FindYourAnime: Anime Recommendation System Report + Code

Introduction

The objective of this project is to develop a recommendation system, FindYourAnime, which helps users discover new and exciting anime titles that align with their unique preferences, beyond mainstream suggestions. This system utilizes collaborative filtering and content-based filtering techniques to provide personalized recommendations. Additionally, a user-friendly web application is developed using Gradio to deliver these recommendations interactively.

Problem Statement:

Despite the vast array of anime titles available, many enthusiasts struggle to discover content that resonates with their unique tastes and preferences. Mainstream recommendation systems often overlook niche or lesser-known titles, leading to frustration and a limited viewing experience for users seeking something different. Additionally, existing platforms may lack the depth and personalization needed to truly understand individual preferences, resulting in generic recommendations that fail to capture the essence of what users enjoy. This gap in personalized and diverse anime recommendations highlights the need for an innovative solution that combines advanced recommendation techniques with user-friendly interfaces to empower anime enthusiasts to explore, discover, and connect with anime titles that truly align with their interests and preferences.

Dataset

MyAnimeList Dataset

The dataset used in this project is sourced from MyAnimeList, a comprehensive database for anime titles, including user ratings, reviews, genres, themes, and other metadata.

Anime Dataframe:

- anime_id (int64): Unique identifier for each anime.
- name (object): Title of the anime.
- genre (object): Genre(s) associated with the anime.

- type (object): Type of anime (e.g., TV, Movie).
- episodes (object): Number of episodes.
- rating (float64): Average user rating.
- members (int64): Number of members who have added the anime to their list.

Ratings Dataframe:

- user_id (int64): Unique identifier for each user.
- anime_id (int64): Unique identifier for each anime.
- rating (int64): Rating given by the user to the anime.

Data Preparation

Merging Dataframes

The anime_data and rating_data dataframes are merged on the anime_id to create a comprehensive dataframe (anime_fulldata) containing all necessary information.
anime_fulldata = pd.merge(anime_data, rating_data, on='anime_id', suffixes=['', '_user'])
anime_fulldata = anime_fulldata.rename(columns={'name': 'anime_title', 'rating_user': 'user_rating'})

Cleaning Data

A text cleaning function is applied to the anime_title column to remove unwanted characters and HTML entities.

import re
def text_cleaning(text):
text = re.sub(r'&quot;', '', text)
text = re.sub(r'.hack//', '', text)
text = re.sub(r'&#039;', '', text)
text = re.sub(r'A&#039;s', '', text)

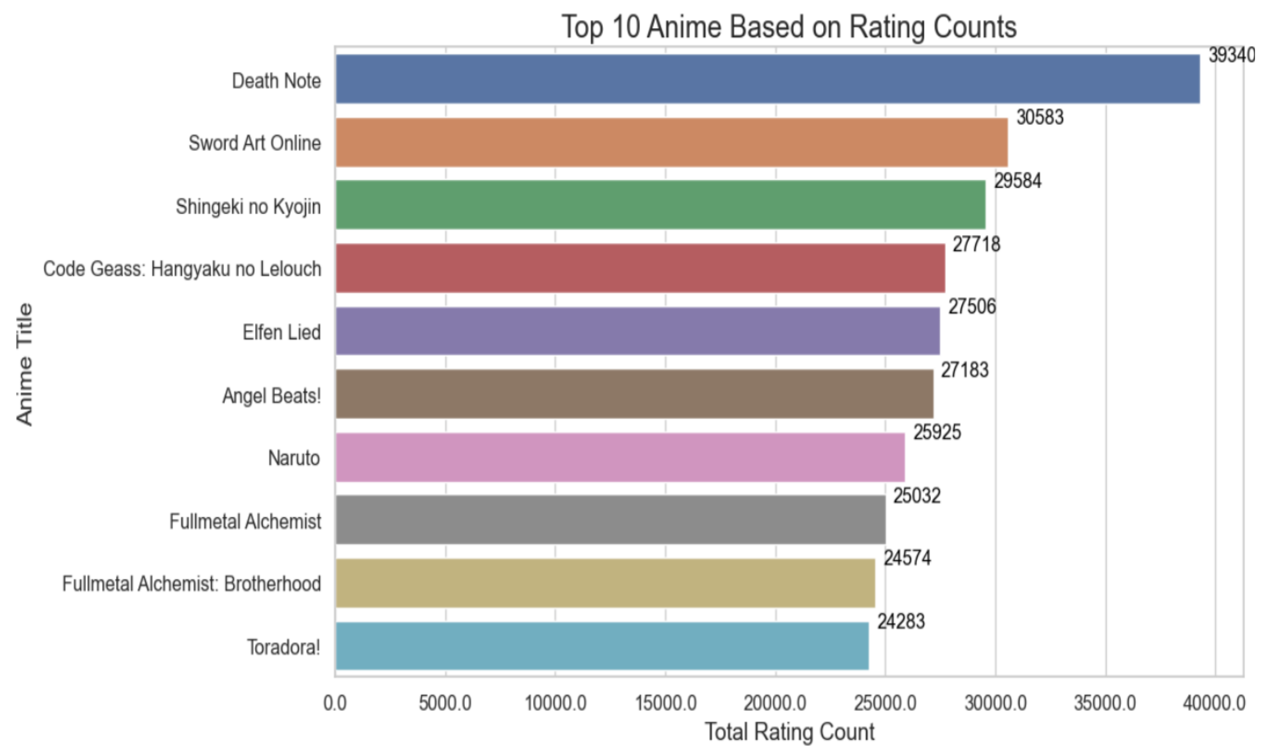
```
text = re.sub(r'I&#039;', 'I\'', text)
text = re.sub(r'&amp;', 'and', text)
return text
anime_data['name'] = anime_data['name'].apply(text_cleaning)
```

