Recommendation system based on semantic analysis for Amazon Video

Objectives

Amazon Marketplace has over 310 million active users worldwide. One of Amazon's main products is its on-demand service, Amazon Prime Video.As of July 2022, Amazon Prime Video has gained immense popularity and is ranked third among the most downloaded video streaming mobile apps in the world, right behind industry giants Netflix and Disney+.

It is important for Amazon to understand its customer base and their preferences. Amazon aims to gain valuable insights from customer reviews. This allows Amazon to customize its content offerings and push out videos that meet customer needs.

This project aims to provide Amazon with a technical solution to better understand customer opinions based on their reviews. By implementing a text categorization model and semantically analyzing reviews, it can gain insight into customer preferences. This will allow them to push more videos that match customer preferences, ensuring a personalized and tailored user experience.

Technology

The Whole Project can be devidied into three parts, Data processing, Model deployment, Recommendation.

Data Processing

In data Processsing, I make use of the NLTK (Natural Language Toolkit) and re (regular expression) packages in Python.

"Pasted image 20231113222830.png" is not created yet. Click to create.

Regular expressions can be used to filter out HTML tags and UTF-8 encoding symbols from data crawled from the internet.

1. **Filtering HTML Tags**: When dealing with data crawled from the internet, it often contains HTML tags. To remove these tags and extract meaningful text, regular

expressions can be used. The re package provides functions to match and replace patterns in text strings. By using appropriate regular expressions, you can eliminate HTML tags and obtain clean text data.

2. Handling Encoding Symbols: Data crawled from the internet might have encoding symbols, especially if it's in UTF-8 format. To handle such symbols, you can utilize Python's built-in support for different encodings. The re package can also be helpful in identifying and replacing specific encoding symbols with the desired representation.

NLTK (Natural Language Toolkit) can be utilized for performing tokenization, lowercasing, and lemmatization tasks on text data.

1. Tokenization: NLTK provides various tokenization methods to split text into individual tokens or words. Tokenization is an essential step in natural language processing as it allows you to analyze text on a granular level. NLTK offers functions for word tokenization, sentence tokenization, and more.
2. Lowercasing: In many text processing tasks, converting all text to lowercase is beneficial. It helps in standardizing the text and avoids the distinction between words based on case. NLTK offers functions to convert text to lowercase, making it easier to compare and analyze.
3. Lemmatization: Lemmatization is the process of reducing words to their base or dictionary form, known as lemmas. NLTK provides lemmatization algorithms that can handle different parts of speech and transform words accordingly. Lemmatization helps in reducing inflectional forms and consolidating related words.

Model deployment

The main focus of this process is to apply the selected models, such as BERT-Sentimental, BERT-Base-Multilingual-Uncased-Sentiment, or the sentiment analysis for Amazon reviews pipeline, to perform semantic analysis on customer reviews. To enhance the efficiency of the analysis, the chosen models are deployed on a local machine.

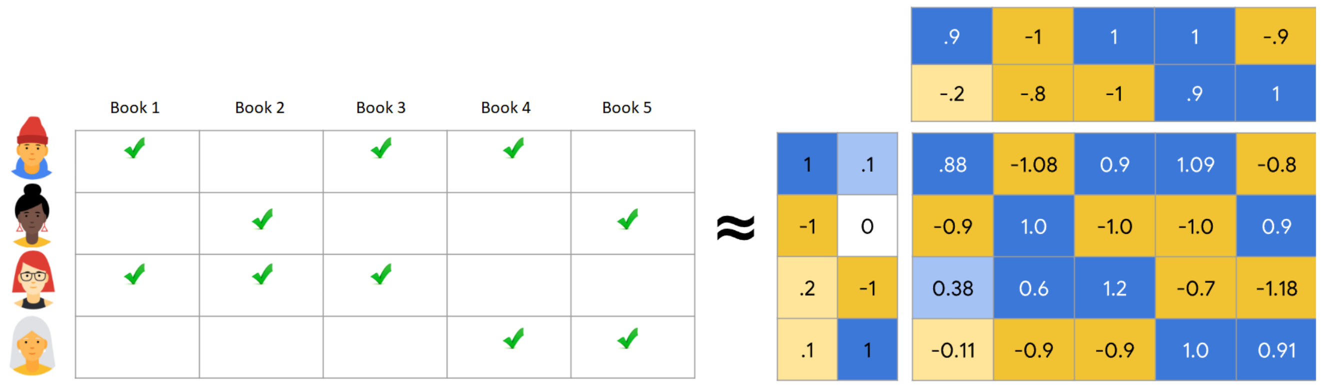
However, due to the slow processing speed when using the CPU, GPU acceleration is using by utilizing CUDA. By using GPU resources, the analysis speed can be significantly boosted, resulting in faster and more efficient sentiment analysis of the customer reviews.

Recommendation System

In my data processing workflow, I incorporate the PySpark package, which is specifically designed for big data processing using Python. The primary objective is to use the ALS (Alternating Least Squares) algorithm for recommendation purposes.

