Final Report

1. Title, Team Name, Members

Title: Content-based Recommender System for Music and Podcasts: Utilizing Spotify, Penguin, and NYTimes APIs and NLP Techniques

Team Name: Group 13

Members:

- Camilo Chaves Beltran
- Chuyi Huang
- Xizhi Ma
- Nishchal Mishra
- Peifong Sun
- Yusong Wang

2. Problem/Motivation Statement

In the era of information overload, users often face difficulty in finding relevant content. In such a scenario, recommender systems play a crucial role in providing personalized recommendations. With the increasing availability of user data, the need for advanced recommender systems has become more important than ever. The aim of this project is to build a content-based recommender system for music and podcasts that utilizes the NYTimes, Penguin, and Spotify APIs to collect user engagement data and provide personalized recommendations.

The project utilizes the power of natural language processing (NLP) and machine learning techniques to analyze the textual content of NYTimes articles, Spotify songs, and Penguin podcasts. By applying vectorization to the textual data, similarities between articles and songs/podcasts of similar flavor are calculated. This approach enables the system to make personalized recommendations based on user engagement history and the similarity of their interests.

The project's significance lies in its ability to enhance the content discovery experience for users. Users can easily discover new music and podcasts by providing personalized recommendations based on their interests. Moreover, this project also offers insights into the user engagement behavior of the content providers. This information can be used to create better content, understand user preferences and improve user experience. Overall, the content-based recommender system built in this project is expected to provide an efficient and effective solution for personalized content recommendation and help users navigate the vast amount of content available to them.

3. Your dataset and analytic goals

The Data:

1. **New York Times:**
   New York Times data was extracted from the nytimes api, three features were selected to be included in the model:
   - 'title' is the string type input that stores the title of the article.
   - 'caption' is the string type input that stores the subheading.
   - 'abstract' is the string type input that stores the article summary.

2. **Spotify:**

   Spotify data was extracted from Spotify's api: https://developer.spotify.com/documentation/web-api/, and 16 features were selected to be included in the model:

   The fields includes:

   `artist, album, track_name, track_id, danceability, energy, key, loudness, mode, speechiness, instrumentalness, liveness, valence, tempo, duration_ms, time_signature`

3. **Penguin:**

The data was from https://developer.penguinrandomhouse.com/io-docs. We used requests package to get json string then read and process the data.

The fields includes:

title, author, onsale (which is the date which it gets on sale), language, praises, authorBio, aboutTheBook, keynote, categories

```
▾ 2:
    title: "The Sensuous Woman"
    author: ""J""
    onsale: "1982-10-01"
    language: "E"
  ▾ praises:
      9780440178590: "&quot;[She] is unmistakably female: no man could  possible be so exqui
        sitely knowing about how a woman can  makes the most of what she's got.&quot; --&#16
        0; <i>Playboy</i>"
    authorBio: "<b>J </b>is the pseudonym of<b> Terry Joan Garrity</b>, author of the runaway bestsel
      ler <i>The Sensuous Woman</i>. She lives in Palm Beach, Florida."
    aboutTheBook: "<b>The book that fired the first shot in the sexual revolution</b><br><br>Not too
      long ago only &ldquo;bad&rdquo; girls had a good time in bed. &ldquo;Good&rdquo; girls endured&
      mdash;and wondered what they were missing.<br><br>Then along came &ldquo;J&rdquo; and suddenly&
      #160; everything was different. She opened the eyes&mdash;and minds&mdash;of millions of A
      merican women with her explicit step-by-step account of her pursuit of the  ultimate
      in sexual pleasure.<br><br>Now here&rsquo;s the book  that has set off fireworks in b
      edrooms across America, the book that will teach every woman how to  free her body, t
      rain her senses, and tap her own hidden sensual resources.<br><br>The book designed  
      to make you the woman every man yearns to make love to&mdash;the woman you yearn to be."
    keynote: null
  ▾ categories: [] 10 items
    ▸ 0:
    ▸ 1:
    ▸ 2:
    ▸ 3:
    ▸ 4:
    ▸ 5:
    ▸ 6:
    ▸ 7:
    ▸ 8:
    ▸ 9:
```


The Analytical Goals:

The primary analytical goal of this project is to develop a content-based recommender system for music that utilizes NYTimes, Penguin, and Spotify APIs to collect user engagement data and provide personalized recommendations.

The system's analytical goals are to:

1. Collect user engagement data: The system collects user engagement data from NYTimes, Penguin, and Spotify APIs to understand user preferences and behavior. This data includes the articles read, songs played, and podcasts listened to by the user.

