Amazon Co-Purchasing Meta Data

Washington State University

December 12, 2022

Table of Contents

Abstract

This project uses Amazon product metadata to produce a graphical user interface that generates book recommendations for a user. The focus of this project is to use co-purchasing history to map the relationship between books and evaluate the connections to make accurate product recommendations. Using a dataset of 548,552 different products on Amazon, this paper outlines the methods used to build a successful product recommender and in turn, create a book recommendation graphical user interface.

Introduction

Consumer-based companies are always looking for innovative ways to increase product sales. Predicting the purchase of products based on a user's order history can help companies recommend valuable products to users. A successful prediction that leads to the purchase of a product can greatly help companies increase revenue. According to McKinsey, Amazon's AI product recommendation engine generates 35% of the company's revenue (McKinsey, 2013.). Understanding the significant role that product recommenders play in a consumer-based company, we set out to create an accurate book recommender.

The Data

The dataset our team used is the Amazon product co-purchasing metadata from Standford Network Analysis Project, https://snap.stanford.edu/data/amazon-meta.html. The data was collected by a web crawler over the Amazon products website. The web crawler collected 548,552 products ranging from books, DVDs, music, and VHS videos. The entire dataset had 1,788,725 pairs of products listed as co-purchased products. This also included 7,781,990 different reviews for each product. The average product had 3.3 co-purchased items for each purchase.

The dataset had four major groups, Books, music, CDs, DVDs, and VHS videos. Table 1 has a breakdown for each group:

Table 1

Group	Total Count
Books	393,561
DVD	19,882
Music	103,144
Videos	26,132

The data was received as one tar, zip file. After decompressing and opening the master co-purchasing file it had the following format (See Figure 1), which was parsed using a Python programming language that will be discussed in detail in the next section.

Figure 1. Original data format.

Data format

```
Id:   15
ASIN: 1559362022
  title: Wake Up and Smell the Coffee
  group: Book
  salesrank: 518927
  similar: 5  1559360968  1559361247  1559360828  1559361018  0743214552
  categories: 3
   |Books[283155]|Subjects[1000]|Literature & Fiction[17]|Drama[2159]|United States[2160]
   |Books[283155]|Subjects[1000]|Arts & Photography[1]|Performing Arts[521000]|Theater[2154]
   |Books[283155]|Subjects[1000]|Literature & Fiction[17]|Authors, A-Z[70021]|( B )[70023]|B
  reviews: total: 8  downloaded: 8  avg rating: 4
    2002-5-13  cutomer: A2IGOA66Y6O8TQ  rating: 5  votes:   3  helpful:   2
    2002-6-17  cutomer: A2OIN4AUH84KNE  rating: 5  votes:   2  helpful:   1
    2003-1-2  cutomer: A2HN382JNT1CIU  rating: 1  votes:   6  helpful:   1
    2003-6-7  cutomer: A2FDJ79LDU4O18  rating: 4  votes:   1  helpful:   1
    2003-6-27  cutomer: A39QMV9ZKRJXO5  rating: 4  votes:   1  helpful:   1
    2004-2-17  cutomer:  AUUVMSTQ1TXDI  rating: 1  votes:   2  helpful:   0
    2004-2-24  cutomer: A2C5K0QTLL9UAT  rating: 5  votes:   2  helpful:   2
    2004-10-13  cutomer:  A5XYF0Z3UH4HB  rating: 5  votes:   1  helpful:   1
```

Our parser works by first creating two empty lists: one for the product info and one for the review info. The empty product list is then populated with one empty dictionary for each product ID. Then the parser works through each line in the document and uses the ID as an index to keep track of which product it is currently on. It gathers all data for each index and then moves on to the next. The reviews list was a little more straightforward to populate. Since the info for each review is listed in one line, we were able to split the line and store the data as a review for the product ID at the current index. Once both lists were populated, we converted them to Pandas data frames because this allowed for easy conversion to CSV or JSON files.