Exploratory Data Analysis (EDA)

Top 10 Anime Based on Rating Counts

A bar plot is created to visualize the top 10 anime based on the number of user ratings.

```
import matplotlib.pyplot as plt
import seaborn as sns
combine_anime_rating = anime_fulldata.dropna(axis=0, subset=['anime_title'])
anime_ratingCount =
(combine_anime_rating.groupby(by=['anime_title'])['user_rating'].count().reset_index()
.rename(columns={'user_rating': 'totalRatingCount'}))
top10_animerating = anime_ratingCount[['anime_title',
'totalRatingCount']].sort_values(by='totalRatingCount', ascending=False).head(10)
plt.figure(figsize=(12, 8))
ax = sns.barplot(x="anime_title", y="totalRatingCount", data=top10_animerating,
palette="viridis")
ax.set_xticklabels(ax.get_xticklabels(), fontsize=11, rotation=40, ha="right")
ax.set_title('Top 10 Anime based on rating counts', fontsize=22)
ax.set_xlabel('Anime', fontsize=20)
ax.set_ylabel('User Rating count', fontsize=20)
plt.show()
```



This graph shows top 10 animes based on the total rating points it has earned. We see some of the famous anime titles like Death Note, Naruto and Fullmetal.

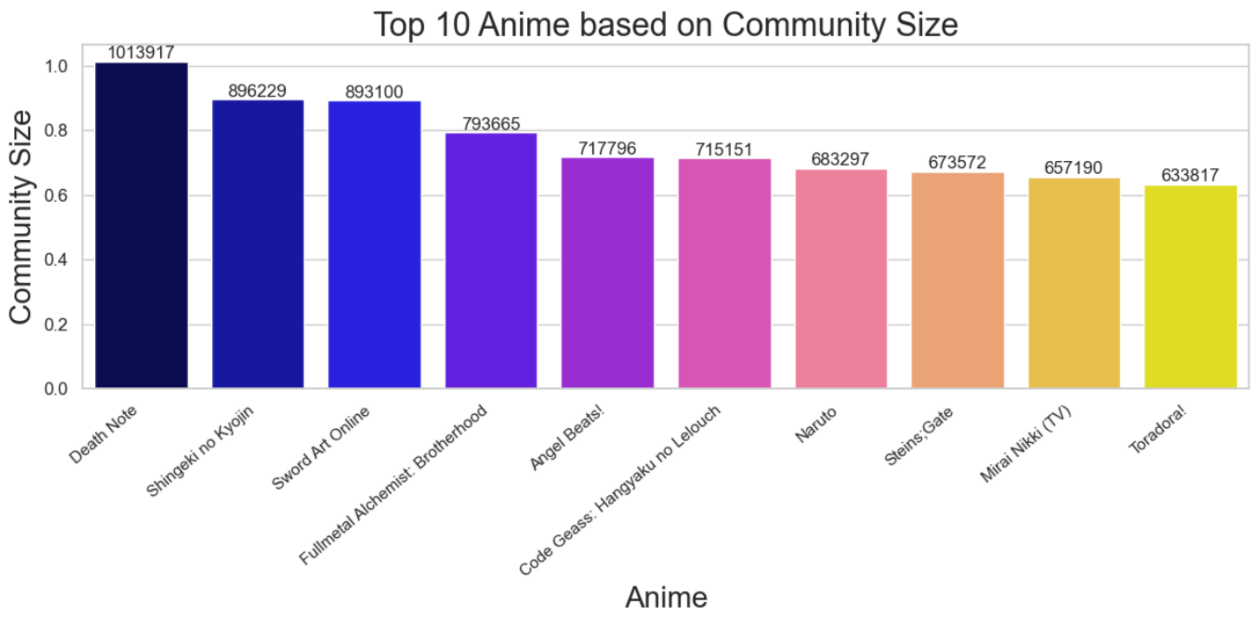
Top 10 Anime Based on Community Size

A bar plot is created to visualize the top 10 anime based on the number of community members.

```
duplicate_anime = anime_fulldata.copy()
duplicate_anime.drop_duplicates(subset="anime_title", keep='first', inplace=True)

top10_animemembers = duplicate_anime[['anime_title', 'members']].sort_values(by='members', ascending=False).head(10)

plt.figure(figsize=(12, 8))
ax = sns.barplot(x="anime_title", y="members", data=top10_animemembers, palette="gnuplot2")
```

```
ax.set_xticklabels(ax.get_xticklabels(), fontsize=11, rotation=40, ha="right")
```

```
ax.set_title('Top 10 Anime based on members', fontsize=22)
ax.set_xlabel('Anime', fontsize=20)
ax.set_ylabel('Community Size', fontsize=20)
plt.show()
```



Yet again the community size of death note is largest and is followed by other famous anime like attack on titan and fullmetal and naruto.

Building the Recommendation System

Content-Based Filtering

A TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer is used to convert the genre information into a matrix of TF-IDF features.

TF-IDF (Term Frequency-Inverse Document Frequency) is essential in content-based filtering scenarios, such as recommending anime based on genre, because it helps capture the relative importance of terms (in this case, genres) within individual anime titles. Here's why TF-IDF is necessary:

Importance Weighting:

- TF-IDF assigns weights to each term based on how frequently they appear in a document (anime title) relative to the entire corpus of documents (all anime titles). This ensures that more common genres (like "Action" or "Comedy") have lower weights compared to less common ones, making the recommendations more nuanced and reflective of the specific interests of users.

- Dimensionality Reduction: By converting genre text data into TF-IDF vectors, we transform the categorical genre information into numerical representations. This enables us to perform calculations and comparisons more efficiently and effectively, especially in scenarios with large datasets.

- Relevance Ranking: TF-IDF helps in identifying the genres that are most relevant to each anime title. Genres with higher TF-IDF scores are considered more discriminative and indicative of the anime's content. This allows for more accurate matching between user preferences and anime recommendations

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfv = TfidfVectorizer(min_df=3, max_features=None, strip_accents='unicode', analyzer='word',
token_pattern=r'\w{1,}', ngram_range=(1, 3), stop_words='english')

anime_data['genre'] = anime_data['genre'].fillna('')
genres_str = anime_data['genre'].str.split(',').astype(str)
tfv_matrix = tfv.fit_transform(genres_str)
```

Similarity Calculation

The linear kernel is used to calculate the cosine similarity between the TF-IDF vectors.

