CPSC 531

Advanced Database Management

DEVELOPMENT OF BIG DATA ANALYTICS APPLICATIONS (E-COMMERCE)

Neelapareddy Niharika - 885205831

Atmakuru Vignasree - 885638759

PROBLEM STATEMENT

Background

E-commerce websites have become popular in recent years as many people have embraced online shopping, and they have a large catalog of products across multiple categories. While having a large catalog may be a good thing, customers can be overwhelmed by the sheer number of products available and may find it difficult to find products relevant to their interests. This can result in users leaving the site without making a purchase.

Problem

Customers are experiencing poor shopping experiences because the e-commerce site does not have a personalized product recommendation system leading to low conversion rates.

Objective

To develop a product recommendation system that would recommend products to customers based on their previous purchases and also that of other customers with similar interests. The system should employ machine learning techniques to come up with user-specific recommendations and will be integrated into the website to provide real-time recommendations to customers when they are on the site. Customers shopping experience should increase significantly once the system is integrated. Subsequently, the conversion rates on the site should improve as purchases will increase.

Approach

Developing a product recommender system can be done with the following approach:

1. Collect and pre-process the rating data and product information. This may involve cleaning the data, removing missing or invalid entries, and transforming the data into a suitable format for the recommendation algorithm.
2. Create a sparse matrix of rating data, where each row represents a product, and each column represents a customer. The values in the matrix represent

customers' ratings for the products. Techniques such as collaborative filtering can be used to create the rating matrix.
3. Use a k nearest neighbors (kNN) algorithm to find similar products in the rating matrix. This can be done by fitting a nearest neighbors model to the rating matrix and using the model to find the products that are most similar to a given product.
4. Integrate the recommendation system on the website and use it to provide recommendations to users based on similar products generated by the algorithm.
5. Evaluate the performance of the recommendation system by measuring how well it can predict the products that a customer will be interested in. This can be done using metrics such as precision and recall.
6. Fine-tune the recommender system by adjusting the parameters of the nearest neighbors algorithm and experimenting with different approaches to preprocessing the data and generating recommendations. This can involve comparing the performance of different algorithms and parameter settings and selecting the best-performing configuration.

Design and architecture

The architecture of this recommendation system using a CSR matrix and a KNN algorithm will consist of multiple components working together to collect, process, and serve recommendations to customers. This approach uses a distributed architecture with multiple components working together to collect, process, and serve recommendations. The components in the recommendation system will include:

1. Data collection: This component would be responsible for collecting data on customer behavior, such as which products they have purchased and the ratings they gave said products. This data would be stored in a database or data file.

2. Data processing: This component would be responsible for converting the raw data into a CSR matrix and training the KNN algorithm using this data.

3. Recommendation generation: This component would use the trained KNN algorithm to generate recommendations for a given customer. It would query the CSR matrix to find the nearest neighbors and generate the recommendations.

4. Recommendation serving: This component would be responsible for serving the generated recommendations to the customer, either on the e-commerce website or app.

5. Evaluation: This component would monitor the performance of the recommendations and provide feedback to the data processing and recommendation generation components to improve the accuracy of the recommendations over time.

Each component will be implemented as a separate software module that communicates with the other component's modules over a network.

Tools and Technologies

To develop this product recommendation system, the following tools and technologies are used:

- Data preprocessing and manipulation tools, such as Pandas and NumPy, for cleaning and transforming the rating data and product information.
- Machine learning libraries, specifically scikit-learn, for implementing and training the recommendation algorithm.
- Nearest neighbors algorithms (k-nearest neighbors) for finding similar products in the rating data.
- Sparse matrices, such as the Compressed Sparse Row (CSR) matrix, for storing and manipulating the rating data in a memory-efficient way.
- Metrics and evaluation methods, such as precision and recall, for measuring the accuracy and effectiveness of the recommendations made by the system.
- Web development technologies such as HTML, CSS, and JavaScript or a server-side language such as Python or Java can be used to integrate the system with the e-commerce website or app.
- Python (Version 3.10) is used as the preferred programming language for developing this system.

Product Recommendation Algorithm

Prerequisites

To use the algorithm, you need to have the python programming language installed on your PC (python 3.10 is recommended).

The following packages are used by the algorithm and can be installed by pip or conda:

- NumPy
- pandas
- scikit-learn
- scipy

The packages can be installed using pip with the following command

- *`pip install numpy pandas scikit-learn scipy`*

The following files are also needed to run the algorithm:

- Products file - a CSV file containing the catalog of all products sold on the site
- Customer data file - a CSV file is containing products and corresponding ratings given by customers. It should have the following columns: customer_id, product_id, rating

Running the application

Follow these steps to run the system:

- Instantiate the `*ProductRecommender*` class passing in paths to customer data and products CSV files as arguments
  `rec = ProductRecommender("./customer_data.csv", "./products.csv")`

- Get all products rated by a user, and for each product, find `*k*` most similar products and add them to the set of products to be recommended to the user
  `
  ```
  user_id = 1;
  recommended_products_ids = set()
  for productId in rec.get_products_rated_by_user(user_id):
  ```

recommended_products_ids.update(set(rec.find_similar_products(productId,k=4)))
`

- The set of products contains unique identifiers(Uniq Id) for the products to be recommended to the user. After integration, the products can then be recommended to the user on the e-commerce website.