Database selection

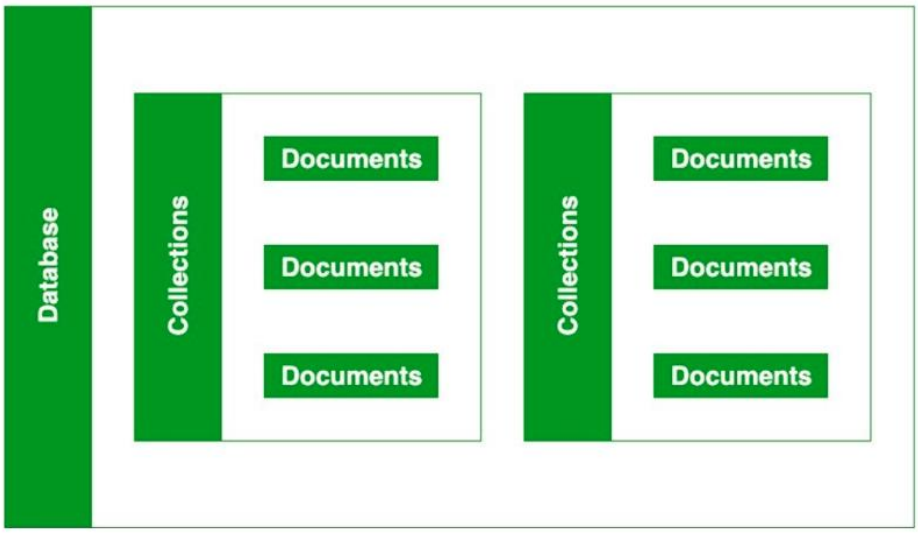
For the For the database type, we reviewed a couple of non-SQL databases, graph, and document. We agreed that the document database was the best solution for the Amazon co-purchasing data set. The document database is a type of nonrelational database designed to store and query data as JSON-like documents which can describe each Amazon purchase item (See Figure 2). The document model works well with use cases such as catalogs, user profiles, and content management systems where each document is unique and evolves, which is best when an Amazon ASIN user purchase additional items, the document can be updated with added content. Document databases enable flexible indexing, powerful queries, and analytics over collections of documents. This made the user interface easy and fast to implement to query the database after changing several input parameters.

Figure 2. An example of a document in the *products* collection.


```
{       "Id":"1",
        "ASIN":"0827229534","title":"Patterns of Preaching: A Sermon Sampler",
        "group":"Book",
        "salesrank":"396585",
        "similar":"0804215715, 156101074X, 0687023955, 0687074231, 082721619X",
        "categories":"Christianity, Clergy, Subjects, Religion & Spirituality, Sermons, Books,
        "reviewTotal":"2",
        "downloaded":"2",
        "avgRating":"5"
}
```


For the document database, we selected Mongo as the provider to store the Amazon data. To further describe what a document database contains, we will describe the structure in Figure 3.

Figure 3



MongoDB:

- A document database that contains collections of documents with the flexibility for indexing and querying. One can create multiple databases on the MongoDB server.
    - Collections function like tables in relational databases which store data in the form of documents. A single database allows you to store multiple collections.
- Data records are stored as a compressed BSON document. BSON is a binary representation of JSON (JavaScript Object Notation) documents. However, BSON contains more data types as compared to JSON. The document is created using field-value pairs or key-value pairs, and the field value can be of any BSON type.

Book recommender

A recommendation engine is a class of machine learning which offers relevant suggestions to the customer. Before recommendation systems, a common way in which we decided what to buy was to take suggestions from friends. But now Google knows what news you will likely read, YouTube knows what type of videos you will watch, and Amazon knows what products you will likely buy. These services know these things based on your search history, watch history, or purchase history.