The ALS algorithm revolves around the concept of creating a matrix that captures user scores for movies. When two users exhibit similar scores for the same movies, it indicates that the preferences of one user may be applicable to the other. By establishing these connections, the algorithm logically infers that a good score given by one customer might also be suitable for the paired user.

The ALS algorithm is particularly useful in collaborative filtering scenarios, where recommendations are based on the preferences and behavior of similar users. By analyzing the user-item matrix and applying matrix factorization techniques, ALS enables the generation of accurate and personalized recommendations.



By leveraging the ALS algorithm, I can process large-scale datasets efficiently and provide valuable recommendations based on user preferences and similarity patterns.

Implementation

Data gathering

In the data gathering stage, I downloaded the data from the Recommender Systems and Personalization Datasets study conducted by Julian McAuley at UCSD. For this project, I utilized the "Small" subset of data from the "Movies and TV" category, which contains over 3,400,000 data points. However, due to limitations with my computer's processing power, I can only handle around 2000 data points at a time. Therefore, in this project, I will perform sampling and randomly select 2000 data points from the entire dataset.

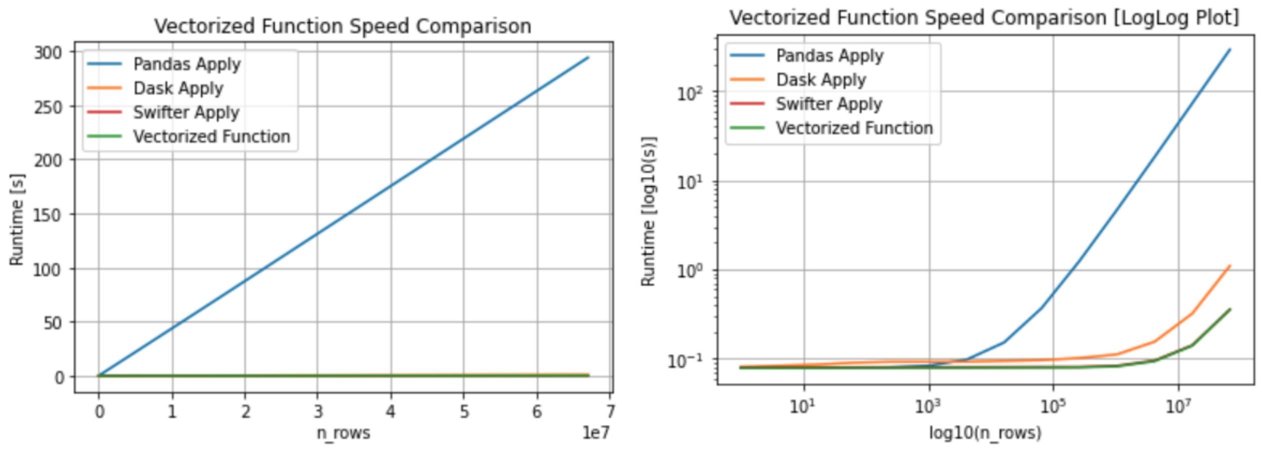
Data Processing

In the Data Processing stage, the first step is to filter out comments that are null or empty, as they are irrelevant to the project's objective of performing semantic analysis based on reviews. These empty comments do not provide any valuable information

```
data = data.dropna(subset=["reviewText"])
```

After filtering out the null comments, the next step is to perform text processing on the reviews using regular expressions and NLTK, as mentioned in the previous section. However, in this project, I will utilize an additional Python package called Swifter to enhance the processing speed.

Swifter is a Python package that optimizes the execution of pandas' apply functions and allows for faster processing of large datasets.



from swifter

Sentiment Analysis stage

During the **Sentiment Analysis stage**, to process the review text, it is advisable to use GPU CUDA for faster execution. Performing the task locally using only the CPU may result in significantly slower processing times. By using GPU CUDA acceleration, which involves deploying the job locally, the sentiment analysis task can be enabling more efficient analysis of the review data.

The first and third models align with each other, but the second model does not, as it is difficult to convert positive and negative sentiments into a 1 to 5 scale.

"Pasted image 20231114142019.png" is not created yet. Click to create.

Recommendation system

In the **Recommendation System**, when using PySpark and the ALS algorithm, it is necessary to convert the string columns "reviewerID" and "asin" into numeric indices. The ALS algorithm requires numeric data for its computations. By converting these string columns into numeric indices, the algorithm can effectively process and generate recommendations based on the numerical representations of the reviewer and item IDs. This conversion allows for integration of the ALS algorithm within the recommendation system pipeline in PySpark.

The whole ALS model can be divided into four stages including data preparation phase, Model Training, Model Evaluation and Generation Recommendation.