2. Analyze textual content: The system uses NLP techniques to analyze the textual content of the articles, songs, and podcasts to extract relevant features and create vectors.
3. Calculate similarities: The system calculates the similarity between the vectors of the articles and the songs/podcasts to identify articles and songs/podcasts of similar flavor.
4. Provide personalized recommendations: The system provides personalized recommendations based on the user's engagement history and the similarity of their interests.

The analytical goals of this project are aligned with the objective of building a content-based recommender system that provides personalized recommendations for music and podcasts. By achieving these goals, the system can provide relevant content to users, enhance the content discovery experience, and offer insights into user engagement behavior for content providers.

4. The Data Pipeline:

Our goal is to build an Automated Scalable ML Pipeline.The data from three APIs are downloaded into our machine as json objects. These json objects are uploaded to our google cloud clusters right after. After this, the data is downloaded as raw data from gcs, then we process the data in our local machine into a list of jsons. After this, the aggregates are uploaded into mongodb where each json file in the list as a document. Next, the data is downloaded from mongodb to databricks and converted into dataframe to build our model.

5.Preprocessing goals, algorithms, and time efficiency (seconds to run) - with the cluster specification, including Databricks Runtime Version, Worker/Driver types, and the number of workers:

Pre-processing:
Our pre-processing includes imputing nulls, merge data frames, vectorize, tokenization, scaling, normalization, transforming the dataframe, create label columns. There are 4 in total: Song classification, Book to song and song to book recommendation model, and news to news recommendation model. Among them, logistic regression model was used for song binary classification. Book to song/Song to book model is based on "TFIDF" model and cosine similarity. News to news recommendation is based on the "word-to-vector model".

Time efficiency:
logistic regression: Training took 47.60 seconds for 20000 rows * 16 columns records. Testing took 7.15 seconds for 110000 records.
TFIDF: Tokenization and TFIDF embedding took 3.3 seconds. Calculating cosine similarity took 1.46 seconds.

Databricks Runtime Version: 7.3 LTS (includes Apache Spark 3.0.1, Scala 2.12)
Worker/Driver types: i3.xlarge
The number of workers: 2 - 5

6. ML Goals, Outcomes, and Execution Time Efficiency:

The ML Goals:

The primary goal of the recommender system built by the team using SparkML and data from three different APIs - Spotify, Penguin, and NYTimes - is to provide accurate and personalized content recommendations to users. The team explored various possibilities to leverage the data available from the three APIs to build a robust and efficient recommendation engine. The focus was on content-based recommendations, which are based on user preferences and behavior rather than collaborative filtering. By analyzing user interactions with the data from these APIs, the team aimed to build a model that could predict user preferences and provide personalized recommendations for music, books, and news articles. The ultimate goal was to improve the user

experience by delivering relevant and engaging content, thereby increasing user satisfaction and engagement.

1. Song Apt for Reading Books/ Articles - Classification

   We implemented a Logistic Regression model to determine whether a song is suitable for listening when reading. This is preliminary research for 2 and 3 ML Models. Our training and validation data is composed of a "positive label" dataframe (songs good for reading) and a "negative label" dataframe (songs not good for reading). The positive dataframe is a compilation of playlists from Spotify that have been curated by Spotify or Spotify users specifically for reading and have received hundreds of thousands of likes by Spotify's users. Some examples of these playlists are 'The Ultimate Reading Playlist', 'Reading Chill Out', and 'Quiet Music for Reading'. The 'negative labels' dataframe is created in a similar manner. Some examples of the negative label playlists are 'Unlistenable: The World's Worst Playlist', and 'Worst Songs Ever Heard'. Each record of our dataframe has features such as danceability, energy, key, loudness, mode, speechiness, instrumentalness, liveness, valence and tempo.

2. Song to Books Recommendation

   Our main objective is to build a recommendation system that provides personalized book recommendations based on a user's current song listening situation. To achieve this, we use a machine learning model trained on the TF-IDF algorithm to analyze the text of books and identify features that are relevant to the user's listening preferences. We then use cosine similarity to calculate the similarity between the user's input and each book in our database, and provide a list of top 10 recommendations based on this similarity score.

   We collected a large dataset of books and their associated metadata, including information on book title, author, authorBio,book categories, and praises. We used the TF-IDF algorithm to extract key features from the text of each book and trained a machine learning model on this data. To provide personalized recommendations based on a user's current song listening situation, we used cosine similarity to calculate the similarity between the user's input and each

book in our database. Based on this analysis, we generated a list of top 10 book recommendations for the user.

