Amazon Product Recommender System

Tuan Dang
Computer Science & Data Science Department
Knox College, Illinois, USA
tmdang@knox.edu

Abstract

The amount of data, content, resources, etc in the world continues to grow exponentially each year. In order to filter out so much noise and find the pieces of information most relevant to users, companies need to use machine learning to build helpful and scalable recommender systems. The goal of this project is to build a product recommender system trained on an Amazon product review dataset. The model will use collaborative filtering to train on information from all users from the dataset (i.e. user ratings) to find similarity between products and provide product recommendations.

1 Introduction:

Recommender systems are applications that predict user responses to certain items and provide recommendations based on those responses that would fit the user's needs and preferences. Businesses rely on recommender systems in many ways (i.e. targeted marketing) to improve user experience and increase revenue.

For example, Netflix's former Chief Product Officer indicated that 80% of movies watched on Netflix came through recommendations, and placed the value of Netflix recommendations at more than $1 billion USD per year (Gomez-Uribe et al., 2016). Recommendations also help promote products that may be the right fit for consumers but may be overlooked from not being on the top list. For example, between 20-40% of sales at Amazon are from products that are not the 100,000 most sold products (Lu et al., 2012). These are only a few examples of a wide variety of use cases for recommender systems.

Specifically for this project, our problem statement is to build a system that provides product recommendations based on a target item. In real-life, this use case appears on each item page whenever we see phrases such as "Recommended for you since you viewed this item" or "Other customers also like these items."

The system built for this project employs the use of two models that will combine results to give effective recommendations in terms of relevance as well as diversity with the goal of maximizing click/purchase rate. The two models are an item-based collaborative filtering model trained on user ratings and a content-based model trained on keywords from product descriptions.

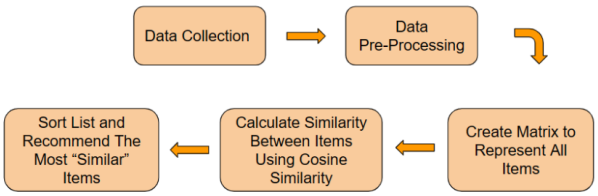
2 Literature Review:

In order to get a better sense of the field, we will go over a quick overview of different methods that other researchers have used to build such systems. Collaborative filtering can be split into two approaches: similarity-based and model-based. Similarity-based methods calculate item similarity for all item pairs and rank the result to provide the items on top of the list as recommendations for the target item. Model-based methods such as matrix factorization or deep learning neural networks do not rely on similarity metrics but instead create embeddings and latent features to look up user-item combinations.

Content-based methods are similar to similarity-based methods in the sense that they also provide recommendations based on similarity between items. They differ in the way they represent items

as vectors, which we will go into deeper detail later on when we talk about model implementation. Lastly, these two approaches can be combined into a hybrid model in some way to take advantage of their strengths as well as cover for each other's weaknesses, which is something that we will attempt in this project.

The following graphic describes the steps to building this system. After collecting and preprocessing the training dataset, we create matrices to represent all items as vectors, calculate the similarity between the target item and all other items, and recommend the top items from that list as outputs. We will now delve deeper into each of these steps.



3 Data and Analysis:

In a paper by Amazon employees detailing their recommender system, they mentioned that "Amazon.com has more than 29 million customers and several million catalog items" (Linden et al., 2003). Data is continuously evolving and grows larger as user activity grows, so the amount of data that they are storing currently would be exponentially larger than that.

For this project, we are using a publicly available dataset of Amazon product reviews (Ni et al., 2019). These researchers crawled data from the Amazon website and created a large repository of datasets for each product category. In order to simplify the problem, we will be focusing on the "Movies TV" category which contains over 3,000,000 reviews from 300,000 users across 60,000 product items. These data points range from May 1996 to October 2018. The data is stored in JSON format, but can be loaded into a pandas dataframe.

Each dataset comes with an extensive list of attributes, although for the purpose of our models, we will only be using a handful. For the collaborative filtering model, we need user and product id as well as the rating (a score from 1 to 5) of each user-product pair. As for the content-based model, we need metadata on the products themselves such as titles, descriptions, brands, or anything else that contains information about the product. We will be using this as keywords for the products.