Although the Amazon co-purchasing dataset included data on books, DVDs, and music, for this recommender we chose to only focus on recommending books. The algorithms implemented

could be applied to the other products in the same manner if so desired. The data set included the following fields:

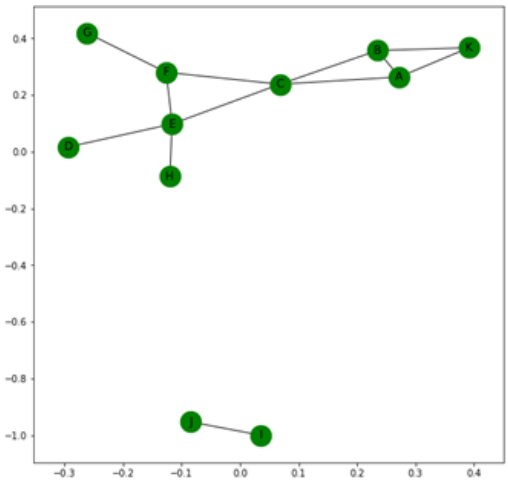
- Id: (Product ID)
- ASIN: (The Amazon Standard Identification Number (ASIN) is a 10-character alphanumeric unique identifier assigned by Amazon.com for product identification)
- Title: Name of the product
- Group: type of product {Book, DVD, Video, Music}
- Sales rank: Sales rank is a number that represents how the product sells in comparison to other products. The lower the rank, the better the product is selling
- Similar: ASIN of other co-purchased products
- Categories: Type of product
- Reviews: Total number of reviews, average rating, number of votes on the review
- Average Rating: Averaging rating for reviews on the product

For the book recommender, we used a Python package called NetworkX to create a node, edges graph. NetworkX is a library built for building, querying, and analyzing complex network graphs. Graphs are data structures to describe relationships and interactions between entities in complex systems and this case of the book recommender, how does one book relationship interact with another book. For the Amazon book recommender graph, the nodes contain the ASIN. Two nodes are connected by an edge if the two were co-purchased together. The edge weight was based on category similarity. The similarity was calculated by the number of common words between the nodes divided by the total number of words in both categories between the nodes. This value is between 0 and 1 where 1 has the most similarities between the two nodes.

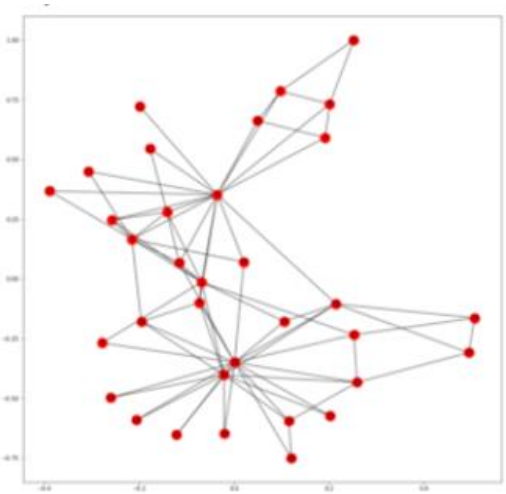
The clustering coefficient was calculated by the example seen in **Graph 1**. A node in a Graph is the fraction of pairs of the node's neighbors that are adjacent to each other. For example, node C of the above graph has four adjacent nodes, A, B, E, and F. Thus, the clustering coefficient calculation is 2/3 = 0.667.

Degree Centrality is based on important nodes that have the most connections **Graph 2**. In the book recommender, a book has many connections to other book nodes. This is calculated by:

Graph 1 (Clustering Coeff)

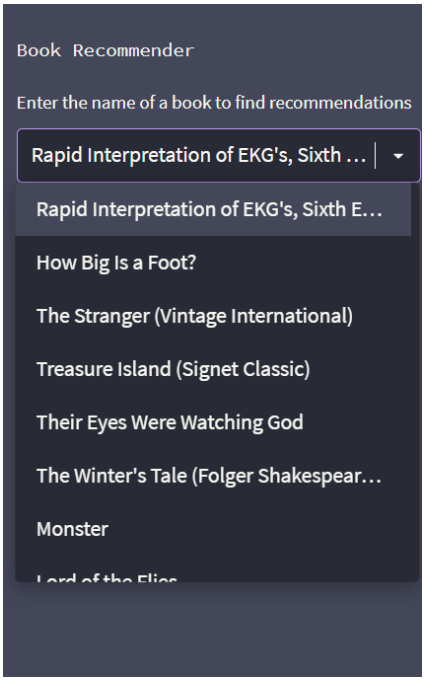


Graph 2 (Centrality)



