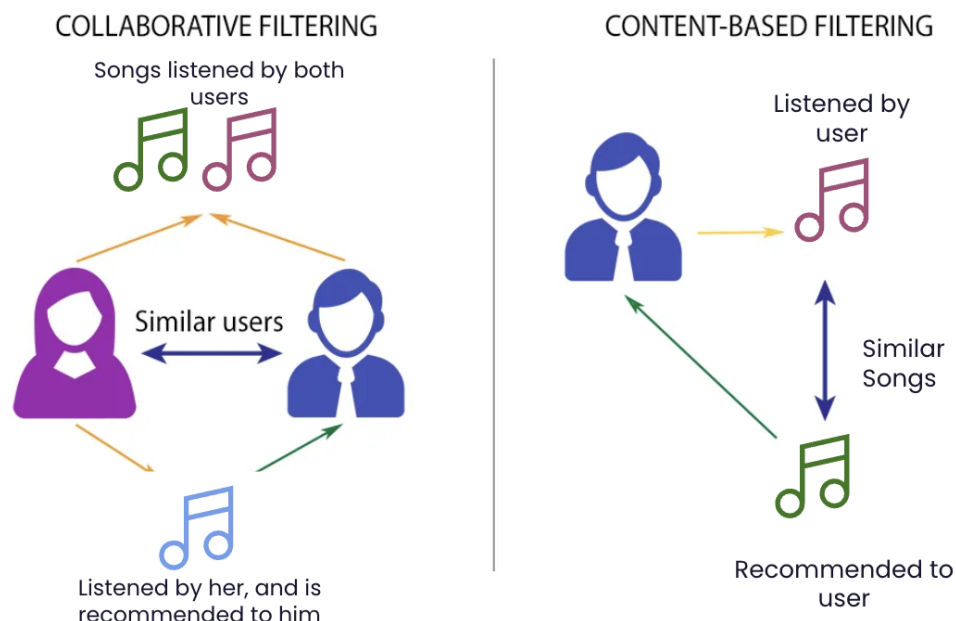
Methodology and Algorithms

Collaborative Filtering

Collaborative Filtering (CF) is a technique used in recommendation systems that makes automatic predictions (filtering) about the interests of a user by collecting preferences from many users in the system (collaborating). This method assumes that if person A has the same opinion as person B on an issue, A is more likely to have B's opinion on a different issue than that of a random person, this is highlighted in the figure below. For our project, we are using the Alternating Least Squares (ALS) algorithm, which is a form of matrix factorization. It works by decomposing the original user-item interaction matrix into two lower-dimensional matrices - one representing latent user preferences and the other representing latent item attributes. The ALS algorithm alternatively fixes one matrix and solves for the other. Mathematically, it tries to minimize the loss function:

$$\text{Loss} = \sum_{u,i} (R_{ui} - U_u^T I_i)^2 + \lambda (\|U_u\|^2 + \|I_i\|^2)$$

where R_{ui} is the user-item interaction matrix, U_u and I_i are the latent factors for users and items, and λ is the regularization parameter.



Alternating Least Squares (ALS) Algorithm - Matrix Factorization

We started by preparing our dataset, ``raw_plays_df_with_int_ids``, for the ALS model. The dataset was randomly split into three parts: 60% for training, 20% for validation, and 20% for testing, using a seed for reproducibility. We also ensured the data types were appropriate for the ALS algorithm, casting 'Plays' to ``DoubleType`` and 'new_songId', 'new_userId' to ``IntegerType``.

Model Initialization and Tuning

We initialized the ALS model with specific parameters:

- Maximum Iterations: 5
- Seed: 42
- Item Column: 'new_songId'
- Rating Column: 'Plays'
- User Column: 'new_userId'

The model's hyperparameters - rank and regularization parameter - were tuned using the validation dataset. We experimented with different ranks (4, 8, 12, 16) and regularization parameters (0.15, 0.2, 0.25). The Root Mean Square Error (RMSE) was used to evaluate the performance of each model configuration. Through iterative testing, we identified the best combination of rank and regularization parameter that minimized the RMSE. This optimal model was then finalized with the best-found parameters.

To establish a baseline for comparison, we calculated the average number of plays from the training dataset. This average plays value was then used to create a baseline model against which the performance of our ALS model could be compared. Finally, we evaluated our ALS model and the baseline model on the test dataset using the RMSE metric. This provided us with a quantitative measure of our model's performance in predicting song plays.

