User Trust and Malicious Voting

Western Michigan University

Jonah Kubath

**Crowd Sourced Data**

Most of todays search engines rely on the use of off-loading work to those that use their product. Crowd sourcing data is when a collective group of people or users come together to combine ideas, opinions, or reviews to form a collective opinion on a “product”. The term “product” here can relate to businesses, items for sale, or honest / dishonest files in P2P file sharing. The users in the crowd can vote on the products. Their votes are then summed and published to the rest of the crowd so other users can make decisions based on other’s opinions. Crowd sourcing data has a few benefits. The first is that the search engine does not have to create new information on their own. With incentive for users, the users will add their own opinions on products to make more information available to others. The second is that users can be more confident in the opinion of a product when it is the aggregate of many users. Many users hinder biased or even malicious data to be the group’s opinion on a product.

As hinted to in the second benefit of crowd sourced data, there is an opportunity for malicious users to change or impact group’s opinion on a product. One of the best solutions to prevent bad information from malicious users is to assign trust scores to all the users in the system. This will put more trust in users who have appeared to vote honestly on other businesses and less trust in new users or users who appeared to vote dishonestly on other businesses. Voting honestly is determined when a user’s vote matching the aggregate of all the users on a specific product. As outline in “Voting Systems with Trust Mechanisms in Cyberspace: Vulnerabilities and Defenses” by Feng, Sun, Liu, Yang, and Dai, user trust scores can be attached efficiently with their algorithm called Reputation Trap (RepTrap).

**Description of RepTrap**

The overall summary of RepTrap is a coordinated attack by the malicious users. The malicious users use the review information to determine a set of businesses that were also voted on by the users who voted on the target business called the correlated set. They will then collectively vote against the crowd on the correlated business set to reduce the system’s trust in other users while simultaneously increasing the system’s trust in the malicious users. This algorithm will reduce the total number of votes needed to attack the target business.

The first step in RepTrap is to calculate the correlated set and uncorrelated set of businesses. The correlated set is defined by any business “A” in which a user has voted on the target business and business “A”. The uncorrelated set is any business not in the correlated set.

The second step is to attempt to trap the target business. Trapping a business will include having all the malicious users vote on the target business. In application this can be changing a business’ review from good to bad, product stars from five to one, or an honest file to a dishonest file in P2P file sharing. If the malicious users do not have enough trust to trap the target business, RepTrap moves on to the next step.

The third step is to trap businesses in the correlated set. Changing the opinion on a correlated business will decrease the system’s trust in the honest users who voted on that business while increases the system’s trust in the malicious users. The second step would then be attempted again after each business is trapped.

When all the businesses in the correlated set have been trapped, the malicious users will then attempt to trap uncorrelated businesses. While this will no longer decrease the trust of the users who voted on the target business, it will continue to increase the system’s trust in the malicious users. As before, the second step would be attempted after each business is trapped.

If the third and fourth steps fail to increase the malicious user’s trust enough to trap the target business, the malicious users will revert to voting honestly on businesses. They will start with businesses in the uncorrelated set as it will have not impact on the users who voted on the target business and finish with the correlated set if needed. Once the process of voting honestly on a business is finished, trapping of the target business will be attempted in step two. If this fails, the number of malicious users was not great enough to attack the target business in the system.

**Application of user trust and malicious voting**

The data used in this project was publicly available data from yelp.com. The dataset includes business information and reviews from users. The yelp dataset was used as a real-world implementation of the RepTrap attack method. Users are assigned trust scores based on the number of aggregate crowd votes that match the vote given by the user. The number of matching votes will be called “good” and the number of votes against the crowd will be “bad”. The total user trust is: (good + 1) / (good + bad + 2). This defaults all new users to a trust score of 0.50 and an upper limit of less than 1. In the yelp dataset, the users can give star ratings from one to five. To make computation easier, these values are converted to binary as anything below three is a bad review and three and above are good reviews. The final business stars are calculated by dividing the summation of user’s trust who voted a business good divided by the summation of all user’s trust who voted on the given business.

**Optimizations to RepTrap Algorithm**

During the study and implementation of the RepTrap algorithm, an optimization was found to decrease the total number of votes required by the malicious users. The general RepTrap algorithm is to trap all the correlated businesses, then trap uncorrelated, and finally vote honestly on the uncorrelated and correlated sets. The optimization that was implemented was to calculate the theoretical number of votes required to raise the total malicious trust score to a high enough value that allows the target business to be trapped. This allows the RepTrap algorithm to break from trapping businesses in the correlated or uncorrelated early, when it would be beneficial (require fewer malicious votes) to simply vote honestly on a few uncorrelated businesses. The results from this optimization will be discussed in the next section.

**Results from RepTrap and Optimization**