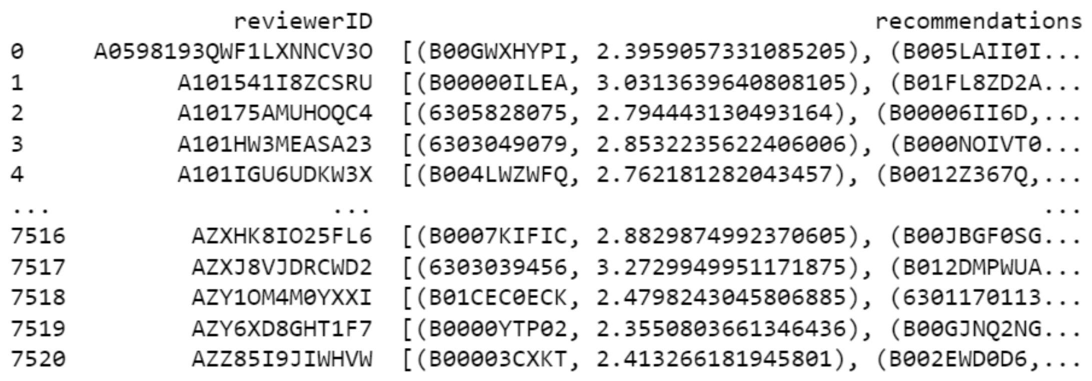
The data preparation phase involves creating a user-item interaction matrix and splitting the data into training and test sets. All index values are converted to numeric values for consistency. The model training stage applies the ALS algorithm to the training data and fine-tunes hyperparameters like MaxIter, regParam, and Rank. Model evaluation is performed on the test set using the RMSE metric. Finally, the trained model is used to generate personalized recommendations by leveraging user preferences and behavior. This comprehensive pipeline encompasses data preparation, model training, evaluation, and recommendation generation to deliver accurate and relevant suggestions to users.

```
# Create StringIndexer transformers for reviewerID and asin columns
reviewerIDIndexer = StringIndexer(inputCol="reviewerID",
outputCol="indexedReviewerID")
asinIndexer = StringIndexer(inputCol="asin", outputCol="indexedAsin")
# Fit StringIndexer on the DataFrame
indexerModel = reviewerIDIndexer.fit(df)
indexedDF = indexerModel.transform(df)
indexerModel = asinIndexer.fit(indexedDF)
indexedDF = indexerModel.transform(indexedDF)
```

By combining the ALS model's predictions with the sentiment-based weights, we can generate recommendations that take into account both user preferences and the sentiment expressed in their reviews.

Another important step is to convert the numeric indices, which were generated by PySpark, back to their human-readable form. After performing the recommendation process using the ALS algorithm and obtaining the numeric indices for reviewers and items, it is necessary to map these indices back to their original string representation.

Example



```
                reviewerID                                          recommendations
0     A0598193QWF1LXNNCV3O  [(B00GWXHYPI, 2.3959057331085205), (B005LAII0I...
1           A101541I8ZCSRU  [(B00000ILEA, 3.0313639640808105), (B01FL8ZD2A...
2           A10175AMUHOQC4  [(6305828075, 2.794443130493164), (B00006II6D,...
3           A101HW3MEASA23  [(6303049079, 2.8532235622406006), (B000NOIVT0...
4           A101IGU6UDKW3X  [(B004LWZWFQ, 2.762181282043457), (B0012Z367Q,...
...                    ...                                                  ...
7516          AZXHK8IO25FL6  [(B0007KIFIC, 2.8829874992370605), (B00JBGF0SG...
7517          AZXJ8VJDRCWD2  [(6303039456, 3.2729949951171875), (B012DMPWUA...
7518          AZY1OM4M0YXXI  [(B01CEC0ECK, 2.4798243045806885), (6301170113...
7519          AZY6XD8GHT1F7  [(B0000YTP02, 2.3550803661346436), (B00GJNQ2NG...
7520          AZZ85I9JIWHVW  [(B00003CXKT, 2.413266181945801), (B002EWD0D6,...
```

given the asin and predication score

Key contributions

key contributions of this project involve using the ALS algorithm and a semantic model to enhance video recommendations for Amazon Prime Video users. By using the ALS algorithm, which considers user-item interactions, personalized recommendations can be generated based on historical review data. Additionally, by incorporating a semantic model for sentiment analysis, customer reviews can be analyzed to gain valuable insights into customer preferences and opinions. This allows Amazon to customize its content offerings and push out videos that align with customer needs and preferences, ensuring a more personalized and tailored user experience on the platform.

Reference

https://medium.com/jackys-blog/pyspark%E7%89%88-%E5%A6%82%E4%BD%95%E5%AE%8C%E6%88%90%E5%BE%9E%E9%A0%AD%E5%88%B0%E5%B0%BE%E5%AE%8C%E6%88%90%E4%B8%80%E5%80%8B%E8%B3%87%E6%96%99%E7%A7%91%E5%AD%B8%E5%B0%88%E6%A1%88-dbf0a6ec9f59

https://medium.com/@patelneha1495/recommendation-system-in-python-using-als-algorithm-and-apache-spark-27aca08eaab3

https://www.youtube.com/watch?v=FgGjc5oabrA&ab_channel=jamenlong1