Testing and Evaluation

Introduction

Testing and evaluation are important steps in the development of a recommendation system. There are several metrics that can be calculated to monitor and improve the system over time as more data becomes available. These metrics help ensure that the system is accurate and reliable, and can provide valuable feedback for improving the system. The metrics to be used in evaluation include:

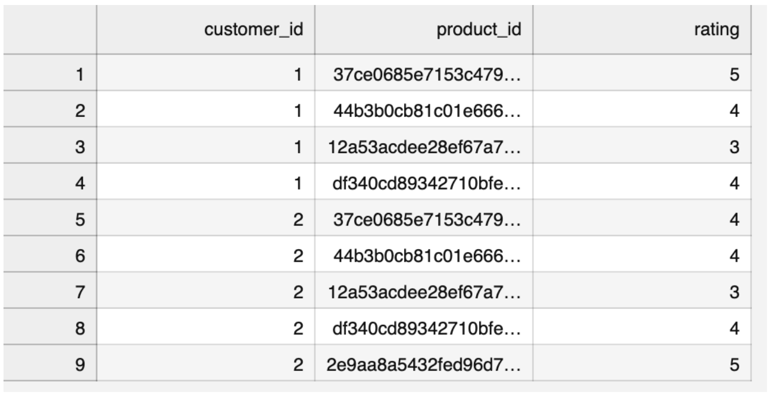
1. Precision: This metric measures the proportion of recommended items that are actually relevant to the user. A high precision indicates that the system is able to accurately identify relevant items and exclude irrelevant ones.

2. Recall: This metric measures the proportion of relevant items that are actually recommended by the system. A high recall indicates that the system is able to identify all of the relevant items, even if it also recommends some irrelevant ones.

3. F1 score: This metric is the harmonic mean of precision and recall, and is commonly used to combine the two metrics into a single score. A high F1 score indicates that the system has a good balance of precision and recall.

Test Results

Given a Customer ratings CSV file with 9 records as shown below:



	customer_id	product_id	rating
1	1	37ce0685e7153c479...	5
2	1	44b3b0cb81c01e666...	4
3	1	12a53acdee28ef67a7...	3
4	1	df340cd89342710bfe...	4
5	2	37ce0685e7153c479...	4
6	2	44b3b0cb81c01e666...	4
7	2	12a53acdee28ef67a7...	3
8	2	df340cd89342710bfe...	4
9	2	2e9aa8a5432fed96d7...	5

The customer with id 1 has rated a total of 4 products with an average rating of 4. The customer with id 2 has rated 5 products with an average rating of 4. Four out of five products rated by the customer with id 2 have also been rated by the customer with id 1.

The system is tested as follows for the customer with id 1 and the dataset above:

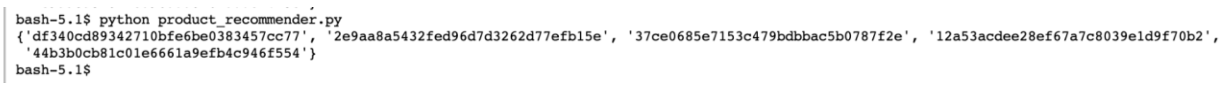
```
# create instance of ProductRecommender class, passing in customer data and
products csv files as arguments
rec = ProductRecommender("./customer_data.csv",
"./marketing_sample_for_walmart_com-
ecommerce__20191201_20191231__30k_data.csv")

user_id = 1;
recommended_products_ids = set()

# Find all products rated by user with ID 1
for productId in rec.get_products_rated_by_user(user_id):
  # For each product, finds the k (nearest neighbours) most similar products and
add them to the set
  recommended_products_ids.update(set(rec.find_similar_products(productId,
k=4)))
```

```
print(recommended_products_ids)
```

The program is then run by running the command in the terminal:

- *`python product_recommender.py`*

The screenshot below shows the output of running the command above:



```
bash-5.1$ python product_recommender.py
{'df340cd89342710bfe6be0383457cc77', '2e9aa8a5432fed96d7d3262d77efb15e', '37ce0685e7153c479bdbbac5b0787f2e', '12a53acdee28ef67a7c8039e1d9f70b2',
 '44b3b0cb81c01e6661a9efb4c946f554'}
bash-5.1$
```

The set of products is as follows:
{'df340cd89342710bfe6be0383457cc77', '2e9aa8a5432fed96d7d3262d77efb15e', '37ce0685e7153c479bdbbac5b0787f2e', '12a53acdee28ef67a7c8039e1d9f70b2', '44b3b0cb81c01e6661a9efb4c946f554' }

From the results, we can see that the algorithm recommends all the products the customer has been interested in previously and also recommends product id ‘2e9aa8a5432fed96d7d3262d77efb15e’ which was rated by Customer 2, who has similar interests to Customer 1.

Evaluation

1. Precision
   To calculate the precision, we need to determine the number of recommended items that are actually relevant to the user and divide that by the total number of items recommended. In the above example, if the system recommends 5 items to user 1. To determine the relevance of a recommended item to the user we can use the ratings data and consider products with a rating of 4 and higher as being highly relevant to the user. In this case, 4 of the recommended items will be considered highly relevant. Precision can then be calculated as follows:

Precision = 4 relevant items / 5 total items = 0.8

2. Recall

   To calculate the recall, we need to determine the number of relevant items that are actually recommended by the system and divide that by the total number of relevant items. In the above example, there are 5 relevant items for user 1 and the system recommends 4 of those items. Recall can be calculated as follows:

   Recall = 4 recommended relevant items / 5 total relevant items = 0.8

3. F1 Score

   The F1 score is a metric that combines precision and recall into a single score. It is calculated as the harmonic mean of precision and recall, and is defined as follows:

   F1 = 2 * (precision * recall) / (precision + recall) = 2 * (0.8*0.8) / (0.8+0.8) = 0.8

Conclusion

The recommendation system can now be rolled out to the e-commerce website or app, where its performance can be continually monitored and optimized. It can be fine-tuned by adjusting the parameters and retraining the algorithm as more data is available. This can be done iteratively to improve the system’s performance over time.