Building a Task Blacklist for Online Social Platforms

Trang Ha, Quyen Hoang, Kyumin Lee
Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA, USA
{tdha, qthoang, kmlee}@wpi.edu

***Abstract*—Recently, the use of crowdsourcing platforms (e.g., Amazon Mechanical Turk) has boomed because of their flexible and cost-effective nature, which benefits both requestors and workers. However, some requestors misused power of the crowdsourcing platforms by creating malicious tasks, which targeted manipulating search results, leaving fake reviews, etc. Crowdsourced manipulation reduces the quality of online social media, and threatens the social values and security of the cyberspace as a whole. To help solve this problem, we build a classification model which filters out malicious campaigns from a large number of campaigns crawled from several popular crowdsourcing platforms. We then build a task blacklist web service, which provides users with a keyword-based search so that they can understand, moderate and eliminate potential malicious campaigns from the Web.**

***Index Terms*—crowdsourcing, crowdsourced manipulation, task blacklist, classification**

I. Introduction

Crowdsourcing platforms such as Amazon Mechanical Turk (MTurk)[1] and Figure Eight[2] have gained considerable favor among companies and users, as they have made the process of appealing to the mass for completion of micro-tasks much easier. The crowdsourcing model provides requesters with a low-cost, constantly available workforce, mitigating the need to hire fixed-time employees to perform simple tasks. Additionally, while AI has been developed to solve numerous problems, it is not always feasible to make use of automation instead of human labor. For instance, to perform a classification task, initial manual labeling is required for training machine learning-based classification models. On the other side, workers can complete micro-tasks and earn money mostly regardless of location and time. The more quickly they finish the tasks, the more money they can make.

While crowdsourcing seems to be an ideal model for both requesters and workers, crowdsourcing platforms face a risk of being exploited by malicious requesters, who create malicious tasks. Choi et al. [1] found that malicious tasks aimed at information manipulation by manipulating search engines, writing fake reviews or creating manipulated accounts, etc. In the work, they found that malicious campaigns have several notable characteristics: shorter estimated completion time along with higher rewards rate, which makes them more appealing to workers when selecting tasks to complete because some workers may not care about ethical issues or consequences.

So far, there exists no research or product that compiles the findings of malicious campaigns into a comprehensive list where affected parties can utilize to moderate the crowdsourcing environment. it has given us the motivation to build classification models inspired by the findings from [1]'s research, and detect additional malicious campaigns within over 446K campaigns crawled from several crowdsourcing sites. In the end, we build a task blacklist, a web service, which provides a keyword-based search service so that users can query for malicious campaigns based on the targeted site, task description or requesters' information. Our task blacklist service will be a useful tool for potential victims such as social media sites, search engine companies and e-commerce sites, to understand malicious attempts and which specific item or product was targeted, as well as for workers to be aware of this service, hoping them to conduct legitimate tasks instead of malicious tasks.

Our contributions are as follows:

- We collected a large dataset consisting of 446K campaigns collected from four crowdsourcing platforms.
- We developed a malicious campaign classifier and achieved higher accuracy than the SVM-based model that used in the prior work. Then, we applied the model to the large dataset to identify new malicious campaigns.
- To support users and potential victims targeted by the malicious campaigns, we built a task blacklist web service, a keyword-based search service where users can retrieve information of matched malicious campaigns.

[1]https://www.mturk.com/
[2]https://www.figure-eight.com/

ASONAM '19, August 27-30, 2019, Vancouver, BC, Canada
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6868-1/19/08...$15.00
http://doi.org/10.1145/3341161.3343705

II. Related Work

A number of researchers have noticed that the rising problem of crowdsourced manipulation in the past few years and have reported valuable findings related to the malicious campaigns on the Web. In 2012, Wang et al. [2] took the first step in tackling the problem of *sybils* (fake accounts) in social networking websites such as Facebook by developing a scalable crowdsourced sybil detection system. On another note, Fayazi et al. [3] showed that malicious requestors targeted popular search engines (e.g., Google, Bing), social media sites (e.g., Facebook, Pinterest), and online e-commerce sites (e.g., Amazon) by creating malicious tasks and hiring workers from crowdsourcing platforms. Choi et al. [1] proposed a novel approach to define, classify and detect malicious campaigns that exist on several popular crowdsourcing platforms. They first analyzed the characteristics of malicious campaigns as opposed to legitimate campaigns, and then made use of these characteristics (or features) to build a classification model to identify malicious campaigns with a high accuracy. This work is the most relevant to our work, and we compare our classification models with their SVM-based model in the experimental section.

Recently, Su et al. [4] looked into a specific crowdturfing problem called *Add to Favorites* that occurs within the realm of online shopping. According to the paper, adding to favorites is a popular function in online shopping sites, which helps users make a record of potentially interesting items for future purchase. Malicious requesters exploit this by putting up tasks for workers to add their items to favorite. In this way, their products will get higher positions in the search results, which will make them more noticeable to shoppers, potentially increasing their number of sales. The authors proposed a factor graph based model to identify this kind of malicious crowdsourcing behavior.

