Jeonghyeon Woo (the only team member)

Cpts 415 – Big Data

Final Project

Video Recommender

Platform business became one of the strongest industries nowadays. As platforms simply provide places of dealings, it is obvious that they could make much money without spending that much. However, this feature led many companies be into platform business; platform business became red ocean now. In the red ocean, platform companies had to find a way to have more users than other companies of the same market; recommendation system was the way. For instance, the main factor of Amazon's success is that Amazon knows what you want. What Amazon suggests is what you want to buy, so it is a natural result that you are buying Amazon's products. A recommendation system is a platform's competitiveness, and so video recommendation system is the competitiveness of video viewing platforms.

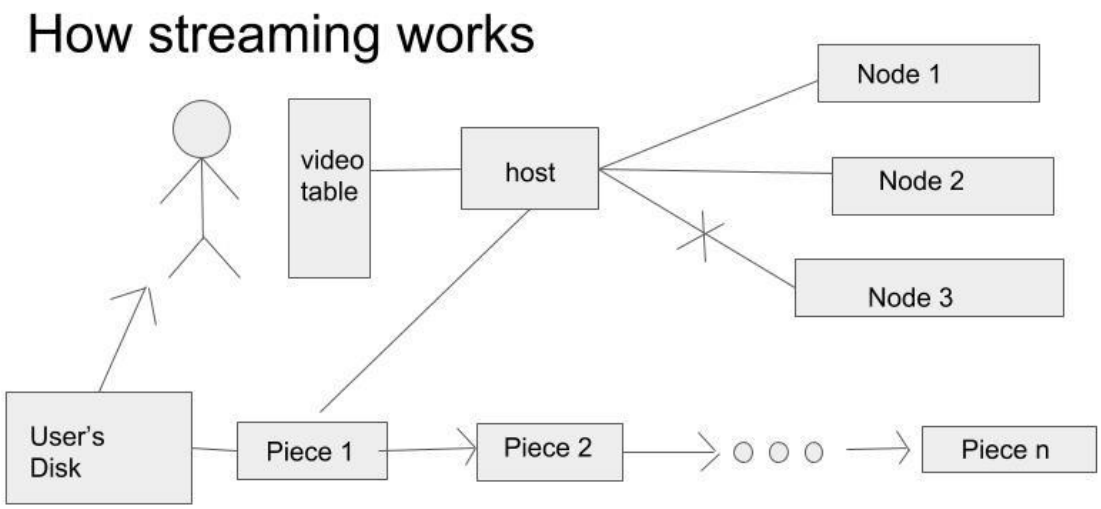
Video viewing platforms, such as Youtube, Netflix, Twitch, have their own recommendation systems. Youtube recommends videos with these criterions: subscribed, watched by related users, searched, video of the user's group/cluster(of the same taste), top/hot/most viewed videos, and related videos. Subscription is very straightforward: users choose what they want to see. New videos of subscribed users are constantly recommended unless it is unsubscribed. Videos watched by related users are also recommended. For instance, if A subscribed B and B watched video q, then q is recommended to A. Related user is defined as users in the same cluster/group or uploaders of watched videos. When a user searches for a specific

type of videos, the system records this behavior. According to the user's records, the system recommends related videos. Finally, above criterions are used for clustering/grouping the user. Netflix has similar recommendation system to Youtube, but it regards playtime significant. In contrast to Youtube that streams user created contents, Netflix tends to stream videos of longer playtime such as movies or dramas, so playtime is one of the most significant features when regarding a recommendation.

Those recommendation systems are mainly focused on videos. However, the systems may perform better recommendations if they focus on scenes rather than videos itself; when a user especially likes a specific scene, the system could recommend videos including scenes that the user would like to see.

How Streaming Works

Before Youtube, video viewing application could not be implemented as platform since video files make huge traffic when users load and upload. In addition, loading and uploading were failed when the host had bad connection to nodes. Youtube solved this problem by streaming videos. When a user chooses a video from video list, the host requests server stream pieces. The host is connected to several nodes, and the nodes are designed to have CAP, so the quorum could handle loading and uploading: the host loads and uploads small stream pieces instead of whole video file, and it becomes easier to implement distributed data management.



