Fake Review Detection System

Sravanthi Ponnana   
*Department of Computer Science*  
*University of Illinois at Urbana Champaign  
Champaign, IL, USA  
ponnana2.@illinois.edu*Xiang Gao  
*Department of Computer Science  
University of Illinois at Urbana Champaign  
Champaign, IL, USA*xiangg3 *@illinois.edu*

**Abstract: Online shopping is a growing business and reliance on the reviews for the businesses and shoppers has become critical than ever. When buying a product/service, shoppers are increasingly becoming reliant on reviews. Positive reviews can result in significant financial gains for businesses/organizations. This, unfortunately, is giving strong incentives for fake reviews. Fake reviews involve giving undeserved positive reviews to target businesses in order to promote them and/or giving false negative reviews to other entities with an intention to damage their reputation.**

**In this paper we propose a solution to identify the authenticity of the reviews by using a combination of natural language processing and unsupervised machine learning techniques like clustering. Although there have been attempts to detect the fake reviews, they have not included most important piece of information about the user / reviewer. Though working with labelled data i.e., supervised machine learning would help us in better determining the authenticity of a review, we do not have the logistics to obtain labeled data, which might involve manually writing fake reviews. The expectation is that unsupervised learning would help us in identifying any hidden trends, thereby in identifying potential fake reviews. These reviews can be flagged for further manual inspection.**

**We obtained more than 250,000 reviews and over 10000 users across 15 cities from the biggest Travel website Tripadvisor.com. After the initial cleanup we used PySpark running on AWS cluster to run clustering algorithm and identify different types of reviews. We found that including the user information increased the accuracy level of identifying fake reviews by around 11%. Detailed results are discussed in the section VII below.**

# Introduction

A 2015 survey by an internet retailer website found that product reviews can boost online sales by as much as 62% [1]. The survey also showed that the average order value increased by 3 percent when shoppers read reviews. If you're a retail business owner, you know that online reviews are important for your business [2]. We would have seen potential customers standing outside the doors of businesses, consulting their phone or tablet to make sure the products and services are highly rated.

But the sanity of the review system is being compromised these days. A few fake reviews can and have caused the businesses to be shut down or on the other hand made a fake product look great [3]. There have been claims that the websites themselves are promoting fake reviews to improve the ratings of certain sponsored businesses [4]

In this document we propose a way to help businesses and online customers by identifying the fake reviews made by people. This is particularly important as the percentage of users doing online shopping is increasing on a daily basis.

# Background

We used the data from the biggest online travel website Tripadvisor.com. There have been a few attempts to identify the authenticity of the reviews by Ott et al. [5] earlier but there are a few shortcomings of those which we will address in our paper.

Firstly, the review detection systems developed so far only look at the review text itself. We believe that information about the reviewer is as important as the review text itself. We would consider the reviewer information while building the model.

Secondly, the dataset considered by Ott et al. comprises of few hundred reviews which were manually entered and classified (Supervised Learning). We would extend this to 200,000 + reviews to build a much more robust and precise model.

Lastly the scale of our operation is much bigger what has been done previously and we will be using unsupervised learning techniques to analyze the reviews since labelled data is not available at such as scale.

# Proposal

In this section we propose our approach to improve the user experience while doing online shopping.

For our analysis we would scrap around 200 K to 250 K reviews from Tripadvisor.com. The review information we would obtain include: Review Date, Reviewer Name, Reviewer Text, Hotel Name, Number of Upvotes for this review etc., The biggest difference between the existing models & ours is that we would consider the reviewer information as well. The user information that we would obtain from the website include User Name, Total Number of helpful votes for the user, Total Number of reviews written by the user.

Once we have the data, we would perform clustering of similar reviews. The model will use a combination of NLP techniques and unsupervised techniques like clustering to detect fake reviews. The expectation is that clustering similar reviews would help us in identifying anomalous/suspicious reviews, and we would like to develop a set rules for fake reviews based on the clusters. We would also leverage the work of Ott et al. to compare the characteristics of anomalous reviews.

For Implementation, we will use PySpark since it has a good collection of machine learning libraries and it can access data stored in hdfs directly.

# Implementation

In this section we will talk about the implementation details of our proposal. It broadly consists of following parts :

* + - 1. Scraping the data.
      2. Clean up and enrichment of the data.
      3. Exploratory Analysis
      4. Building the clustering algorithm.