3. Books to Song Recommendation

   The goal of this project is to recommend top 10 songs based on user input with a book title. To achieve this, we apply the TF-IDF model to train the song lyrics and calculate the similarity between the user input and the song lyrics using cosine similarity.

   We follow a few steps like below.

1. We begin by defining the input text, which is the book title provided by the user.
2. We then use the TF-IDF model to train the song lyrics.
3. Using the cosine similarity method, we calculate the similarity between the user input and the song lyrics.
4. The output of the model is a list of top 10 songs that are most similar to the book title provided by the user.

4. News to News Recommendation

To perform unsupervised clustering on New York Times data by converting the text data into vectors.

Outcomes:

1. Song Apt for Reading Books/ Articles - Classification
   We got a ROC = 0.99871 after evaluating the model in the validation set, and predict the labels of “positive” (1) or “negative” (0) using testset.

2. Song to Book Recommendation

   User input:"**I am listening to a love song**"

   Outputs:

```
+------------------------------+------------------+
|title                         |cosine_similarity |
+------------------------------+------------------+
|Poems That Touch the Heart    |0.7117336561384187|
|Being a Green Mother          |0.6897578121430935|
|What a Song Can Do            |0.6620589650763575|
|The "I Love Lucy" Book        |0.6590175734881594|
|Castle Roogna                 |0.6400154036549401|
|J.M.W. Turner                 |0.6207636169425094|
|In the Belly of the Beast     |0.6186008639596937|
|Phenomenal Woman              |0.6175659527893639|
|The Truelove Bride            |0.6127886988806708|
|Diamond Heart: The Freedom to Be|0.6112445940978041|
+------------------------------+------------------+
```

   Based on our analysis, we found that our recommendation system was able to accurately identify relevant books based on a user's current song listening situation. For example, when the user input "I am listening to a love song", our system recommended books such as "Poems That Touch the Heart" and "Being a Green Mother", which have a high degree of similarity to the user's input based on our model.

3. Book to Songs Recommendation

As an example, we used the book title "The Truelove Bride" as the user input and obtained the following top 10 song recommendations:

   User input:"**The Truelove Bride**"

   Outputs:

```
+--------------------+------------------+
|           Song_name| cosine_similarity|
+--------------------+------------------+
|  There Will Be Rain|0.7247226152427609|
|        Authenticity|0.6474468691304123|
|Portrait of a Lon...| 0.629552671299129|
|            Epiphany|0.6267799756484367|
|                 Ala|0.6152825878746678|
|           Equipoise|0.6052418908089235|
|       Woolgathering|0.6046448663520931|
|Clair de Lune, L. 32| 0.590186036367559|
|     Summer Memories|0.5873170606848128|
|  Behind Frozen Eyes|0.5796934731864298|
+--------------------+------------------+
```

4. News to News Recommendation
   K-Means clustering algorithm was applied on the data to bucket the articles into 10 clusters, these clusters were generated from the content of the articles and future articles based on user engagement can be passed through the model to be clustered in a bucket, similar articles can then be suggested based on similar content.
   The model takes around 15-20 seconds to train on a feature.

7. Lesson Learned:

1. Data pipelines through Airflow often start small, but as data volumes grow, the pipeline needs to be able to handle the increased workload, so it's important to design a pipeline that can scale horizontally and vertically, and to anticipate future growth.

2. During Machine Learning, data quality can be a significant challenge, and It's important to establish data quality standards, validate data at each stage of the pipeline, and have processes in place to handle data quality issues.

3. Testing the pipeline is critical to ensuring its reliability and accuracy. It's important to test the pipeline at every stage, from data ingestion to transformation and loading, and to have automated tests in place to catch errors quickly.

8. Conclusion:

We were set off to do a book to song content-based recommendation system. The part we finished in the group project was a song filtering classification model for songs that are good for reading. What's more, we also made a TF-IDF similarity model with song to book and book to song recommendation based on book information like "about the book", "author information", etc., and song lyrics. The news recommendation recommends news to only news. Future development would include evaluating the recommendation carefully to see the performance. Based on the evaluation of performance the model can be improved. Meanwhile, to give a complete pipeline, Songs suitable for reading will be classified correctly with a high probability according to our AUC score on the validation set. Based on this filtered output, other models will use our logistic regression model to filter their song data to then map a song to a list of articles or books.