Clayton Cohn

DSC 478

Prof. Mobasher

4 Jun 2020

Final Project Report

I worked on a group project with Richard Olekanma and Rosario Fabien. We chose to do a recommender system with [this](https://www.kaggle.com/mkechinov/ecommerce-events-history-in-cosmetics-shop?select=2019-Dec.csv) dataset from Kaggle. The dataset is comprised of a month's worth of transactions from an online cosmetics store. Because there are three of us, we each took a different approach. Rosario tackled the problem using clustering, Richard did user-based collaborative filtering with matrix factorization, and I did item-based collaborative filtering. Each of us individually turned in our notebooks, videos, and project reports for our respective areas.

The first thing I had to do was clean the data. There were a lot of extraneous columns, so I got rid of the ones like "user\_session." I also ignored all of the transactions that didn't have "purchase" as the event type. Since I did an item-based collaborative filtering system, I was only concerned with purchased items, and not with "viewed" items or items "added to card." After I isolated the "purchased" transactions, I dropped the "event\_type" column. Next I had to reduce the number of products. There were almost 27,000, and trying to compute an item-item similarity matrix crashed my computer. I opted to only use the 1000 most frequently purchased products, as after 1000 the data became exceedingly sparse. After factorization, I was left with a cleaned up matrix of every product purchase where one of the top 1000 products was purchased.

After I cleaned the data, I created a dictionary of all the data where each key represented a user and each corresponding value was an array of every item that user had purchased. I then took that data and made a Pandas DataFrame of user/product/count combinations. For example, the entry [0,0,1] says that user 0 bought product 0 once. Creating a pivot table from this gave me vector representations for all users across all 1000 products by number of purchases. Next I transposed the user matrix to get an item matrix, and used Scikit-Learn' cosine similarity function to create an item-item similarity matrix.

Once I had the similarity matrix, I was able to recommend products to a given user. My recommend function takes a user as a parameter, isolates the user's favorite product (based on purchase count), and spits out the n closest products (I used n = 5). To evaluate this, I checked every user's top 5 recommendations, and then checked the user to see if they had actually purchased any of the recommendations. In this model, the user had purchased at least one of the recommendations roughly 26% of the time.

I felt that the results were good, but obviously not representative of reality. For one thing, I dropped all non-essential columns, and they could have added an additional degree of dimensionality to help improve the system. I also only used 1000 products due to memory constrictions. The original data had roughly 27,000 products—most of which were sparsely populated. Lastly, I did not split the data, so there is almost certainly some bias in the training. I was unable to find any real-world data relating to the percentage of time users bought recommendations placed in front of them from which to compare my system to.