Hypothesis and Algorithm for best Offer recommendations

Challenges:

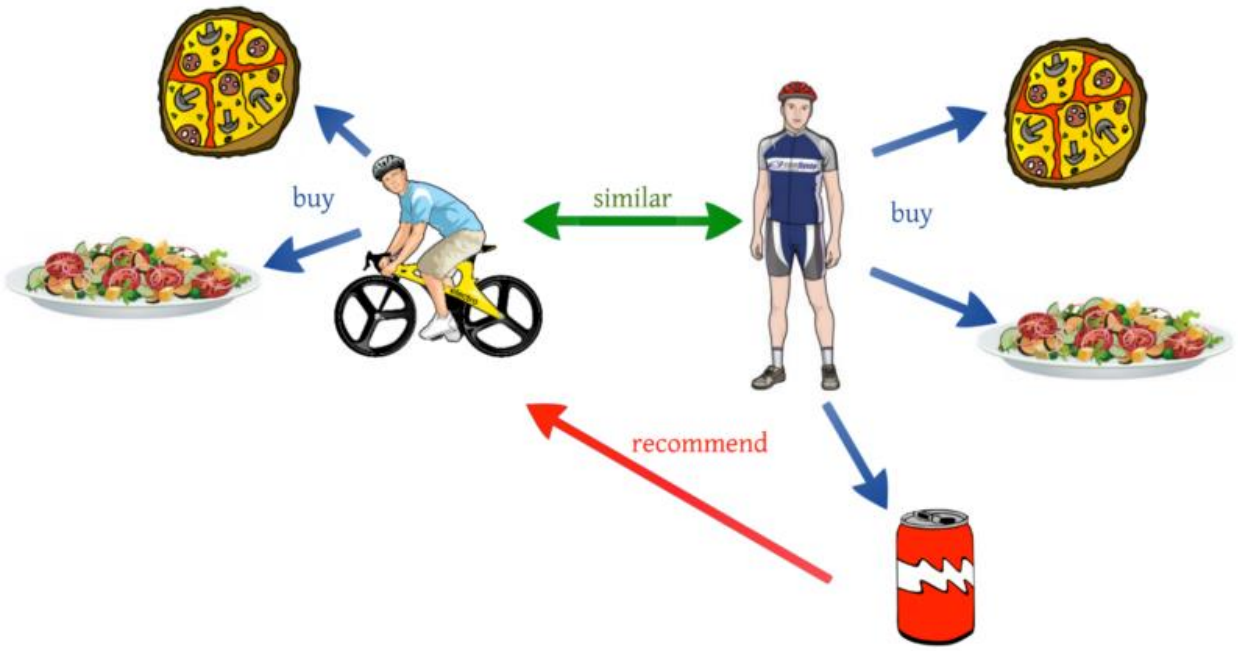
- Huge amount of data, millions of customers and very large number of offers
- Producing high quality recommendations in very less time (less than a second).
- Producing Recommendations for New customer with very limited information, based on only a few offers that he is showing interest in.
- Volatile customer data

Three Hypotheses for Recommendations

1. Collaborative filtering
2. Cluster models
3. Item-to-Item collaborative filtering

Note- Here keyword "item" refers to "offer"

Collaborative filtering



The collaborative filtering represents a customer as an N-dimensional vector of items(offers), where N is the number of distinct items (offers). The components of the vector are positive for viewed/explored or positively rated items (offers) and negative for negatively rated items.

The algorithm generates recommendations based on a few customers who are most similar to the user. It can measure the similarity of two customers, A and B, by measuring the cosine of the angle between the two vectors-

$$similarity(\vec{A}, \vec{B}) = \cos(\vec{A}, \vec{B}) = \frac{\vec{A} \circ \vec{B}}{\left|\left|\vec{A}\right|\right| * ||\vec{B}||}$$

The algorithm can select recommendations from similar customers'items by ranking each item according to how many similar customers viewed/explored it.

Disadvantage:

Data Sparsity: In case of large number of items, number of items a user has viewed/ explored or rated reduces to a tiny percentage making the correlation coefficient less reliable.

Cluster models

To find the customers who are similar to the user cluster models divide the customers base into many segments and treat it as classification problem.
The algorithm's goal is to assign the user to the segment containing the most similar customers. It then uses the viewed/ explored items(offers) of the customers in the segment to generate recommendations.
The segments typically are created using clustering algorithm or can be manually determined. Using similarity metric, a clustering algorithm groups the most similar customers together to form cluster or segments.
Because optimal clustering over a large data sets is impractical, various forms of greedy cluster generation can be used to solve this problem.
Once the algorithm generates the segments it computes the user's similarity to vectors that summarize each segment, then choses the segment with the strongest similarity and classifies users accordingly.

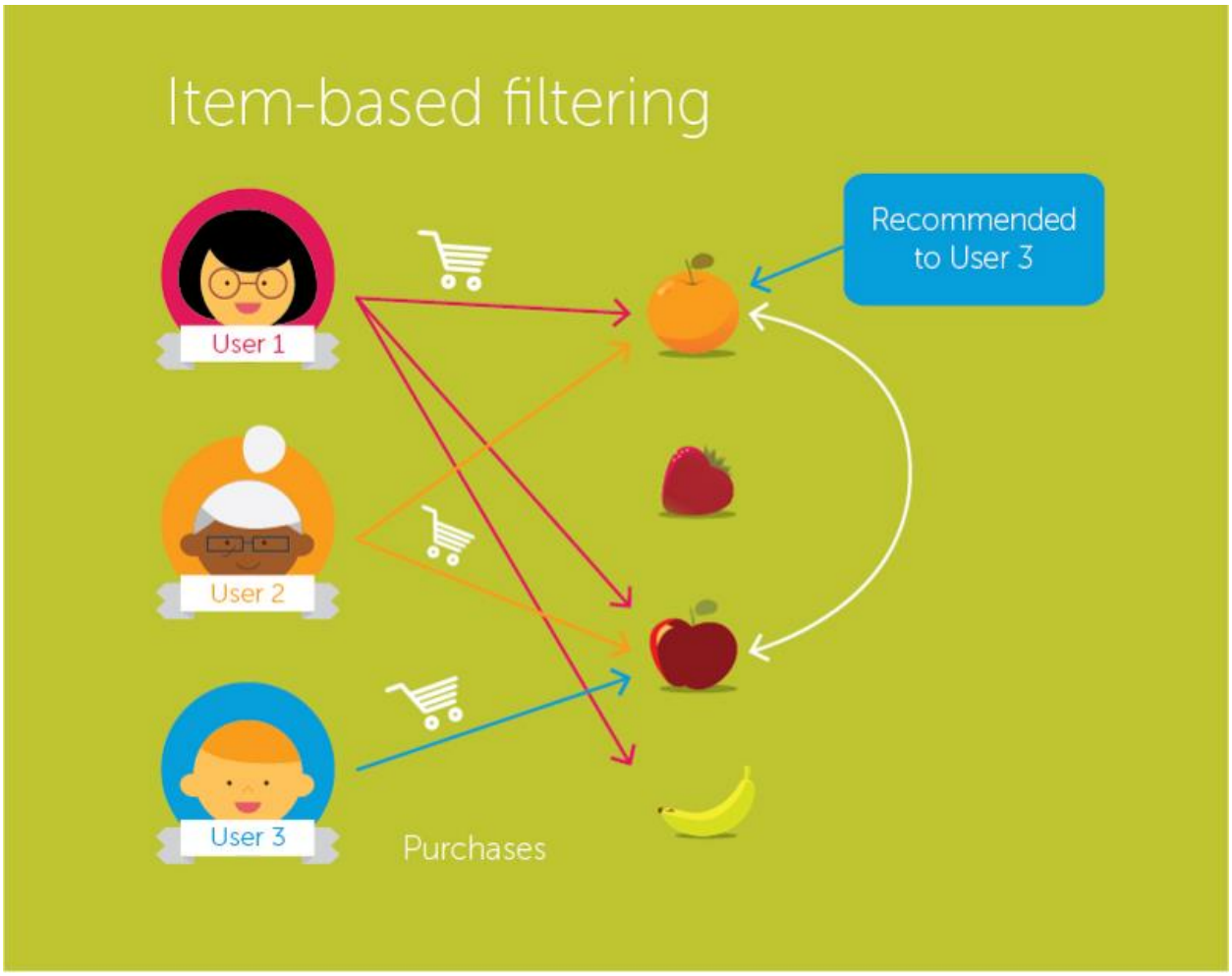
Advantage over traditional collaborative filtering

Better online scalability and performance because they compare the user to a controlled number of segments, rather than the entire customer database. The complex and expensive clustering computation is run offline

Disadvantage

Poor recommendations quality- Cluster models group numerous customers together in a segment, match a user to a segment, and then consider all customers in the segment similar customers for the purpose of making recommendations. Because the similar customers that the cluster models find are not the most similar customers, the recommendations they produce are less relevant. It is possible to improve quality by using numerous fine grained segments, but then online user-segment classification becomes almost as expensive as finding similar

Item-to-item collaborative filtering



ITEM-ITEM collaborative filtering look for items(offers) that are similar to the items that user has already viewed/explored and recommend most similar items(offers).
Item-to-item collaborative filtering, scales to massive data sets and produces high quality recommendations in real time.

Algorithm

Rather than matching the user to similar customers (as in case of traditional collaborative filtering), item-to-item collaborative filtering matches each of the user's viewed and rated items(offers) to similar items(offers), then combines those similar items(offers) into a recommendations list.
To determine the most-similar match for a given item(offer), the algorithm builds a similar-items table by finding items(offers) that customers tend to view/explore.

The following iterative algorithm provides the best approach to build similar-items table by calculating the similarity between a single and all related offers.

for each offer O_1 in offers:
    for each customer C who viewed/explored O_1:
        for each offer O_1 customer C viewed/explored:
            record that a customer viewed/explored O_1 and O_2

    for each offer O_2:
        compute the similarity between O_1 and O_2

To compute the similarity between two items(offers) calculate the cosine angle between the vector corresponds each item(offer), rather than a customer, and the vector's M dimensions correspond to customers who have viewed/explored that item(offer).

$$similarity(\vec{A}, \vec{B}) = \cos(\vec{A}, \vec{B}) = \frac{\vec{A} \circ \vec{B}}{\left|\left|\vec{A}\right|\right| * ||\vec{B}||}$$

Then from the similar-item table algorithm find items(offers) similar to each of the user's viewed/explored item and then recommends the most popular or correlated items(offers).

Complexity Analysis

Offline computation of the similar-items table has $O(N^2M)$ as worst case time complexity, however, it's closer to O(NM), as most customers have very few viewed/explored items(offers). Sampling customer who views/explores top-offers titles reduces runtime even further.

The online time computation that user will actually experience is very quick, depending on the number of items the user viewed/explored or rated.

Comparision with Traditional collaborative and Cluster models:

- Traditional collaborative filtering does little or no offline computation, and its online computation scales with the number of customers and offers. The algorithm is impractical on large data sets. Unless it uses dimensionality reduction. Sampling which reduce recommendation quality.
- Cluster models can perform much of the computation offline but recommendation quality is relatively poor.

Conclusion

Item-to-item collaborative filtering is best suitable for our need because-

- It creates the expensive similar-items table offline.
- The algorithm online component –looking up similar items(offers) for the user's viewed/explored and rated items(offers) scales independently of the catalog size or the total number of customers, it only depends on how many offers the user has viewed/explored or rated.
- Offer recommendation quality is very high in compared to traditional collaborative filtering and cluster models
- Unlike traditional collaborative filtering the algorithm also performs well with the limited data.