III. Datasets and Methodology

In this section, we describe our datasets and proposed framework. We use two datasets called a small dataset and a large dataset. The small dataset consists of 23,220 crowdsourcing campaigns with manually labeled information (i.e., which campaign is malicious or legitimate), including 3,356,153 tasks[3]. This dataset was collected from MTurk, Microworkers, Rapidworkers, and Shorttask by using a crawler that we developed for the prior work [1]. The crawler collected 23,220 campaigns, including detailed campaign descriptions within a period of three months between November 2014 and January 2015. Each campaign in the collected dataset was manually labeled to either malicious or legitimate based on its description. Out of the 23,220 campaigns, 5,010 campaigns were labeled as malicious and the remaining ones were labeled as legitimate. In the same manner, we collected a large dataset, consisting of 446,167 crowdsourcing campaigns from the four crowdsourcing platforms in a 20-month period from July 2015 to February 2017. The large dataset is unlabeled. Manually labeling the dataset requires a lot of efforts and time-consuming job. This problem motivated us to study how to build a classifier based on the labeled small dataset and predict a class of each unlabeled campaign in the large dataset.

[3]A campaign contains multiple tasks, and each task is assigned to one worker.

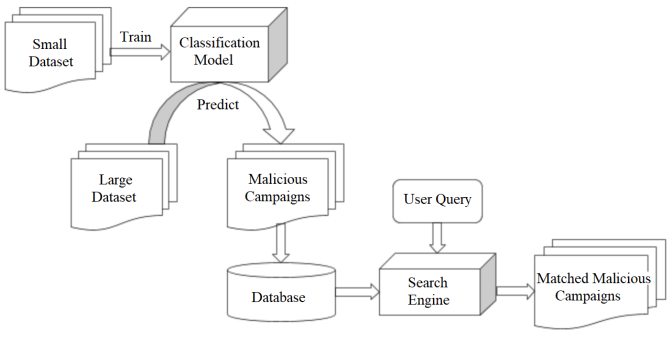


Fig. 1. A high level workflow of our proposed framework.

Figure 1 depicts a high level workflow of our proposed framework. As we briefly mentioned earlier, we use the small dataset to develop a classification model. Then, we apply the model to identify malicious campaigns in the large dataset, and save these campaigns into the database. We then index the malicious campaigns and build a keyword-based search engine. Finally, we deploy the web-based search engine so that users can understand what kind of malicious campaigns have been created, which item or product has been targeted by malicious requestors.

IV. Building Classification Models

A. Features

Following our prior work [1], we used the following features to build malicious campaign classifier: reward, number of tasks, estimated time to complete (ETC), hourly wage, number of URLs in task instruction, number of URLs in task instruction, number of words in task instruction, number of words in a task title, number of words in task instruction, and text features extracted from task title and task instruction.

To extract text features, we went through the following steps:

- Combine task title and task instruction, remove stopwords and apply stemming
- Build a TF-IDF vectorizer with the sklearn library that uses the provided text features as its vocabulary and automatically extracts unigram, bigram, trigram features from the text
- Generate TF-IDF vectors for each campaign based on its processed title and instruction text with the vectorizer

After the TF-IDF vectors were generated, we combined them with all other proposed features to get the final vectors for our model. The entire process of feature extraction can be summarized in Fig. 2.

When we extract text features, a way to combine task title and task instruction may lead slightly different bigram

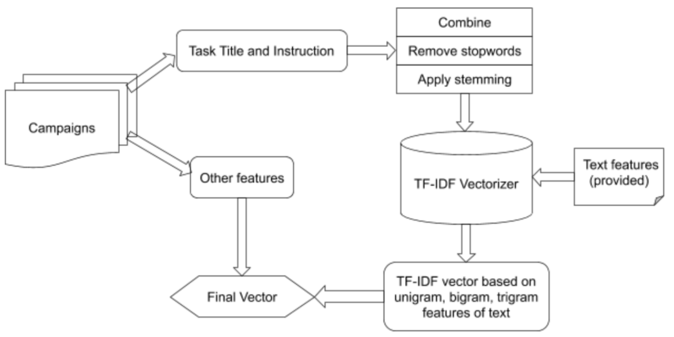


Fig. 2. Feature extraction workflow.

and trigram features. Therefore, we tried two different text concatenation approaches:

- Concatenate the task title with the task detail
- Concatenate the task detail with the task title

B. *Classification in the Small Dataset*

Choosing which classification algorithm to use is an important decision, so we tried various classification algorithms to see which one produces the best result in this malicious campaign detection domain. In the previous work, [1] found that Support Vector Machine (SVM) produced the best performance. Therefore, we selected SVM as one of our classification algorithms, and used its performance as a baseline. We also chose CART (a simple decision tree algorithm), Random Forest, Gradient Boosting Tree, and Xgboost to make the Decision Tree algorithm more powerful.

