1a.

Sim(user1,user2) – 1/6

Sim(user1,user3) = 1/6

Sim(user2,user3) = 4/6

1b.

Cos(a,b) = 0.6065

Cos(a,c) = 0.5130

Cos(b,c) = 0.6139

1c.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Item 1 | Item 2 | Item 3 | Item 4 | Item 5 | Item 6 | Avg |
| User 1 | 0.25 | 1.25 |  | 1.25 | -2.75 |  | 3.75 |
| User 2 |  | 0.4 | 1.4 | 0.4 | -1.6 | -0.6 | 2.6 |
| User 3 | -0.5 |  | -1.5 | 0.5 |  | 1.5 | 2.5 |

1d.

Cos(a,b) = 0.3544

Cos(a,c) = 0.6821

Cos(b,c) = -0.544

2.

One of the most popular companies to use recommender systems is Amazon. It is essentially how their business is operated. It considers what the user is currently looking at then recommends other products that users who bought that product would also buy with that item. Not only amazon was using item-item filtering for their products. Companies like Netflix, YouTube, and more were jumping on this algorithm. It was found that about 30% of Amazon’s page views came from recommendations. A major thing I learned from this reading is the importance of time. It is important to use only relevant data when recommending. Someone could’ve bought a product I am buying today two years ago. There is a good chance that what was bought with it back then is not relevant today. So, it is important to keep in mind when recommending the timestamps of when things were bought, and the recommended items are relevant to the time. The future of recommender systems is still huge. There can always be more data to mine and a better way to recommend items. Another important topic discussed was defining what related meant. I think this was the most important part of the first reading. Finding the right equation to compute what related is, is everything for your system. If your recommender system has a bad equation, then it will produce bad results. It is why it is important to keep refactoring and updating you recommendation system. The thing I find interesting about this field is it is always changing and growing. People are always finding new ways to improve the recommender systems and make them more relevant and efficient.

The second paper goes in depth about traditional and cluster models. The equation shown in the traditional section looks a lot like the one we are studying. It is interesting to learn about the scaling issues when it comes to dealing with large data sizes. It was recommended to reduce the customer with fewer purchases and some randomness to increase efficiency. The downside to that is it reduces the actual recommendation quality in many ways since you are removing bits of data. It has less of a pool to analyze through. I found the cluster model really interesting to learn about. It takes the customer base and separates it into segments and treats each task as its own classification problem. Once the segments are made it computer the user similarity to vectors too summarize each segment and then chooses the segment with the strongest similarity. This method seems to have better scalability and it was really interesting to learn about this method. Another method described is Search-Based models. I found this one interesting also since it takes a user’s purchased and rated item and it creates a search query to find other items with similar key words. I like this one for its simplicity. It seems very intuitive and straight forward to understand. I can see how it may be inefficient, looking at the implementation it has a complexity of , which can take a long time especially if the data set is huge.

Overall, between the two articles I noticed a common theme. Scalability is always considered and talked about in each model. It is always important to keep scalability in made since you want your algorithm to run in the most efficient way possible. Recommender systems are a very powerful tool that have helped business grow in many ways possible, and it is interesting to watch this area grow and become more sophisticated .