**Discovering Networks of Fake Product Reviews**

**Abstract**

With the rise of e-commerce, consumers have gained unprecedented choice in choosing products sourced from all over the world. To sort through these vendors, consumers rely increasingly on product reviews (eWoM) to make informed purchases. As a result, incentive for vendors to engage in the purchase of fake reviews has proportionally skyrocketed in order to gain consumer trust. In this study, I explored if it was possible to crawl a network of vendors/reviewers engaged in the fraudulent reviews market, and if so, how far could one trace this network across Amazon.com. To realize this system in a dynamic fashion, two alternating processes were created: the spam review identifier, and the web crawling service. This project is an ongoing work, as Amazon has made significant strides to obfuscate/remove evidence of fake reviews since the start of this project and distributors of fake reviews are adopting more sophisticated tactics. Nevertheless, I will outline a framework for a dynamic fake-spotting machine to be used in an unsupervised fashion on Amazon.com (with perfect information).

1. INTRODUCTION

The rising influence of e-commerce sites such as Amazon.com and Jet.com has been felt across nearly every industry, especially in America. Almost 40% of all money spent on the internet is spent on Amazon.com and nearly 60% of US households have Amazon Prime subscriptions. The presence of disingenuous reviews on products has grown significantly over the past 3 years [ <https://www.forbes.com/sites/emmawoollacott/2017/09/09/exclusive-amazons-fake-review-problem-is-now-worse-than-ever/#4f659eed7c0f> ]. Amazon has been particularly vulnerable to this type of deception, as account-holders can write reviews for products they have not purchased (on Amazon). We will see later on that clusters of positive “unverified reviews” on products were (up until recently) a red-flag to their collective authenticity. In general, research on spam review detection either focuses on linguistic features of reviews or behavioral patterns of reviewers. SPIDER uses features about each ASIN (Amazon Standard Identification Number) as well, such as the quantity of a product’s unverified reviewers or burstiness of its reviews. To my knowledge, there is no research that attempts to perform spam detection + data-collection prioritization dynamically; that is, starting with a (small) seed dataset and branching outwards in the direction of likely spam instead of working with a large static dataset. Because of this, SPIDER has to address unique problems not faced by previous research: how can we assess fraudulence using an incomplete dataset? How do we determine priorities of scraping items?

My research also aims to shed light on the following questions: How far can one traverse a single graph of fraudulent reviewers and businesses before it is exhausted? How do we convert classifiers meant for static assessment of complete datasets into classifiers meant to steer a growing dataset?

SPIDER was devised to be a tool that can systematically discover and identify fake reviews on Amazon.com. It was born out of frustration with the ever-pervasive presence of falsified reviews and Amazon’s negligence towards solving the issue. I imagined SPIDER to be akin to a robotic-vacuum for spam reviews on the internet. The structure of this application boils down to crawling/data collection and review fraud detection. SPIDER is meant to exploit the graphical relationship of fake reviews found by the detection component to direct the web-crawling service towards products or users that have a high-probability of contamination by fraudulent activity. I originally believed my project to be similar to the research of NetSpam, the fake review classifier introduced by Shehnepoor *et. al* [2], however much of their framework had to be overhauled for SPIDER as it was dependent on complete information (a full dataset at the start).

1. IMPLEMENTATION

As mentioned, two main ingredients of SPIDER were implemented. First, a web scraper to collect data such as product reviews, reviewer profiles, and vendor profiles.

Product review data includes: review text, date posted, verified/not, number of helpful votes, profile url, profile id, review stars, product asin. Reviewer profile data includes: products reviewed, review text snippets, number of reviews, name, review dates, and review stars. As it turns out, collection of reviewer profile data became SPIDER’s largest hurdle; Since embarking on the project, review fraudsters have begun to take advantage of Amazon’s profile privacy settings. Almost every single fraudulent-seeming profile scraped had set all of their public profile settings to maximum privacy. This meant that the collection of the critical network data and all of the information it would provide (list of next products to scrape). Additionally, Amazon started to take action in removing