Scraping the data :

We implemented a python script which will scrape the TripAdvisor website for reviews. We started with following 15 cities: New York, Los Angeles, Chicago, Seattle, Las Vegas, Hawaii, Boston, Phoenix, Los Angeles, Houston, Miami, Washington DC, Orlando, San Francisco, San Diego. We go to the homepage of the city, scrape all the hotels and store it in a queue. Now we go through the items in the queue, go the hotel page and scrape all reviews of the hotel. We then go to the user home page and get all the user information as well. Added characteristics of user name such as number/presence or absence of space, special characters, capital letters etc. to the data set since we believe this would add value to the identification of the genuine users(vs computer generated users through the Trip advisor apis etc)

Each review we obtained looks like Figure 1. The explanation of each of the field is as follows:

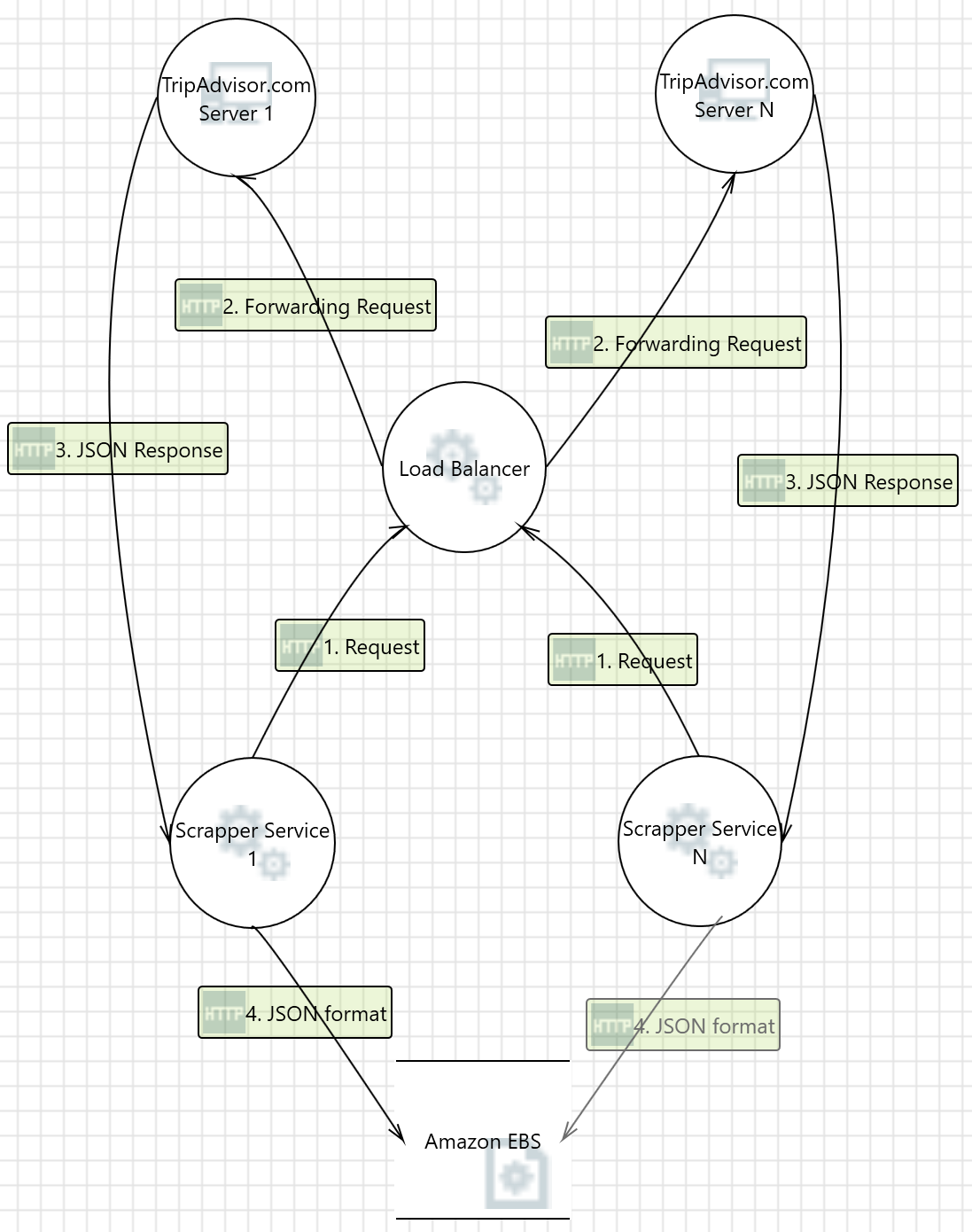
|  |  |
| --- | --- |
| Column Name | Explanation |
| Rating | Overall Rating of the hotel/business |
| Name | Name of the business |
| Url | Url of the hotel |
| Highlights | Feature of Hotel |
| HotelId | Id of the hotel |
| ReviewInfo | List of user reviews |
| UserName | Name of the reviewer |
| HelpfulVotes | Number of helpful votes received by this user |
| ReviewDate | Date of the review |
| ReviewText | Actual review content |
| Num\_Reviews | Total number of reviews by the user |
| RatingDict | Rating for different categories |
| Overall\_Rating | Overall rating given by the user |
| Helpful\_Reviews | Total Helpful reviews given by the user. |