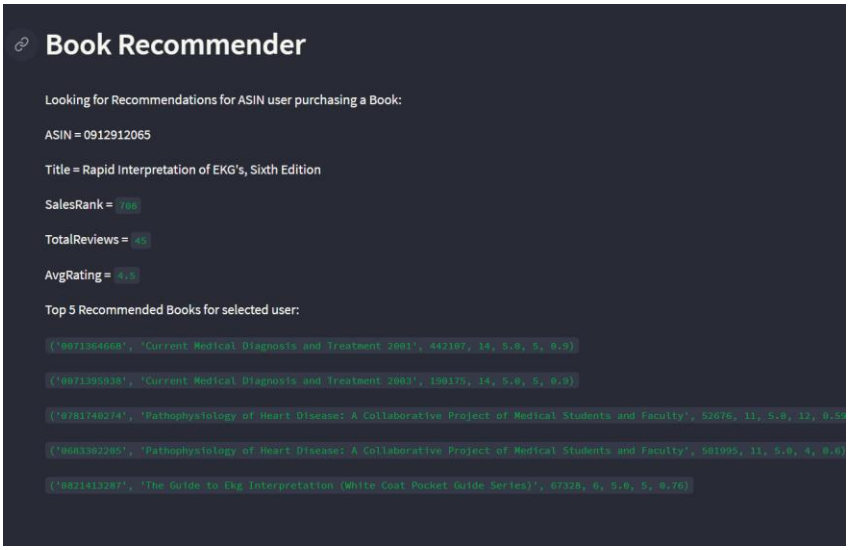
For each book in the Amazon co-purchasing data set, the clustering coefficient and Degree of centrality were calculated. From the creation of the Streamlit web app, a tab was created for a user to select from a pull-down menu a list of books. (See Figure 1)

Figure 1:



From the book selection, the algorithm cross-referenced the ASIN number associated with the selected book. The ASIN number was used to locate the best fit to the clustering coefficient, and degree of centrality. What was discovered was the considerable number of connected edges to nodes for books, so an additional criterion was used to narrow down the selection of recommended books. The criteria selected include the average rating and total reviews for each book. The highest of each of these criteria was used to select the top five recommended books for a user. If a user read a book and wanted an additional reading, they could use this algorithm to find similar books they would enjoy. (See Figure 2)

Figure 2:



For improvement to the algorithm, a little EDA (Exploratory Data Analysis) was completed upfront. The book selection could be improved by removing books that had low sales, reviews, and ratings from previous readers. This is always debatable since those “hidden” gems of books could be forever lost if removed from the initial data set. Instead of using a graph, node, or edge algorithm, the use of K-nearest and clustering based on Euclidian distance could have been incorporated. This comparison would have been interesting to explore but, given the project's time and scope, this can be completed at another time.

Conclusion

The Amazon co-purchasing data set was parsed first in JSON format and then loaded into Mongo database as a document in its collection. Three collections were created, products, reviews and BookRecommend under the database called CPTS415_project. These collections were used in our Python, Streamlit web application to query for statistics of the dataset and

create a book recommender system. The book recommender algorithm was based on a graph, node, edge model using clustering and centrality coefficients, along with averaging rating and reviews to narrow the selection for the best books to fit the user's preferences.

Executive Summary

This project involved the analysis of the Amazon product co-purchasing metadata from the Standford Network Analysis Project. The objective of this project was to use customer co-purchasing history to make accurate product recommendations. Our approach was to use a document database to store and query the dataset and use a node, edge graph to analyze the similarity between books. The book analysis was used to create a Streamlit web app that used the input of a book title and output book recommendations that were measured to be similar to the input book. This resulted in a successful and user-friendly book recommender web application. Future exploration into different algorithms such as K-nearest neighbors and Euclidian distance clustering could provide an interesting analysis of the accuracy of the different recommender models.

Reference

1. McKinsey & Company. (2013). *How Retailers Can Keep Up With Customer* https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers