Vince Laird

CAPSTONE 2 PROJECT: RECOMMENDER SYSTEM

DATA SCIENCE - CAREER TRACK

E-commerce: Recommending items

The second capstone project will be based on the RecSys Challenge 2015. The website and details about this challenge are here: http://2015.recsyschallenge.com/

This is a problem of coding a recommender system. The data used provides a sequence of click events / click sessions. For some sessions, there are also buy events. The goal is to recommend a number of items to a user, based on the initial item they have viewed.

This information is valuable to e-commerce businesses as it can indicate what items to suggest and how to encourage a user to buy (such as via promotions, discounts, etc.)

The data represents six months of activities of an e-commerce website selling random items such as garden tools, toys, clothing, electronics and more.

Data set

The data set provided on the RecSys website contains click information, buy information and test data. The click information and buy information will be utilized; since the challenge is over, determining a final score using the test data will not be possible, as the correct "answers" for the test data are not available.

Exploratory data analysis was performed on the data; the click data set contains 33M rows of information, and the buy data set contains 1.15M rows. Further information regarding the data set can be found in the notebook here:

https://github.com/vincelaird/springboard/blob/master/capstone2/Capstone%202%20milestone%20notebook%20EDA.ipynb

Approach

The goal was to use two similarity distance measurements (cosine similarity and Jaccard similarity) to determine which items in the dataset are related. Content-based recommendation will be used; we know the content of both session and item. The similarity of items can be computed using similarity scores.

Details of the steps taken can be viewed in the notebook at the following link:

https://github.com/vincelaird/springboard/blob/master/capstone2/Capstone%202%20recommendation%20system.ipynb

First challenge was setting up Amazon Web Services (AWS) to run the notebook, which would be very processor intensive given the size of the datasets. This involved creating EC2 instances, configuring Jupyter Notebook to allow remote access, and setting up security groups and keys.

Once AWS was up and running and the datasets were uploaded, the next step was to load the data and create a pivot table so that each row would correspond to a unique session, and each column would be a unique item.

Problem with this step was that there were 9.25M unique sessions, and 52,739 unique items. The dataset was sparse: the average session lasted only long enough to view 3.5 items. Creating a pivot table proved to be a challenge due to memory errors.

Various methods were utilized to create a pivot table that included all (or most) of the data: 20 different dataframes were exported to .csv files (and from the buy dataset only, which included only buy events: 510K unique sessions, 20K unique items), however only the first 3 (containing 50K rows each) out of 20 would load into memory using 8 GB of RAM.

A new EC2 instance was created with 256 GB of RAM, which required all of the initial EC2 setup steps to be repeated, however only 250K rows of the buy dataset would load.

At this point, use of the clicks dataset was scrapped. The original idea was to utilize the clicks dataset, and then combine with the buys dataset - also, the buys would count five times what the views (or clicks) would count as, with the assertion that items bought together (or sessions with a buy) would be weighed more heavily.

Subsetting buys dataset

Given the difficulty of working with the entire dataset, to show proof of concept, only the first 50,000 rows of the buy dataset in an EC2 instance with 8 GB of RAM was used. 55% of sessions in this subset contained purchase of just one item, so over half of the sessions would be useless to determine which items were similar as only one item corresponded with one session. However, there still were items that were purchased together, and both cosine and Jaccard similarity rankings were determined from the data available.

Selecting the first item to appear in the dataset (ItemID 214507331, no descriptions of the items were provided), it can be seen that for all sessions that contained a purchase of this item and at least one other item, 16 unique items were purchased in conjunction with this one.

Using cosine and Jaccard similarities, it was determined that two items (214603138 and 214648247) had the highest similarity to ItemID 214507331. This indicated that the process worked and could be scaled.

Also, one specific row of the dataframe was added four additional times, to create a multiplier effect which would simulate weighing a particular purchase more heavily than other purchases. This changed the cosine similarity rating substantially, with the multiplier item being rated as the most similar, proving that purchases could be weighed more heavily than clicks, if both datasets had been utilized.

Next steps

Though the inability to analyze both the clicks dataset and buys dataset is disappointing, both cosine and Jaccard similarity scores showed logical results. Both of these methods were able to assign similarity “values” to all items that were purchased along with another specifically chosen item. Loading the entire dataset would be a great next step, and methods using distributed processing would make this a possibility.

Pandas also has sparse data structures, which could be used to load more data while taking up less memory. These sparse data structures can be viewed as “compressed” where any data matching a specific value is omitted.

Conclusion

Recommender systems using cosine and Jaccard similarity scores will give values for the relationship between items. Given one item ID, several other “suggested” items can be determined. Comparing these suggestions to items that were viewed in other sessions illustrates the effectiveness of these similarity scores.

Slide deck

Slide deck presentation of the complete project can be found here:

https://github.com/vincelaird/springboard/blob/master/capstone2/Capstone%20project%202%20-%20recommendation%20system%20slides.pptx