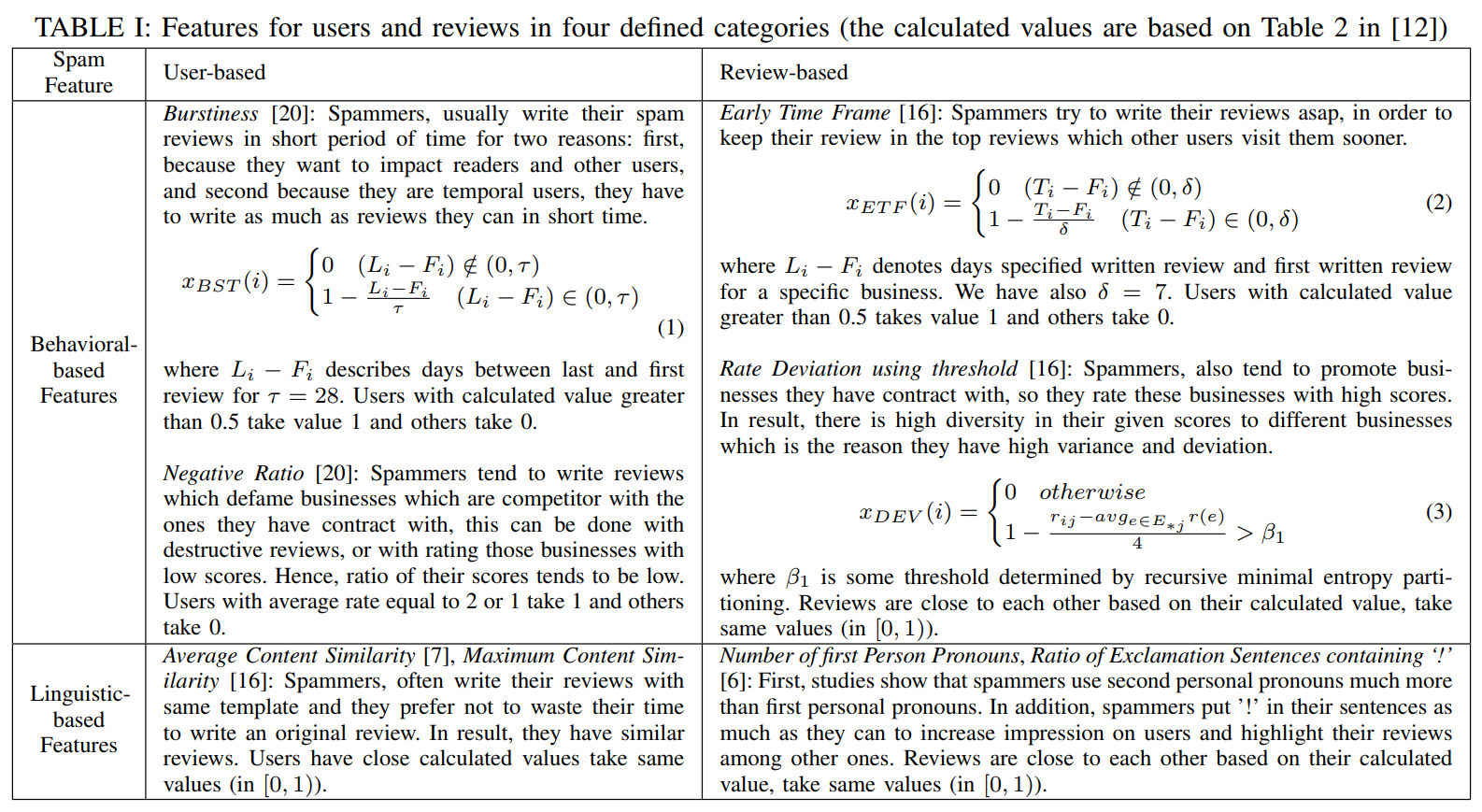
I have already implemented scrapers for each of these entities, and while they have not erred thus far, it is possible Amazon will serve the program robot-checking pages if it detects regular requests in high volumes. If data collection becomes a hurdle because of robot-checking, Professor Muresan pointed me to a set of reviews with authenticity labels from Lie *et al.* [1] that would work instead. Since the scraper integration is the more novel aspect of this project, pre-collected data will only be used as a last resort. The web scraper is implemented in python using the selenium library because it can operate a chrome browser, allowing for random page behavior to decrease the likelihood of robot checks.

The web scraper will work in tandem with a fake review classifier that takes as input the most recent network of review/profile data, updates weights and labels within the model, then outputs a list of entities (reviews/profiles) to scrape prioritized by their likelihood of being spam. This model will be a variant of the NetSpam model, introduced by Shehnepoor *et. al* [2], modified to synthesize current beliefs of entity fraudulency into a list of unscraped entities sorted by likelihood of fabrication involvement.

1. LITERATURE REVIEW

As e-commerce grows into an increasingly dominant force in consumer retail, so does the attention given to product reviews as a source of electronic Word of Mouth. Despite Amazon’s review policies, many businesses have been incentivized enough to take the risk and purchase phony promotional reviews. Recently researchers have developed methods that exploit graphical relationships between review authors and the reviewed products to identify *groups* of spammers and shady product vendors. As mentioned earlier, to the best of my knowledge there has been no published research that aims to serve the same purpose as SPIDER: to work outwards from a seed dataset and dynamically discover a *single* network of fake reviewers.

I took the most inspiration from the paper NetSpam because the way it structures and addresses the problem of fake review detection seemed compatible with the goals of my project. I originally thought there were similarities in our applications because NetSpam structures the problem as a heterogeneous information network, where nodes represent reviews, users, and features derived from them while edges represent ownership and relation between nodes. While this method worked well for their data sources, it actually does not scale well to smaller (seed) datasets. I originally thought this problem structure to be complementary to the data scraping component because each classification round over users or product reviews would generate a list of probabilities to prioritize the next round of data collection. This actually proves to be incompatible with the goals of SPIDER because NetSpam is designed to be run across a complete dataset with full network information of both profiles and products. SPIDER can not use this framework because it is persistently in a state of incomplete information, which manifests as either products without review data or reviewers without profile data. Because of this, a novel classification method and narrowing of scope had to occur. Still, features from Shehnepoor *et. al* [2] proved to be useful in the classification step of SPIDER.