In order to pick the best performing model, we compared each model's false positive rate (FPR), false negative rate (FNR), accuracy, and area under the receiver operating characteristic curve (AUC) by using the labeled small dataset (again, containing 23,220 campaigns). AUC shows how well a model can discriminate between positive and negative classes. AUC is useful for evaluating binary classification, especially with high bias data (one class is much more common than the other). Since the number of malicious campaigns is significantly less than the number of legitimate campaigns in both of our datasets, this metric is extremely valuable for our model evaluation. We randomly split the labeled small dataset into training and test sets, which consist of 17,415 campaigns and 5,805 campaigns, respectively.

Overall, as shown in Table I, the Random Forest model based on the second text concatenation approach (concatenating the task detail with the task title) outperformed the other models including SVM (the baseline used in the prior work), achieving 0.016 FPR, 0.003 FNR, 0.994 Accuracy and 0.990 AUC. Based on the result, we chose the Random Forest model to identify new malicious campaigns in the large dataset. The Random Forest model identified 51,605 new malicious campaigns from the large dataset.

TABLE I
CLASSIFICATION RESULTS IN THE SMALL DATASET.

Model	FPR	FNR	Accuracy	AUC
SVM	0.104	0.011	0.969	0.943
CART	0.021	0.006	0.991	0.987
Gradient Boosting	0.025	0.004	0.992	0.986
Xgboost	0.025	**0.003**	0.992	0.986
Random Forest	**0.016**	**0.003**	**0.994**	**0.990**

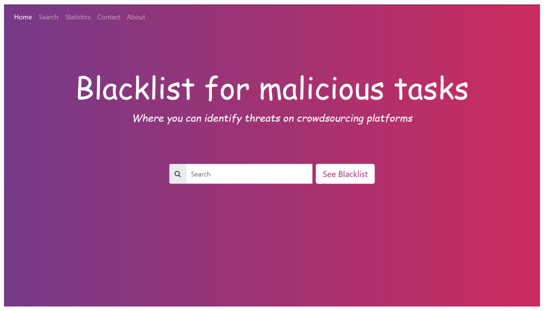

Home
Search
Statistics
Contact
About
Blacklist for malicious tasks
Where you can identify threats on crowdsourcing platforms
Search
See Blacklist

Fig. 3. Landing page of our task blacklist service.

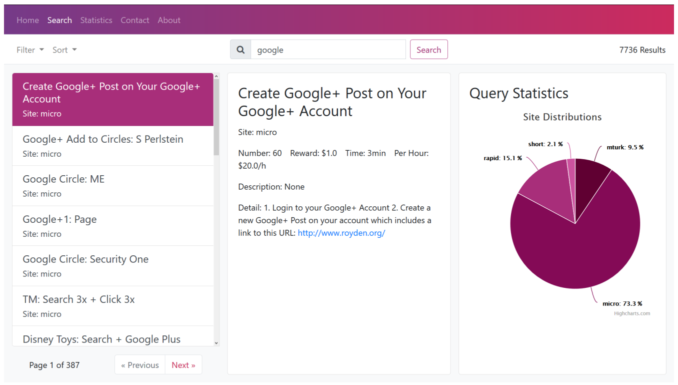
V. CREATING TASK BLACKLIST WEB SERVICE

In this section, we describe front-end and back-tend system of our task blacklist web service.

A. *Front-end Design*

Our first task in creating the front-end UI for the web application was to come up with the overall structural design. In order to create an efficient design for displaying a large number of malicious campaigns, we decided to create a landing page which includes a search bar for users to put in their search criteria (e.g., task name, name of targeted sites, name of task requesters). From this, users will be redirected to the search result page that consists of malicious tasks that satisfy their search criteria. The landing page (see Fig. 3) only contains a title for the project, a search bar and a navigation bar for users to get to other parts of the website. This includes links to the *Search*, *Statistics*, *Contact* and *About* page. The *statistics* page displays graphs and charts to inform users about the percentage of malicious tasks on crowdsourcing websites, the breakdown of malicious task rate in each platform, and typical pay rate for one malicious campaign, etc. *About* page shows a short description of the project, and *Contact* page provides contact information of this project's team members.

When we designed a search result page, we got inspiration from popular sites such as Google Careers, Monster and Glassdoor. There are three main panels: the left-hand panel contains a list of malicious tasks (see Fig. 4). Only 20 campaigns are shown at the same time in the list to avoid crowding the page and to improve page loading time. When a user chooses an item from this list, the details of the malicious task would appear on the central panel. Concurrently, the right-hand panel will display some statistics related to this particular task.