Implementation of Suggested Problem

To implement the problem, let the design focused on stream pieces rather than videos. Suppose user cluster/group is already implemented. With the given user cluster/group, when a user is loading some specific scene again, reposition the cluster point in the user's cluster or cluster the user in the user's group. To record users' behavior on scenes, add a relation to user query: name of relation is scene, and video name is foreign key.

Let scene query includes scenes. For each scene, the stream header will be the primary key, and list of users who liked the scene will be the value. To define scenes, let e be a range value such that a scene is $(\mathrm{p}-\mathrm{e},\ \mathrm{p}+\mathrm{e})$ for p is indicating which position a given stream piece points. With e, the initial set of scenes will be like this: scene: 0, 2e, 4e, 6e … end scene. e could be modified depending on the total playtime of the video. For instance, e may indicate 3 stream pieces when the number of total stream pieces is 24 and the number of total scenes is 5.

Claim: user, video, *sp <map> &list

For scenes in video:

If sp is in range of scene:

sp=&scene.header

user.scene[video].[scene.header] 1 if null, else ++

return &scene.list

user.scene[video][sp->header] 1 if null, else ++

scene[video][sp->header]=empty list

return &scene[video][sp->header]

input: user, video, stream_piece response: list

Let a be significance indicator such that a scene is significant when a user has loaded a specific scene more than a times.

Claim: user, video, *sp <point adjust> 1

*list=map(user, video, sp)

if user.scene[video][sp->header] > a:

if list is none: list points array that include only the user and return 1

else:

if user not in list: append the user

reduce(user, list, video)

return 1;

else return 0;

The algorithm is as followed. When a user loads stream pieces, map the pieces. After map is occurred, sp will now point the very head of the scene, and point adjust process will get list pointing users who liked the scene. If the user has loaded the stream piece more than a times, the system will regard it is significant; it is under assumption that the user played the video for significant playtime. If the list includes no member, then the list will point to the user. Else, reduce is occurred. Reduce will reposition the user's cluster point or cluster the user in the user's group.

Algorithm of Reduce

Case 1. User is in cluster and reached by other points.

It is simple to implement reduce in this case. A cluster map handles taste group, and the user already has cluster point in the cluster as well as other users in the list have. With the cluster points, reposition the user's cluster point: new position could be mean, median, etc. The system will recommend videos watched by users who the user has reachability.

Case 2. User is in user groups of similar tastes.

Suppose there are taste groups. One taste group can not explain a user's taste. For instance, if a user likes Star Wars and Last Samurai, the user could be in three taste groups: blade fighting, SF, alternate history. The user's taste could be explained by those three groups, but it is impossible to explain the user's taste with

only one among those. Suppose $A = SF, B = \text{alternate history}, A \cap B = \text{blade fighting}$. It is not sure if the user would like $A \cap (A \cap B)^c$. For this reason, a user will be in multiple user groups.

After reduce, users would have cluster points in their user groups: it is double clustered. Use the given list to cluster, but a difference to case 1 is that users who are not in any of given user's groups will be ignored. Cluster will be done for the user's groups that are related to the given video.

Result

The approach for reduce has inefficiency for both cases. For case 1, the cost is relatively lower than the cost of case 2. However, recommendation pool of case 1 is too broad. This could cause a problem that the system is recommending videos what users don't want to see. For example, fantasy movie may be recommended to a user who does not want to see fantasy though the user might like some scenes in the movie. Case 2 will provide fair recommendations. However, it has big cost since cluster is done for all related groups.

In addition, it is not determined when to clear the data of scene relation of user. This problem is not significant when the data pool is small, but it will be a big problem when a user watched, for instance, 10000 videos. The user's scene relation will have all the records of scenes of the videos and will not release them.

Another problem is that users may not reload scenes. For instance, when a user plays a video that has playtime of 1 hour and never goes back to any scene, scene recommendation will not be happened. However, in this case, the user has

significant playtime, so related videos will be recommended anyways.

For an overall observation, scene recommendation is good at assuming users' specific preferences. However, it is weak at assuming users' general preferences. In addition, it has problem when the data pool is too small or too big. Thus, this method could be used for some specific recommendations, but it will be inefficient to use this method as the main one.