```
from sklearn.metrics.pairwise import linear_kernel
```

```
linear = linear_kernel(tfv_matrix, tfv_matrix)
```

Collaborative Filtering with k-Nearest Neighbors

A k-Nearest Neighbors (k-NN) model is used to find anime similar to a given anime based on user ratings.

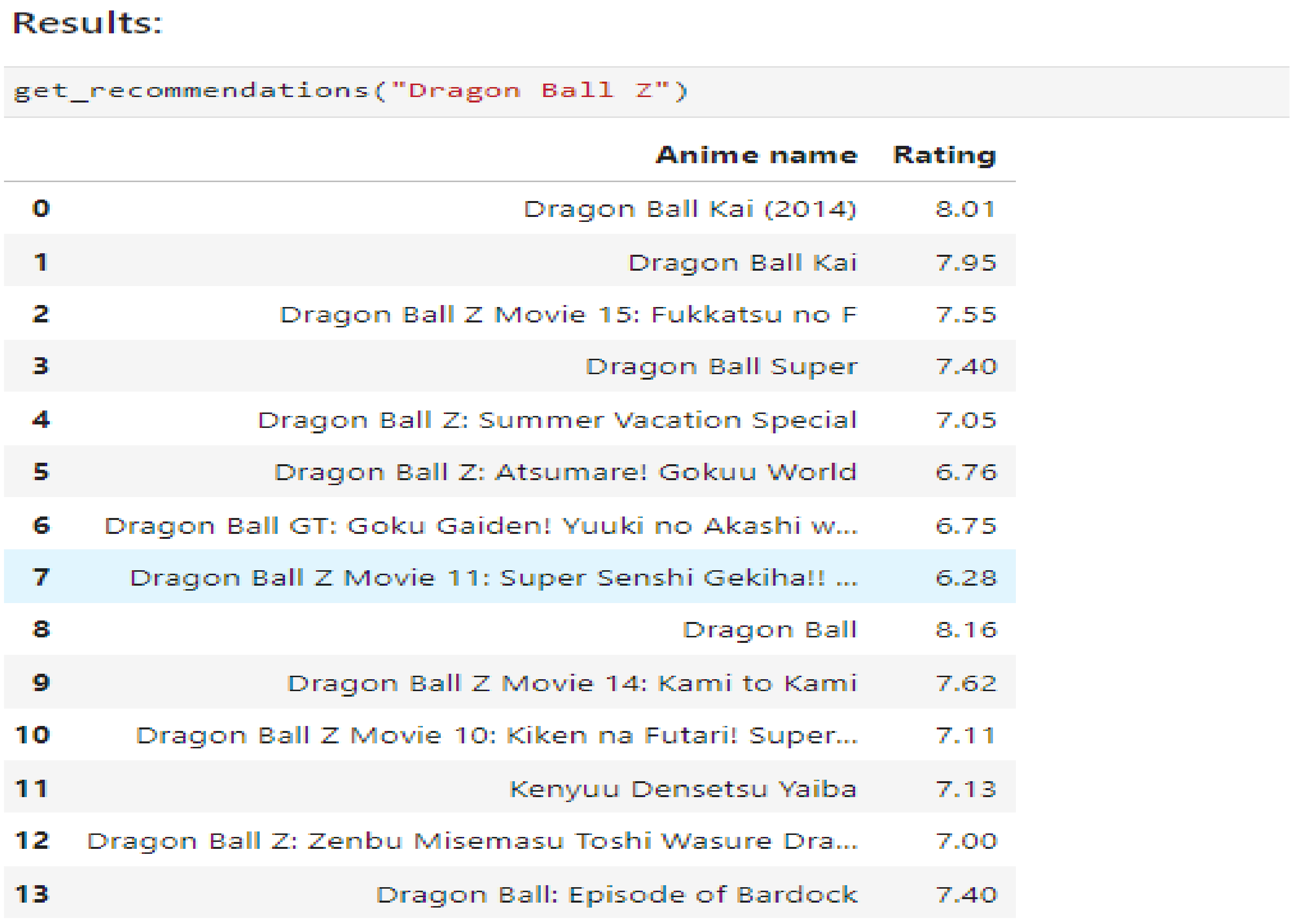
```
from sklearn.neighbors import NearestNeighbors
model_knn = NearestNeighbors(metric='cosine', algorithm='brute')
model_knn.fit(anime_matrix)
```

Recommendation Function

A function is created to provide recommendations based on the similarity scores.