 **Figure [1]. Screenshot of Sample Review Data**

The code to scrape the data is available in github[6] and is under active development. The scrapper can be started by updating the config.cfg file with the starting city’s URL and it would scrape all the hotels and reviewer info for that city. Use the command below to run the script.

*Python scrapper.py*

We ran it on multiple AWS instances in parallel so that we won’t breach the thresholds of the TripAdvisor website and prevent the scrapper from getting blocked. The high level system design is shown in the figure 2. We have N scrapper services, each one pointing to one city running on a separate EC2 instance. The load balancer will forward the request to one of the Trip advisor servers and we get the HTTP response from that server. We then parse the HTTP response and perform certain enrichment like joining with user data and store it on Amazon EBS for future reference.



**Figure [2]. High level System Design of the scrapper**

Each city is around 50 MB in size and has approximately 450 reviews. We started our analysis with dataset from a couple of cities (New York, Los Angeles) on a single machine, but now extended it to all the 15 cities mentioned in Section 1).

Data Cleanup and Enrichment:

We used python NLTK kit to preprocess the data. Removed stop words, performed stemming (reducing words to their stem i.e., writing to write etc.) Used TF-IDF (reflects how important each word is to document or a corpus) to represent the reviews.

Apart from the usual clean up of the unformatted data like cleaning up of spaces, exclamation etc., we de-normalized the data and joined the user information with the reviews as that approach is best suited with developing clustering algorithms.

Exploratory Analysis :

Initially we performed some exploratory analysis on the reviews for hotels in New York and Los Angeles. When we looked at number of reviews for each user, there were multiple reviews for some users.

While this looks normal at the outset, users with a space and a capital letter in their username have multiple reviews, and most of these are positive reviews. This looks a little suspicious and is a good candidate to be classified as fake.

Clustering Algorithm :

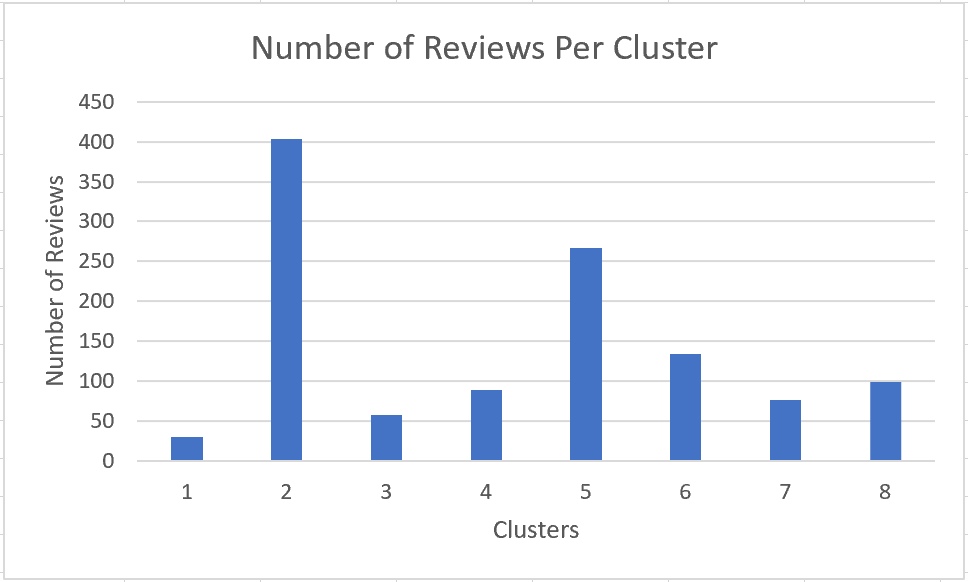
We built a K means clustering algorithm based on reviews from two cities as a prototype and then extended this to all the cities we have mentioned in the initial proposal section.

