Personalized Fashion recommendation system

Team: Sai Krupa Jangala(sj3140), David Heagy(dh2868), Karunakar Gadireddy(kg2911), Jugal Shah(js5950), Jiayuan Cui(jc5670)

Overview

Many online shopping sites face the issue of customer's scrolling through their page but not making a purchase. We decided to build a product recommendation system based on H&M's data from previous transactions, as well as the customer's and product meta data. The data includes garment type, customer's data, product description, and image data of the product. The recommendation system will help the customer decide which product to buy easily from the plethora of options. A well developed recommendation system will help businesses improve their mac. An efficient recommendation system reduces the risk of return and thereby reducing the cost the firm pays for the logistics on the return policy.

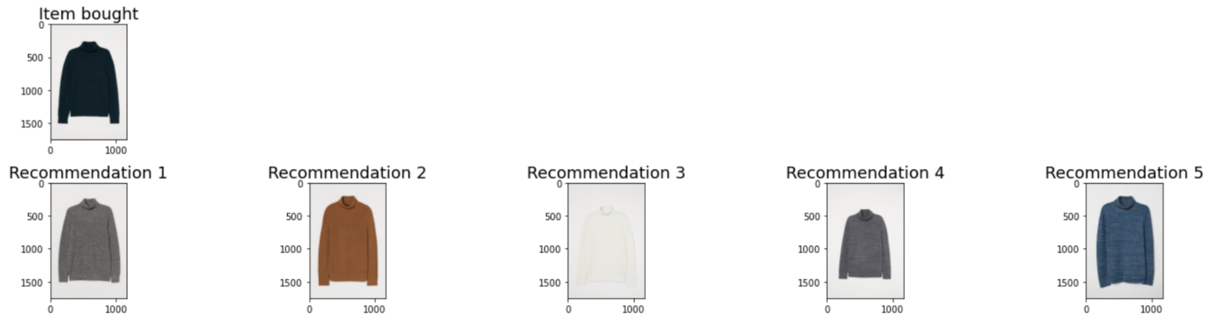
Data Description

The dataset gives us the purchase history of customers across time, along with supporting metadata. Our goal is to predict what articles each customer will purchase in the 7-day period immediately after the training data ends. We have three csv files with different information along with the image of each article. File articles.csv contains article_id, which shows available articles that can be purchased. Folder images/ contains images corresponding to each article_id. File customers.csv contains customer_id, which has the detailed information of customers. File transactions_train.csv contains the training data, consisting of the purchases of each customer for each date.

Methodology

K-Means

We used different recommender algorithms in our project, the first one we used is the k-means clustering algorithm, which is an unsupervised machine learning algorithm that can be used to categorize data into different groups. We use this algorithm to categorize all articles into different clusters, and then group_id gives us the top 5 recommendations. Before executing our algorithm, we counted the number of articles bought by each customer, dropped duplicate customers and sorted by the number of articles and the time. Then, we generated features based on feature expansion, so we got a list of all unique article ids. Last, we dropped the categories. After the data cleaning, we implemented the k-means clustering algorithm. Firstly, we selected the numbers of clusters which we want for our dataset. In order to find the right K we are trying to use the elbow method for the same for the selection of the optimal number of clusters. We got an elbow which is around 12. Then, we have to select k random points called centroid from our dataset. The k-means algorithm will assign each data point to its closest centroid which will finally give us k-clusters. Once we had these clusters, we will be predicting clusters for each data point and once we get the cluster, we tried to find the cosine similarity between the given point and all the points in the cluster. We picked the top point whose cosine similarity is the maximum with the given data point as our recommendation. We computed metrics like Jaccard similarity and Cosine similarity for this.



Content Based Filtering

Our second system is based on content based filtering. We created a customer matrix and item matrix based on certain available features for both such as “colour_group_name”, ”graphical_appearance_name” etc. Once we have these two matrices we normalize them and take the dot product of the two. This dot product will generate a matrix where each row corresponds to a customer and each column corresponds to an article. Each element in the matrix corresponds to a score and the higher this score the better is the article as a recommendation for the corresponding customer. Items that have already been purchased are ignored when suggesting new items for each customer.

Content based filtering methods can be hard to scale as the memory and computational cost increases exponentially with increase in number of customers or items. One way to address this issue is to perform PCA on both the user and the item matrix to reduce the size of these matrices without losing performance.

Figure below shows previously purchased items of a customer along with what the content based recommendation system recommended for that customer.

Alternating Least Square

Our third recommendation system is based on alternating Least Squares (ALS), which is a collaborative filtering method. Matrix factorization algorithm for explicit or implicit feedback in large datasets, optimized for scalability and distributed computing capability. In our project, we only took last month's data for training. Then we got the count of purchased items by user and item, and created Sparse User-Item COO Matrix. The ALS model factors user-to-item matrix A into user-to-feature matrix U and item-to-feature matrix M. The ALS model is used to calculate U and M, such that A is approximately = U x M. The Alternating part comes from optimizing U and fixing M and vice versa.   After the training process, we randomly sampled users with their previous purchases, fed into the model, and returned the same number of recommended items for comparison.

Metrics:

We have compared our models against multiple similarity metrics

	Cosine Similarity	Jaccard Similarity	Euclidean Distance	Pearson's Correlation

ALS	0.994	0.294	72093.78	0.9956
Content Based Filtering	0.989	0.989	143132923.45	-0.1416
K-Means	0.466	0.543	Very high value.	0.99

As we can observe, we ran different evaluation metrics to compute the similarity between the purchased item and the recommended items. We have used K Means as our base line model and proceeded with advanced techniques like ALS and conten based filtering. We have observed good enough accuracy for both of them, however ALS performed slightly better compared to Content Based Filtering.

Future work:

We have currently only explored simple recommendation system engines. Given more time we would like to use the various meta data available to construct and train more complex models such as deep neural networks.

Our current comparison metrics are based on similarity which means that our metrics are good if we suggest items similar to the ones already bought by the customer. However the customer might not buy the same or similar product again. We can also explore other more robust comparison metrics to compare the efficiency of our models.

There also might have been a better way to sample, as opposed to random sampling, since there is a temporal part to recommendation systems, given that different time periods do not have all users nor all the items they bought overall.