```
def give_rec(title, sig=linear):
idx = indices[title]
sig_scores = list(enumerate(sig[idx]))
sig_scores = sorted(sig_scores, key=lambda x: x[1], reverse=True)
sig_scores = sig_scores[1:15]
anime_indices = [i[0] for i in sig_scores]
return pd.DataFrame({'Anime name': anime_data['name'].iloc[anime_indices].values,
'Rating': anime_data['rating'].iloc[anime_indices].values})
```

Results:



Results:

```
get_recommendations("Dragon Ball Z")
```

	Anime name	Rating
0	Dragon Ball Kai (2014)	8.01
1	Dragon Ball Kai	7.95
2	Dragon Ball Z Movie 15: Fukkatsu no F	7.55
3	Dragon Ball Super	7.40
4	Dragon Ball Z: Summer Vacation Special	7.05
5	Dragon Ball Z: Atsumare! Gokuu World	6.76
6	Dragon Ball GT: Goku Gaiden! Yuuki no Akashi w...	6.75
7	Dragon Ball Z Movie 11: Super Senshi Gekiha!! ...	6.28
8	Dragon Ball	8.16
9	Dragon Ball Z Movie 14: Kami to Kami	7.62
10	Dragon Ball Z Movie 10: Kiken na Futari! Super...	7.11
11	Kenyuu Densetsu Yaiba	7.13
12	Dragon Ball Z: Zenbu Misemasu Toshi Wasure Dra...	7.00
13	Dragon Ball: Episode of Bardock	7.40

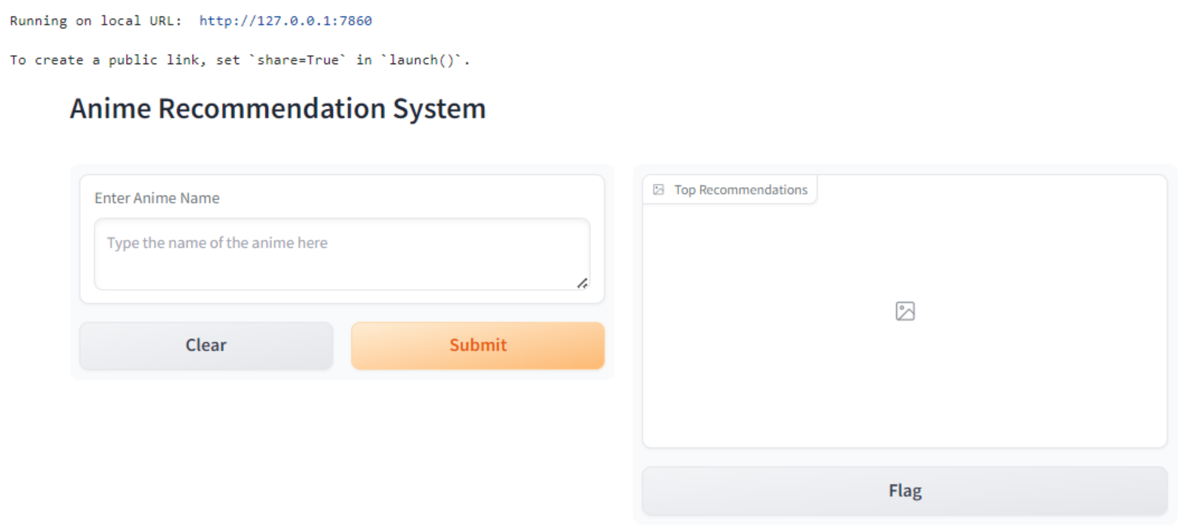
The recommendation engine performed admirably, swiftly generating a list of anime titles closely related to "Dragon Ball Z" based on their respective ratings. The engine's functionality was seamless, swiftly processing the input and retrieving a diverse range of recommendations reflective of the user's interest in the original series. Notably, the engine showcased a keen ability to identify both direct sequels and spin-offs, as well as related titles within the broader Dragon Ball universe. Each recommendation was accompanied by its corresponding rating, providing users with valuable insights into the perceived quality of the suggested titles.

