Cameron Sonido

Dr. Chad Williams

CS 398 – Independent Study

15 November 2016

The Security of Modern Robust Collaborative Filtering Systems

This study was an observation of how collaborative filtering systems, such as those used in dating sites, amazon, and spotify, can be defended from shilling (spam) attacks, in the manner that other courses do not allow. The goal was for me to both understand how collaborative filtering takes place, and also gain a working knowledge of the leading programming language for data miners, “R”. My plan was to obtain an open source database and use its tables to conduct isolated shilling attacks, as well as implementing the BADSA defense scheme.

The database I chose to use was the LibamSeTi database, released by Charles University in April of 2006. The data is pulled from a Czech dating site, where males and females can attribute a rating to one another on a scale of one to ten. Each user has only an integer id associated with them, and so the privacy of the users is upheld. For the sake of calculation speed, I classified ratings of 5 to 10 as a “Like”, and ratings of 0-4 as “Dislike”. While the database did contain female to male ratings, I focused only on male to female ratings, because I wanted to use the dataset with the most diverse ratings – Figures 1-3 shows that the female ratings of males were predominantly positive, and therefore it would be much harder to make confident recommendations.

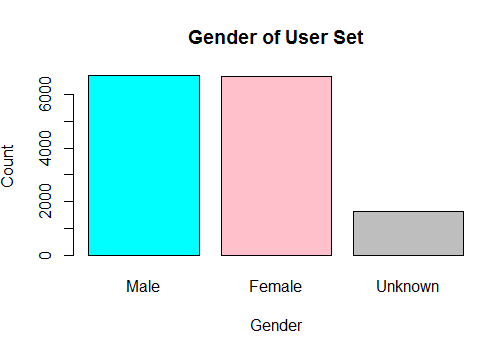


Figure 1

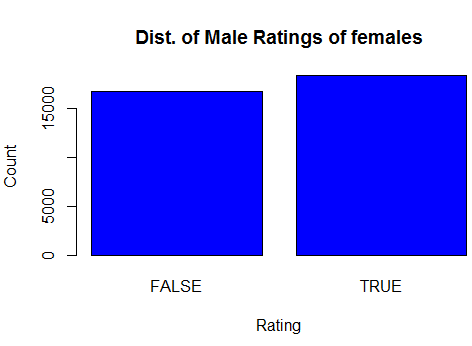


Figure 2

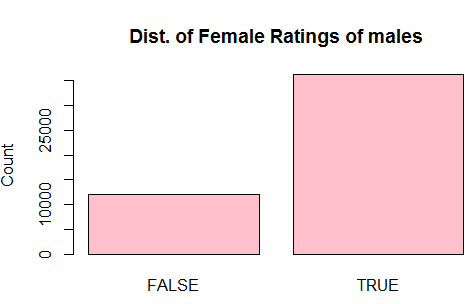


Figure 3

Collaborative filtering itself is simply the practice of recommending items to users based on what other users like – in this way, the sample users are “collaborating” to create a new recommendation for the new user. The backbone of collaborative filtering is the K-nearest-neighbors algorithm, denoted “knn”. The algorithm determines which users are “most similar” to a given user, with each such user being a “neighbor”, and with the number nearest neighbors pre-specified to some integer “k”. For example, if Bob, Carl, and Alice like the Chicago Cubs, and Bob and Carl like the Chicago Bears, then we can infer that Alice probably likes the Chicago Bears (assuming that those three are the only members of the sample set).

A shilling attack is the practice of injecting fake profiles into a recommendation system, such that a particular product is recommended more in the knn scheme. In the previous example, say we receive some data on 3 fake users, all who like the Chicago Cubs but dislike the Chicago Bears. If we were to run the knn scheme on all 6 users, our knn model would assume that Alice does not like the Chicago Bears.

The prevalence of shilling attacks has led to the development of robust collaborative filtering systems, where the “robustness” comes from discounting shilling attackers from the knn recommendation model. Using Rstudio and some of R’s default packages, I was able to compare the efficiency of various shilling attacks on both a normal collaborative filtering scheme and a robust collaborative filtering scheme.

I implemented three different shilling attacks via an R program. For each attack, one female profile is being “pushed”, or trying to be recommended more often, which will be denoted profile “n”. A random attack is done by injecting profiles that have random, meaningless ratings for all female users other than profile “n”, which has a positive rating. An average attack is conducted by calculating the average rating of each female user, across all female users, and injecting profiles that align with that rating, other than of course user “n”, which has a positive rating. The segment attack is conducted differently that it would in real life, since lack of user data prevented me from discerning true segments. Instead, the program conducts a k-means recommendation of 4 segments of the data, and tried to find the segment which has predominantly positive ratings of user “n”, which is the target segment. After this, the program finds the average rating for each female user across only users in the target segment, and injects profiles aligning with those averages, again, with the user “n” being recommended positively.