Content Based Filtering

Content-Based Filtering (CBF) is a recommendation approach that uses item features to recommend additional items similar to what the user likes, based on their previous actions or explicit feedback. Unlike CF, which relies on user-item interactions, CBF uses specific characteristics of an item to recommend new items. In our project, methods like TF-IDF, Word2Vec, and LDA are used to recommend new songs.

LSH for Approximate Nearest Neighbor Search

The primary technique used here for making predictions is Bucketed Random Projection for LSH. LSH is ideal for large-scale, high-dimensional data as it efficiently approximates nearest neighbor searches. We have chosen to use Locality-Sensitive Hashing (LSH) instead of cosine similarity because, although cosine similarity is a reliable method for comparing two feature vectors, it becomes significantly resource-intensive as the dataset size increases. LSH offers a more scalable alternative for large datasets. LSH is used to find songs with similar features (lyrics, artist, title, artist similarity) to a given target song. The similarity is determined based on the "distance" between feature vectors, with closer vectors indicating more similar songs.

TF-IDF

The TF-IDF model emphasizes the uniqueness of song lyrics. By tokenizing the lyrics and applying HashingTF, it converts the lyrics into numerical vectors, with each word's importance in a song adjusted by its frequency across all songs. Applying IDF (Inverse Document Frequency) further refines these vectors, emphasizing words unique to particular songs. This model is particularly effective in distinguishing songs based on their lyrical content. The final step involves scaling these features and applying LSH (Locality-Sensitive Hashing) for efficient similarity searches, making it ideal for recommending songs with similar or unique lyrical themes.

TF(word) measures how frequently a term (word) occurs in a document (song). The formula is:

$$TF(word) = \frac{(Number\ of\ times\ the\ word\ appears\ in\ a\ song)}{(Total\ number\ of\ words\ in\ the\ song)}$$

This formula normalizes the raw count of the word in the song lyrics, preventing it from favoring longer songs where the word might appear more frequently just due to the song's length.

Inverse document frequency measures the importance of the term across a corpus of song lyrics. The idea is that words that appear frequently in many songs are less informative than those that appear less frequently. The formula is:

$$IDF(word) = \log\left(\frac{(Total\ number\ of\ songs)}{(Number\ of\ songs\ containing\ the\ word)}\right)$$

TF-IDF is the dot product of TF and IDF:

$$TF - IDF(word) = TF(word) \times IDF(word)$$

This value gives a numerical representation of how important a word is to a song in a collection. A high TF-IDF score for a word in a song indicates that the word is both frequent in that specific song and rare across the entire collection of songs. This makes TF-IDF an effective way to vectorize text and to filter out common words that are less informative about the song's content.

Word2Vec

The Word2Vec model, in contrast, captures the contextual relationships between words in lyrics. It transforms lyrics into word tokens and then into vectors, where semantically similar words are mapped to proximate points in the vector space. This approach is adept at understanding the underlying themes and nuances in lyrics, beyond mere word presence. By scaling and applying LSH to these Word2Vec features, the model can recommend songs that are contextually and semantically related, offering recommendations that resonate with the thematic essence of a user's preferred songs.

We utilized the Word2Vec model to capture the contextual relationships between words in song lyrics. Unlike methods that focus on presence or frequency of words, Word2Vec understands the semantic relationships within the lyrics. This approach allows us to map semantically similar words to proximate points in a multi-dimensional vector space, by doing so encapsulating the subtle thematic and contextual essence of the songs. The steps to achieve this is shown below:

Tokenization

The first step involves transforming the lyrics into individual word tokens. This is achieved by splitting the lyrics into words, considering each word as a basic unit for analysis. We use PySpark's `split` function from `pyspark.sql.functions` to accomplish this, ensuring a thorough breakdown of the lyrical content.

Vectorization

The Word2Vec model then takes these tokens and translates them into numerical vectors. Each vector represents a word in a 100-dimensional space (`vectorSize=100`), where the positioning is determined by the word's contextual usage across the entire lyrics dataset. We set the `windowSize` to 5, allowing the model to consider the context within the proximity of five words around each target word.

Model Training

After defining the Word2Vec model, it is trained on the tokenized lyrics to generate word vectors. This step involves the internal mapping of words to vectors, capturing the nuanced semantic relationships between them.

LDA

We also used Latent Dirichlet Allocation (LDA) for topic modeling, combined with Locality-Sensitive Hashing (LSH), as a content-based filtering approach. This content based method captures the thematic essence of songs through their lyrics but also matches users with songs sharing similar thematic content. Recommendations from this model can be used to compare to the previous TF-IDF and Word2Vec methods.