After collecting the data, we will process it into matrices to use as inputs for our models. From the product ratings, we will build a sparse matrix to store the ratings where the columns and rows represent the items and products, respectively. This matrix tends to be very sparse (in this case, it's only 0.02% filled with data points) since each customer won't be rating that many products and each product won't be bought by most customers so most of the space will be empty.

For the content-based method, we will combine all the attributes mentioned above into the tags feature which will contain all the keywords (in lowercase) for the item. We will also employ the use of stemming to group words together into their word families in order to help the program recognize similar words (i.e. eat, ate, eating should have the same significance in content). Then these texts will be turned into vectors of TF-IDF scores of the keywords in order to represent the corresponding item. We can accomplish this with the use of TfidfVectorizer from the scikit-learn library.

4 Models and Implementations:

Before we get into the model implementation for this project, we should briefly go over Amazon's approach to building their system and how it's different from the one described in this project. Amazon employs a method first launched in 1998 which they termed "item-to-item collaborative filtering" (Linden et al., 2003). The use case in question is providing product recommendations for a target customer from their purchase history. Their algorithm looks at all the items in their history and makes a list of other customers who bought those items as well as items from their lists. Then they calculate the similarity between those items to the corresponding item on the target customer list, rank them, and provide recommendations based on the top results. This

method helps provide recommendations based on other customers' preferences. However, it takes a gigantic amount of storage and time to train these models, so the model needs to be trained offline and the results need to be saved. When it's time for the customer to go on their website, the program can go into their database and perform a look up to provide recommendations in real-time.

The problem with this approach, especially within the scope of this project, is that it is too costly to train such a model on that amount of data, which is why we simplify the problem to only one target item.

Once the data has been pre-processed as described in the previous section, we can move on to calculating the similarity between the target item and all other items with both of our models. This will be done using cosine similarity: items A and B are represented as two vectors and the assumption is that the smaller the angle between those vectors the more similar A and B are to each other. For the collaborative filtering model, the input will be vectors of ratings for the items from all users. As for the second model, the input will be vectors of TF-IDF scores of 5000 keywords for the items. We could have used other methods to tokenize the contents which could be more effective (i.e. Word2Vec or other neural network language models) but TF-IDF is easily understandable and inexpensive to compute.

$$tf\ idf\ (t, d, D) = tf\ (t, d)\ .\ idf\ (t, D)$$

$$tf\ (t, d) = log\ (1 + freq\ (t, d))$$

$$idf\ (t, D) = log\ (\ \frac{N}{count\ (d \in D : t \in d)}\ )$$

(Image Source: https://monkeylearn.com/blog/what-is-tf-idf/)

Here is an example of outputs from each model when the target item is "Harry Potter and the Prisoner of Azkaban":

Collaborative Filtering Model:

- Harry Potter and the Goblet of Fire
- Harry Potter and the Order of the Phoenix
- Harry Potter and the Half-Blood Prince
- Harry Potter and the Deathly Hallows, Part 1
- HP7: Deathly Hallows, P2 (DVD)
- Spider-Man 2
- Shrek 2
- The Lord of the Rings: The Return of the King
- Batman Begins
- Star Wars: Episode III - Revenge of the Sith

Content-Based Model:

- Harry Potter - Years 1-4: (Harry Potter and the Sorcerer's Stone / Chamber of Secrets / Prisoner of Azkaban / Goblet of Fire)
- Harry Potter Years 1-3
- Harry Potter Collection - Years 1-7 Part 1 Region Free
- Harry Potter and the Philosopher's Stone [Widescreen, 2 Disc Edition] DVD
- Harry Potter 1-7 Collection - 8 DVDs (Mandarin Chinese Edition)
- harry potter - 4 grandi film #01 (4 dvd) box set dvd Italian Import
- Harry Potter Years 1-6 Gift set
- Harry Potter Collezione Completa (8 Blu-Ray)
- Seekers Guide To Harry Potter
- The Parables of the Potter

As we can see, the collaborative filtering model not only gives relevant recommendations in order but also introduces the customers to items from different subjects (in this case, it branches out to Spider-Man and Shrek from Harry Potter). This quality of diverse recommendations is lacking from the content-based model since it solely focuses on giving results with content strictly related to the target item. In this case, the recommendations are strictly related to only Harry Potter. This can be good if the item is new or is unpopular since the program can still "reach" it if it has the right keywords. Although for popular items such as Harry Potter, it's not as useful to only stick to other Harry Potter theme items but to branch out to other items so that customers would be exposed to more products, which is more beneficial to the business from a marketing perspective.

