Product Recommendation for Ecommerce Website

Problem Description

According to Statista, over 4.33 billion people access the internet globally. Many service providers and retailers have moved online and the popularity of ecommerce platforms continue to rise . Unlike physical stores, online retailers are not restricted to limited shelf space. Retailers are able to sell a wide variety of products online because the costs of logistics is lower. However, the huge amount of goods available makes it difficult for customers to navigate through their product of interest. To influence the user's buying decision and increase the company's revenue, we will create a product recommender system to provide a suggested list of items with the current item being viewed.

To build a recommendation system, there are several potential challenges we must consider. First, we must select an approach that is appropriate for the context and dataset. In this project, the recommender is given a classification task since it should produce a limited list of top-ranked items. Based on user preference data, this will help us determine the type of rating the recommender will use to compare similarity of items. After establishing an approach, we would also need to select an evaluation metric and devise a way to partition the data for modeling. Another challenge to think about is how we can alleviate the shortcomings of recommender systems, such as making suggestions for new items or users in the cold-start problem.

Dataset Overview

Retailrocket, a service provider that creates product recommendation engine for eCommerce platform, has released data from a real-world ecommerces website on Kaggle. The dataset contains three tables: events, category tree, and item properties. The events table captures visitors' behavioral data that was collected over 4.5 months in 2015. A visitor can perform three types of events: view, add-to-cart, or transaction. In total, there are 2,756,101 events, which consists of 96% views, 2.5% add-to-carts, and 0.8% transactions. The website received 1,407, 580 unique visitors during the collection period. Approximately 90% of itemIDs in the events table corresponds to the item properties table.

The item properties table consists of four columns (timestamp, itemid, property, and value) which provides a concatenated snapshots of 417,053 unique items for every week during the collection period. If a property of an item remains constant over the observed period, there will only be a single snapshot value in the table. If the price of an item changed three times within 4 weeks, there would be three different properties for that item. There is a total of 20,275,902 rows of item properties. All values of properties were hashed, except 'categoryid' and 'available'.

Finally, the category tree table consists of 1,669 rows and two columns with a categoryid and its corresponding parentid. Not all categoryid have a parentid.

Table I: Events

Timestamp	Visitor ID	Event	Item ID	Transaction ID
1433221332117	257597	view	355908	NaN
1433224214164	992329	view	248676	NaN
1433221999827	111016	view	318965	NaN

Table II: Item Properties

Timestamp	Item ID	Property	Value
1435460400000	460429	categoryid	1338
1441508400000	206783	888	1116713 960601 n277.200
1436065200000	285026	available	0

Table III: Category Tree

Category ID	Parent ID
1016	213.0
809	169.0
570	9.0

Methodology

Data Preprocessing

As mentioned above, item properties are hashed due to privacy issues, which makes it difficult to interpret results if a content-based approach was taken. Instead, we can recommend item pairs that were purchased by previous customers from the events table. With the assumption that the data is sparse, we could clean the data by removing users with no activity and items with only a single view or activity.

Model

LightFM is a python recommendation algorithm that can work with the implicit feedback in our dataset. It also incorporates item and user properties of collaborative filtering into a matrix factorization to reduce dimensionality. We would also need to construct an user-item or item-property matrix that would be used for train and test split.

Evaluation

We will use classification accuracy metrics, like ROC AUC, to assess the recommendation algorithm ability classifying relevant and irrelevant items. Recall will be weighted more than precision since we are most interested in the probability that a relevant item is recommended.

Reference

M. Daoud, S.K. Naqvi, "Recommendation System Techniques in E-Commerce System", International Journal of Science and Research 4(1) 569-573, 2015

G. Linden, B. Smith, and J. York, "Amazon.com Recommendations: Item-to-item Collaborative Filtering", *IEEE Internet Computing,* IEEE Computer Society, 2003, pp. 76-80

G. Schroder, M. Thiele, and W. Lehner, "Setting Goals and Choosing Metrics for Recommender System Evaluations",