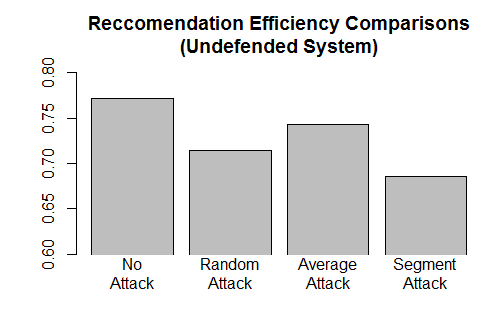


Figure 4

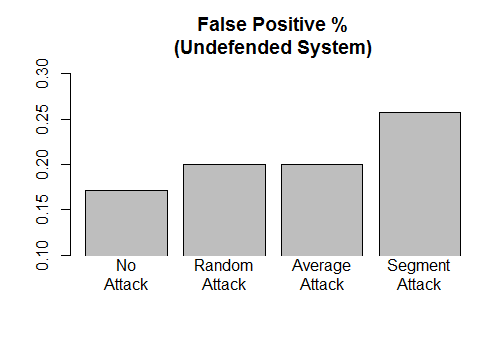


Figure 5

The results for figures 4 and 5 are conducted for attacks where 50 females and 163 males are used, with k equal to 3 for all knn models. The recommendation efficiency for each is how accurate each knn model is, calculated as the number of male users who were recommended the target user by knn, divided by the number of male users who actually “liked” the target female user n. The false positive rate for each is calculated as the number of users who were recommended user n who “disliked” user n, divided by the total number of male users.

As expected, the knn efficiency is significantly lower on attacked knn models. Additionally, the false positive rate is drastically higher than the rest for the segment attack, probably attributable to the ability the segment attack to create fake profiles that are closer to the target segment’s nearest neighbors than the other attacks.

The program then conducts the same volume of attacks on the same data, but then performs the Basic Algorithm for Determining Shilling Attackers (BADSA). The algorithm takes advantage of the face that shilling profiles are all very similar to each other, and so it classifies profiles with values higher in three metrics as shilling profiles. The standard deviation in ratings, degree of agreement with k neighbors, and number of prediction differences (formally, the number of predictions that differ if the given user were removed from the recommendation model). BADSA removes all accounts with high values in all three of these metrics, allowing knn to be conducted on a potentially clean user set.

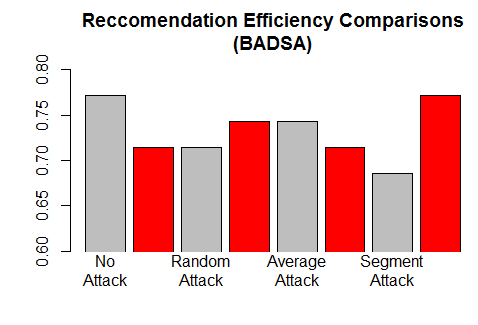


Figure 6

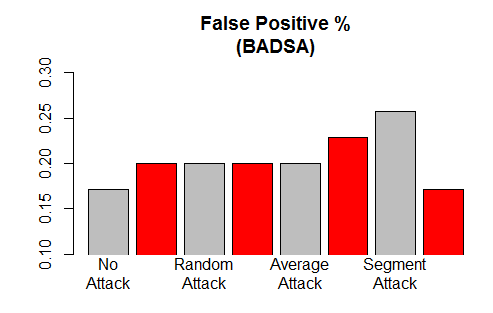


Figure 7

Figures 6-7 compares the knn efficiency of an undefended system (grey) with a system purged with BADSA (red). The data shows that BADSA has nearly eliminated the negative effects of a Segment attack, which was the biggest threat to an undefended system. The knn efficiency is nearly that of an un-attacked knn model. The false positive rate for segment attack has also dropped below that of an undefended system. The reason that the random attack’s false positive rate is that same throughout is probably because the random values of the ratings for non-targeted users are unlikely to be similar to each other, and therefore unlikely to be detected. Another perceived flaw would be how the performance of a system drops significantly if BADSA is implemented, but no attack has been conducted. For this reason, I would not recommend a system that is not likely to be attacked to use BADSA.

In conclusion, robust collaborative filtering is the best we can do to defend against shilling attacks, in a world where big data and its influence on programming as a whole grows by the day, and as attacks grow more robust, so too, should our defenses.