5 Evaluation:

The biggest challenge for this project was finding a way to evaluate the model. In real-life, the actual best way to evaluate a product recommender system is to use A/B testing. The machine learning team would launch two or more models on the website for a sample of users. Then they would perform A/B testing to compare results from the control model and the others being tested on. This provides empirical evidence that the recommender can provide value to the business in terms of revenue and/or user engagement.

In a research project such as this where we don't have the resources to perform A/B testing, we would need to come up with an offline method that can predict how relevant the recommendations would be to real customers. For this project, an accuracy metric such as hit rate would make sense since it is intuitive to understand. If the recommended items match what customers have historically bought with the target item, then those items will most likely also be hits in the future. However, the data available for this project was not appropriate for this evaluation method. Many product ids were either missing or could not be matched. An explanation and example can be seen below.

Harry Potter and the Prisoner of Azkaban

Also buy:

- HP7: Deathly Hallows, P2 (DVD)
- HP Double Feature: Year 7 (DBFE) (DVD)
- Harry Potter and the Order of the Phoenix
- Harry Potter and the Order of the Phoenix (Two-Disc Special Edition) by Warner Home Video
- Harry Potter and the Order of the Phoenix/Harry Potter and the Half-Blood Prince (Limited Edition Double Feature)
- Harry Potter Double Feature: Harry Potter and the Order of the Phoenix /Harry Potter and the Half-Blood Prince
- Journey to the Center of the Earth
- Real Steel
- Fantastic Mr. Fox
- Disney's A Christmas Carol

Also view:

- Harry Potter Double Feature: Harry Potter and the Prisoner of Azkaban/Harry Potter and the Goblet of Fire
- HP Double Feature: Year 7 (DBFE) (DVD)
- Harry Potter and the Order of the Phoenix (Two-Disc Special Edition) by Warner Home Video
- Harry Potter: Years 1-5
- Harry Potter Double Feature: Harry Potter and the Order of the Phoenix /Harry Potter and the Half-Blood Prince
- Harry Potter: Wizard's Collection
- Batman Begins

This is the list of items that customers also bought or viewed along with "Harry Potter and the Prisoner of Azkaban," the same example product as before. As highlighted, multiple items on this list contain "Harry Potter and the Order of the Phoenix" which complicates the evaluation process. On Amazon, one product can have multiple vendors or multiple varieties of the same product, and they would be represented by different product ids. Meaning that, even though the human eye can see that these items are a "hit," the program would fail to recognize this across all products since their ids would not match.

6 Conclusion:

In this project, we were able to build a recommender system using collaborative filtering and content-based models which complement each other's strengths and weaknesses. The fact that the problem statement was also simplified to one target item also allowed the system to take much less time and memory to train, which was essential for the scope of the project.

As mentioned in the previous section, a big weakness of this project is the evaluation of the models, and this would be a great place for improvement for future endeavors. Another aspect

that could be explored further is the complexity of the models themselves. The ones explored in this project only scratched the surface of this field. Other models considered to be state-of-the-art and common practice in the industry such as matrix factorization or deep learning neural network can also be applied for this sort of problem to improve performance.

References

Carlos A Gomez-Uribe and Neil Hunt. 2015. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):1–19.

Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53.

Yehuda Koren, Steffen Rendle, and Robert Bell. 2022. Advances in collaborative filtering. *Recommender systems handbook*, pages 91–142.

Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80.

Linyuan Lü, Matúš Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang, and Tao Zhou. 2012. Recommender systems. *Physics reports*, 519(1):1–49.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 188–197.

Michael J Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer.