Recommendation algorithms are best known for their use on e-commerce web sites where they use input about a customer’s interests to generate a list of recommended items.

At amazon.com, we use recommendation algorithms to personalize the online store for each customer.

The click-through and conversion rates vastly exceed those of untargeted content such as banner advertisements and top-seller lists

* Large retailers have huge amounts of data – lots of customers and items
* Results set must be returned in real-time
* New customers typically have limited information, older customers can have too much information
* Customer data changes regularly, algorithm must respond immediately to new info

There are three common approaches to solving the recommendation problem:

* Traditional collaborative filtering
* Cluster models
* Search based methods

Compare these to Amazons item-to-item collaborative filtering

This algorithm scales independently of the number of customers and items

Produces recommendations in real-time, scales to massive data sets, and generates high-quality recommendations

Two popular versions of recommender algorithms are collaborative filtering and cluster models

Other algorithms focus on finding similar items not similar customers. For each of the user’s purchased and rated items, the algorithm attempts to find similar items. It then aggregates the similar items and recommends them.

Typically multiplies the vector components by the inverse frequency making less well-known items much more relevant. For almost all customers, this vector is extremely sparse.

Generates recommendations based on a few customers who are most similar to the user

Collaborative filtering to generate recommendations is computationally expensive O(mn) worst case because average customer vector is extremely sparse, tends to be O(m+n)

For very large data sets, the algorithm encounters severe performance and scaling issues.

Possible to partially address these scaling issues by reducing the data size by randomly sampling customers or discarding customers with few purchases and reduce N by discarding very popular or unpopular items.

Dimensionality reduction techniques such as clustering and principal component analysis can reduce m or n by a large factor

Unfortunately, al these methods also reduce recommendation quality in several ways

Cluster models divide the customer base into many segments and treat the task as a classification problem.

Segments typically are created using a clustering or other unsupervised earning algorithm

Because optimal clustering over large data sets is impractical most applications use various forms of greedy cluster generation

For very large data sets especially those with high dimensionality, sampling or dimensionality reduction is also necessary

Cluster models have better online scalability and performance than collaborative filtering

Complex and expensive clustering computation is run offline however recommendation quality is low

Search or content based methods treat the recommendations problem as a search for related items.

For users with thousands of purchases however, it’s impractical to base a query on all the items. Must use a subset or summary of the data, reducing quality

Item-to-item collaborative filtering

Targets to each customer

Existing recommendation algorithms cannot scale to amazons tens of millions of customers and products, we developed our own. Our algorithm, item-to-item collaborative filtering matches each of the user’s purchased and rated items to similar items then combines those similar items into a recommendation list

Builds a similar items table by finding items that customers tend to purchase together

Common method is to use cosine measure

Offline computation of the similar items table is extremely time intensive O(n2m) as worst case, in practice however, closer to O(nm)

Sampling customers who purchase best selling titles reduces runtime even further with little reduction in quality

Given a similar items table the algorithm finds items similar to each of the users purchases and rating aggregates those items an then recommends the most popular or correlated itesm this computation is very quick, depending only on the number of items the user purchased or rated

For very large data sets, a scalable recommendation algorithm must perform the most expensive calculations offline