Preprocessing and Tokenization

The initial step involves processing the 'lyrics' column of our data frame. The lyrics are tokenized into individual words using PySpark's `split` function. This tokenization breaks down the lyrics into manageable units (words).

Vectorization with CountVectorizer

We employ `CountVectorizer` to convert these tokens into a numerical format. It creates a vector representation of the lyrics where each vector has a dimension of up to 10,000 (`vocabSize=10000`), representing the most frequent words across the corpus. Words appearing in at least 5 documents (`minDF=5`) are considered, ensuring a focus on relevant terms while filtering out rare words.

LDA for Topic Discovery

The LDA model is then applied to these count vectors. LDA, a popular topic modeling technique, discovers latent topics within the lyrics. Each song is represented as a mixture of various topics, where a topic is a collection of dominant words. We set the number of topics to 10 (`num_topics=10`), although this can be adjusted based on the dataset. Increasing the number of topics can lead to a large increase in computation time which can be a limiting factor.

Topic Distribution as Features

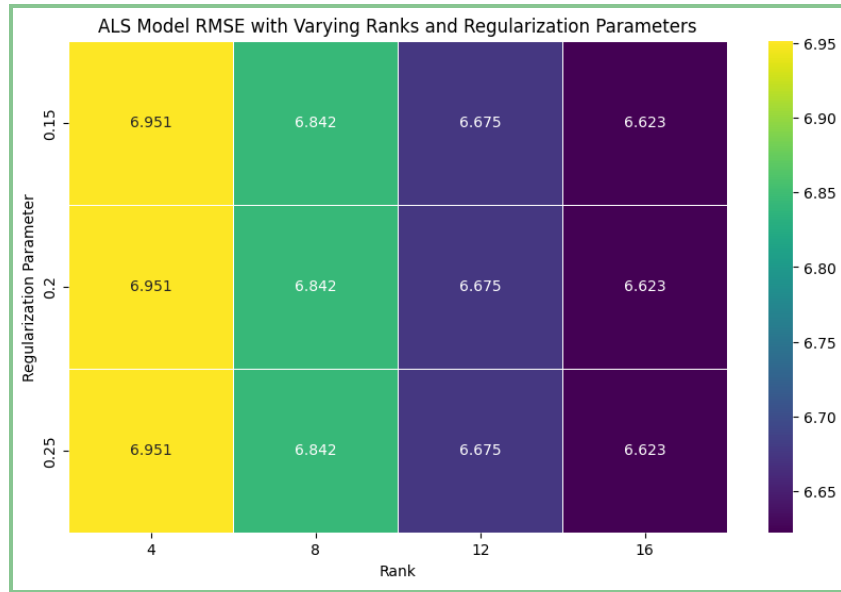
The LDA model outputs a 'topic distribution' for each song, indicating the presence and proportion of each topic within the song. This distribution serves as a feature vector, encapsulating the thematic essence of the lyrics.

Each model, TF-IDF, Word2Vec, and LDA provides a unique lens through which to analyze and recommend songs. TF-IDF highlights lyrical uniqueness, Word2Vec captures semantic nuances, and LDA reveals thematic patterns. In a music recommendation system, these models can be used either independently or in combination to cater to different aspects of user preferences.