Web Application with Gradio

A web application is built using Gradio to allow users to input an anime name and get recommendations displayed interactively.

import gradio as gr
def get_recommendations(anime_name):
recommendations_df = give_rec(anime_name)
return recommendations_df
def plot_recommendations(recommendations_df):
plt.figure(figsize=(12, 8))
sns.barplot(x="Anime name", y="Rating", data=recommendations_df, palette="viridis")
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.yticks(fontsize=12)
plt.xlabel('Anime', fontsize=14)
plt.ylabel('Rating', fontsize=14)
plt.title('Top Recommendations', fontsize=16)
plt.tight_layout()
with tempfile.NamedTemporaryFile(suffix=".png", delete=False) as temp_file:
plt.savefig(temp_file.name)
return temp_file.name
anime_name_input = gr.inputs.Textbox(label="Enter Anime Name", lines=2, placeholder="Type the name of the anime here")
recommendation_output = gr.outputs.Image(type="file", label="Top Recommendations")
def recommend_anime(anime_name):
recommendations_df = get_recommendations(anime_name)

image_path = plot_recommendations(recommendations_df)
return image_path
interface = gr.Interface(fn=recommend_anime, inputs=anime_name_input, outputs=recommendation_output, title="Anime Recommendation System")
interface.launch()



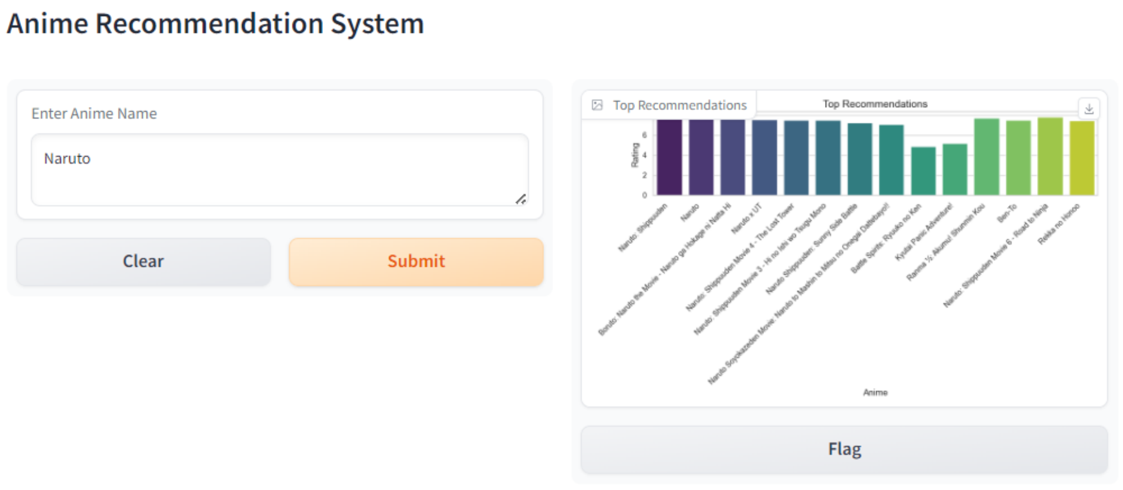
```
Running on local URL:  http://127.0.0.1:7860