Problems:

There were many problems that grew from the original inception of SPIDER. The two problematic components of the project were Amazon’s growing efforts removing fake reviews and fraudulent reviewers using new methods to maneuver fraud detection. This project was born of frustration towards Amazon’s complicity with their fake review problem; it seems as if they have finally addressed my many feedback reports because nearly all of the fraudulent reviewers from my seed dataset have been removed from the site. While this was a victory for economic consumer information, it created a major hurdle for SPIDER’s implementation; there was no longer a seed dataset with fraudulent profiles to work backward from. A workaround I had in store was to find another product with identifiably-fake reviews (for this project, I considered a review to be 100% fake if it evidenced being a copy another product’s review). Luckily, I found a few other products with identifiably-fake reviews with the same brand as my original seed dataset. This is where the new methods of spam-detection-evasion threw another hitch in the implementation. Up until recently, one could view the profile of a reviewer on Amazon and see all of their historical review activity; It has now become standard among reviewers (likely *fraudulent* reviewers) to hide as much of the public profile in an effort to evade fake-review detection. As mentioned earlier, this means network information of reviewers and products can no longer be collected directly from a profile, instead it can only be happened upon by co-occurrence of reviewers on products. In effect, this is the largest hurdle the project faced because the network information could no longer be accessed at-will, which stunted the project immensely. I did not use the data from Lie *et al.* [1] in the end because I did not have enough time to transform their data to the format of the scraped data and also build a simulated scraping service. Additionally, their data was difficult to use on a short time-span because it is unlabeled; SPIDER relies on a seed set with at least 1 fake reviewer.

Unlike previous methods that ignore network features of spam reviews and rely solely on patterns that emerge from review-linguistic features in large datasets (BoW + SVM, etc), NetSpam needs minimal data with a seed fake-review and its local network relationships to estimate unknown labels. Shehnepoor *et. al* [2] proposed both a semi-supervised and unsupervised version of NetSpam, demonstrating state-of-the-art results (AP, AUC) across both modalities. Furthermore,NetSpam has four categories of features: Review-Behavioral, Review-Linguistic, User-Behavioral and User-Linguistic. Review-Behavioral features include early time frame and threshold rating deviation of review. Review-Linguistic refers to features extracted from the review text, such as the ratio of first-person pronouns and the ratio off sentences with exclamation points. I intend to experiment with the review-linguistic features the most, particularly by adding a “similarity search” of the review text across all amazon reviews using exact-match google queries. The reason why I am adding this feature is because many fake reviews copy the exact text from a real review on a similar product; this is an extremely valuable feature because it provides with 100% certainty that the review is fraudulent. The User-Behavioral features include the burstiness of reviews and the negative/positive ratio given to different businesses. Lastly, the User-Linguistic features refer to the Average Content Similarity (ACS) and Maximum Content Similarity (MCS) of all of a user’s review text versus another’s. This category aims to identify similar verbiage based on the hypothesis that reviews could be made from a mad-lib style template. Details on feature calculation and parameters to be optimized can be found in the table below:

Table I from Shehnepoor et. al [2]

Any additional features must adhere to the user-based and review-based categorizations; complex network features between users or reviews are summarized by their metapaths in the graph. This aspect of the data representation is conducive to analysis that combines textual evidence of fraud with network clues. Metapaths represents heterogeneous relations between elements, and are a key step in the authors’ feature engineering work. They are mathematically derived from the features (above) in order to create a metric for level of fraudulency. A step-function is used to scale this metapath spam into a variable number of bins, *s*, such that lower values of *s* means bipolar metapath certainty values and higher values of *s* means more granular categorizations of spam levels. Two elements are connected in the network schema if their metapath values for a feature *l* are equal. In other words, users or reviews with similar enough levels of spamicity (scaled by *s*) create a link in the analysis schema. The NetSpam algorithm uses weights to calculate the importance of each metapath type (below) and optimizes their influence in the labeling accordingly.

1. WORK ALLOCATION

I am pursuing this project alone due to the murky ethics surrounding the possibility of it being sold as a data product by the research firm where I am currently employed.

REFERENCES

[1] Huayi Li, Zhiyuan Chen, Bing Liu, Xiaokai Wei, and Jidong Shao (2014). Spotting Fake Reviews via Collective Positive-Unlabeled Learning. Proceedings of IEEE International Conference on Data Mining (ICDM-2014, short paper), 2014.

[2] Saeedreza Shehnepoor, Mostafa Salehi, Reza Farahbakhsh, and Noel Crespi (2017). NetSpam: A Network-Based Spam Detection Framework for Reviews in Online Social Media. Proceedings of IEEE Transactions on Information Forensics and Security (TIFS), 2017.