Results

ALS -Content Based Filtering

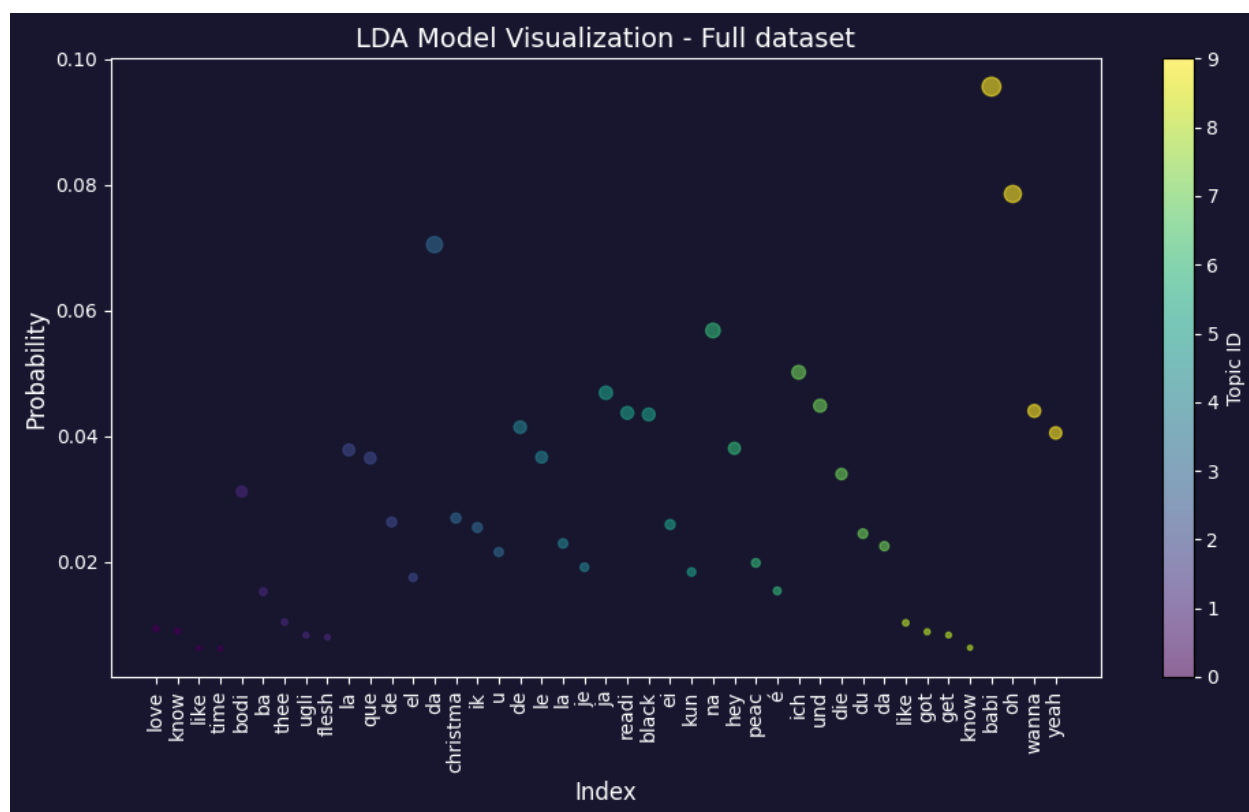
The model was trained using a large dataset consisting of 1,742,149 data points, which represent user-item interactions. For validation, a subset of the data containing 580,227 instances was used to fine-tune the model's parameters, ensuring that the model can generalize well. The model's performance was then assessed on a test set comprising 579,449 instances, which represents unseen data. Regarding the model's optimal parameters, a regularization parameter of 0.25 was the best fit for preventing overfitting, a common problem where the model learns the noise in the training data rather than the actual pattern. Additionally, the model used a rank of 16, which refers to the number of latent factors that characterize user and item interactions. The rank defines the dimensionality of the user and item vectors in the model, and a higher rank allows the model to identify more nuanced patterns in the data. We found that the average number of plays per song in the dataset was 3.0, indicating that users, on average, played each song three times. This metric gives an idea of user engagement and helps in understanding the scale of interactions.



The Root Mean Square Error (RMSE), a standard metric for evaluating the accuracy of a predictive model, was recorded at approximately 6.67 for this model. The RMSE quantifies the average magnitude of the model's prediction errors, meaning the difference between the number of times the model predicts a song will be played and the actual number of plays. An RMSE of 6.67 suggests that the model's predictions deviate from the actual play counts by an average of 6.67 plays. While the RMSE gives an indication of the model's prediction error, without a benchmark or comparison to other models, it's difficult to fully assess the efficacy of this value. However, it provides a starting point for understanding the model's performance and areas where it might be improved.

LDA

The LDA model was used to identify topics within a collection of songs for a recommendation system based on content filtering. We were able to visualize the topics and the probability of the song belonging to the topic.



The colors represent different topics that the LDA model has identified in the dataset. The position of the dot on the x-axis (labeled as "Index") corresponds to individual songs, while the y-axis represents the "Probability," which is the probability of the song belonging to the topic designated by its color. The color bar on the right side indicates the topic ID with a scale, showing a range of topics from 0 to 9. Larger dots indicate a higher probability that a particular song strongly associates with the topic represented by the color. For example, songs with larger yellow dots might be closely related to a certain theme or set of words that define Topic ID 9.