To create a public link, set `share=True` in `launch()`.
```

Anime Recommendation System

Enter Anime Name

Type the name of the anime here

Getting the results from the created dashboard interface for the anime Naruto



Anime Recommendation System

Enter Anime Name

Naruto

Clear
Submit
Top Recommendations
Top Recommendations
Rating
0
2
4
6
Naruto: Shippuuden
Boruto: Naruto the Movie - Naruto ga Hokage ni Natta Hi
Naruto
Naruto x UT
Naruto: Shippuuden Movie 4 - The Lost Tower
Naruto: Shippuuden Movie 3 - Hi no Ishi wo Tsugu Mono
Naruto Shippuuden: Sunny Side Battle
Naruto Soyokazeden Movie: Naruto to Mashin to Mitsu no Onegai Dattebayo!!
Battle Spirits: Ryuuko no Ken
Kyutai Panic Adventure!
Ranma ½: Akumu! Shunmin Kou
Ben-To
Naruto: Shippuuden Movie 6 - Road to Ninja
Rekka no Honoo
Anime
Flag

Conclusion

The FindYourAnime recommendation system provides personalized anime suggestions by leveraging collaborative and content-based filtering techniques. The integration with a web application using Gradio allows for an interactive user experience, making it easier for users to discover new anime that matches their preferences.

This project demonstrates the effectiveness of combining different recommendation techniques and the ease of deploying such systems using modern web frameworks. Future improvements could include incorporating more sophisticated natural language processing techniques and expanding the dataset to include more diverse user preferences.