Create Google+ Post on Your Google+ Account
Site: micro

Google+ Add to Circles: S Perlstein
Site: micro

Google Circle: ME
Site: micro

Google+1: Page
Site: micro

Google Circle: Security One
Site: micro

TM: Search 3x + Click 3x
Site: micro

Disney Toys: Search + Google Plus

Page 1 of 387

Create Google+ Post on Your Google+ Account

Site: micro

Number: 60 Reward: $1.0 Time: 3min Per Hour: $20.0/h

Description: None

Detail: 1. Login to your Google+ Account 2. Create a new Google+ Post on your account which includes a link to this URL: http://www.royden.org/

Fig. 4. An example of a search result page.

B. Back-end Components

a) Back-end framework: We chose Django as our back-end framework. Django is a popular Python-based back-end web application framework which follows the model-view-template (MVT) architectural pattern. Some notable sites that use Django include Disqus, Instagram, and Mozilla. Django includes an object-relational mapper (or ORM) that arbitrates the data models (which are defined as Python classes) and the database (the "Model"). The "View" component is the Python callback function for a particular URL, and a template system for the user interface (the "Template"). Django is well-known for its emphasis on making sure that developers can avoid most common mistakes related to security. These common mistakes include SQL injection, cross-site request forgery (CSRF), clickjacking and cross-site scripting (XSS). By using a strong user authentication system, Django significantly reduces security risks. Furthermore, by employing a separation-of-concerns method, Django projects are highly scalable. This means that developers can add hardware, middleware or caching servers at any point and still make sure the project is not corrupted.

b) Database management: Since the data that we used for training and testing were stored in SQL tables, we used MySQL for database management.

c) Search engine: One of the main features of our web service is a search function. This is where users put in a search criterion and it will be used to filter out relevant malicious tasks. We used Django Haystack library[4] as our search engine for its numerous advantages. First of all, as it is a search engine supported by Django, our choice of back-end framework, the installation and usage of the search engine is much easier and faster than any other searching modular. Secondly, Django Haystack is a reusable app and relies only on its own code. This makes it work nicely with both first-party and third-party apps without requiring us to modify existing code.

Finally, Django Haystack is extremely flexible and supports some of the most popular search back-ends (e.g., Elasticsearch[5] Solr, Whoosh). The back-end is also pluggable, meaning the developer can change the search back-end with minimal code changes. Among the popular search back-ends, we chose Whoosh, which is much simpler but still works well. Our web service is running on a CentOS 7 server. The address for our website is: http://blacklist.wpi.edu:8000/bl/.

[4]https://django-haystack.readthedocs.io/en/master/tutorial.html
[5]https://www.elastic.co/products/elasticsearch

VI. Conclusion and Future Work

In this project, we have developed malicious campaign classifiers based on a labeled 23,220-campaigns dataset. Random Forest model achieved the best performance among five models, and produced 0.994 accuracy, and 0.016 FPR. Then this model was applied to a large unlabeled dataset consisting of 446K campaigns, and found over 51K new malicious campaigns. These malicious campaigns were indexed and used for our task blacklist web service.

We created the task blacklist web service, where users can query for malicious campaigns based on the targeted site, task description or requesters' information. This web service uses Django as the back-end framework, which promotes separation of concerns, security and scalability. The database management system of our choice was MySQL for ease of use and popularity. Django Haystack was the search modular used for the searching functionality for its compatibility with Django, our choice of back-end framework. This web service was deployed to a server machine.

In the future, we are interested in building and running a real-time data collection and index updating system so that our task blacklist provide users with up-to-date malicious campaign information. We are also interested in expanding the number of crowdsourcing platforms from four to many so that our task blacklist service provides users with more diverse malicious campaign information. The current search engine back-end is Whoosh. While it is sufficient for now, this back-end is not optimal for scaling and performing more complicated tasks such as faceting. We plan to replace this back-end with a more powerful one such as Elasticsearch and Solr.

Acknowledgment

This work was supported in part by NSF grant CNS-1755536, and WPI Academic & Research Computing. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsors.

References

[1] H. Choi, K. Lee, and S. Webb, "Detecting malicious campaigns in crowdsourcing platforms," in *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2016.
[2] G. Wang, M. Mohanlal, C. Wilson, X. Wang, M. Metzger, H. Zheng, and B. Zhao, "Social turing tests: Crowdsourcing sybil detection," *arXiv preprint arXiv:1205.3856*, 2012.
[3] A. Fayazi, K. Lee, J. Caverlee, and A. Squicciarini, "Uncovering crowdsourced manipulation of online reviews," in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015.
[4] N. Su, Y. Liu, Z. Li, Y. Liu, M. Zhang, and S. Ma, "Detecting crowdturfing add to favorites activities in online shopping," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, 2018.