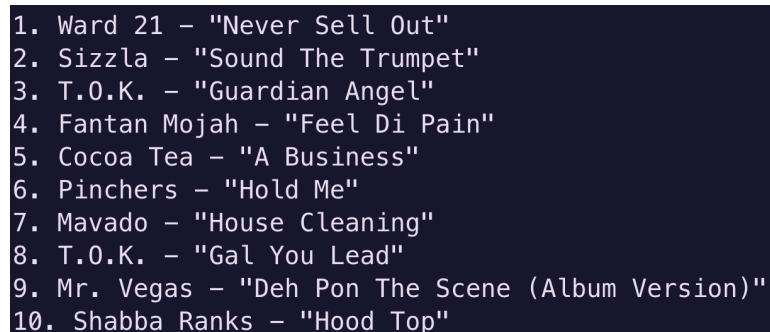
The model has identified ten topics within the song dataset, each characterized by a distinct set of keywords that suggest thematic content. Topics 0 and 8 are predominantly English, with a strong emphasis on common terms such as "love," "know," and "like," indicating mainstream and romantic themes. Topics 2 and 4 are characterized by Spanish and French words, respectively, indicating substantial clusters of songs in these languages, which are significant for users with language-specific preferences. Topic 7 contains German words which could suggest songs in German or other languages similar to it.

Word2Vec and TF-IDF

We also utilized two distinct models: Word2Vec and TF-IDF. Our strategy involved predicting song recommendations for users who have no prior listening history. Both the Word2Vec and TF-IDF models were employed to analyze and find songs similar to a single track or a set of genres initially chosen by the user. Despite their different approaches to text analysis and feature extraction, both models yielded identical recommendations. This convergence in results shows the effectiveness of our method in generating relevant initial song suggestions. These suggestions can then serve as a foundation for gathering user listening data, which is very important for refining future recommendations through ALS.

Input Song: Super Cat - "Trash and Ready"

Predicted 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"
```

Since we do not have ground truth labels to identify which of these recommendations are relevant or even genre tags associated with these songs. We had to manually check these songs on YouTube or Spotify in relation to the input song. What we found is that the input song is of the reggae genre and all of the 10 predicted songs were also reggae, suggesting that our model worked and it was able to find similar songs based on the content of the songs.

Limitations and Further Work

One significant challenge in evaluating the effectiveness of the content-based filtering methods like LDA, Word2Vec, and TF-IDF in this project is the absence of ground truth labels. Without these labels, it's difficult to objectively assess the accuracy and relevance of the recommendations provided by the models. Ground truth labels serve as a benchmark, against

which the model's outputs can be compared. Without these labels, determining whether the recommended songs truly align with the user's preferences or the characteristics of the input song becomes a subjective process, often reliant on manual verification. This manual approach is not only time-consuming but also introduces a level of human bias, as the assessment is based on personal judgment rather than a quantifiable metric. Therefore, establishing a method for generating or acquiring ground truth labels, perhaps through user feedback in combination with collaborative filtering techniques as a hybrid system would be a critical step towards enhancing the accuracy and reliability of these techniques.

Conclusion

In conclusion, this project looked at collaborative and content based filtering for music recommendation, including TF-IDF, Word2Vec, LDA, and ALS, to create an advanced music recommendation system. Each method brought unique strengths, with TF-IDF focusing on lyrical uniqueness, Word2Vec capturing contextual relationships, and LDA identifying thematic patterns in songs. The ALS model was used to establish a collaborative filtering technique to recommend songs, this method can be used in hybrid with content based filtering in order to tackle the cold start problem. However, the absence of ground truth labels posed a challenge in objectively evaluating the system's effectiveness. This underscores the need for integrating user feedback and collaborative filtering or access to more detailed user interaction data in future work to enhance recommendation accuracy.

Despite these challenges, the project demonstrated the potential of combining content-based and collaborative filtering methods in music recommendation. The ability to recommend songs across various models, especially in genre-specific contexts, highlighted the system's effectiveness. As a step forward, refining these models with user feedback could lead to a more personalized and accurate music discovery experience and also a way to implement ground truth labels as a way to evaluate the model further rather than manual inspection. We have learned a lot during this project, being able to handle multiple large datasets and using the latest machine learning techniques to recommend songs to users. We have developed a foundational understanding of how to implement a recommendation system with the help of AWS cloud computing services and we aim to build on this project in the future to potentially combine our algorithms and deploy an app to recommend songs based on any input song or a list of songs, or successfully deploy other models like neural collaborative filtering.

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

Liao, Kevin. "Prototyping a Recommender System Step by Step Part 2: Alternating Least Square (ALS) Matrix Factorization in Collaborative Filtering." Towards Data Science, November 17, 2018.

Vatsal. "Word2Vec Explained: Explaining the Intuition of Word2Vec & Implementing it in Python." Towards Data Science, Jul 29, 2021.

Kulshrestha, Ria. "A Beginner's Guide to Latent Dirichlet Allocation (LDA): A statistical model for discovering abstract topics, aka topic modeling." Towards Data Science, Jul 19, 2019.

Scott, William. "TF-IDF from scratch in Python on a real-world dataset." Towards Data Science, Feb 15, 2019.