# Results & Observations

From the initial run of k-means, there are around 8 clusters (We dropped clusters with reviews less than 50, and also looked at elbow plot to determine the number of clusters).

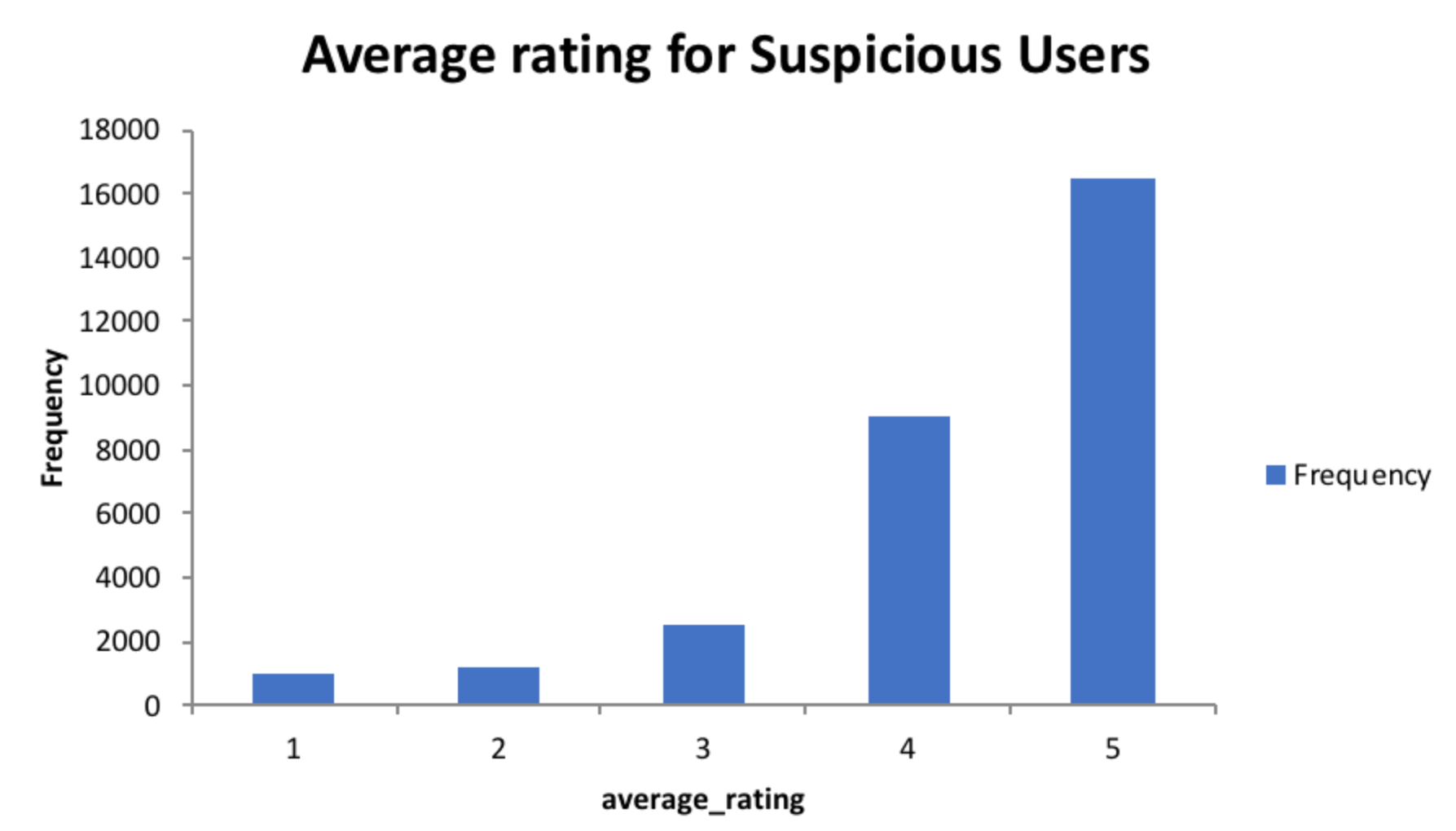
One particular cluster caught our attention with positive and very short reviews (less than or equal to 2 lines). We’ll have to check if this pattern extends to all the cities, if it does it’s probably a good pattern for them to be classified as fake.

Below is the sample chart for the number of reviews per each cluster and in the below distribution cluster 1 is the suspicious cluster we talked about previously and Cluster 5 is the one with the short and multiple reviews.

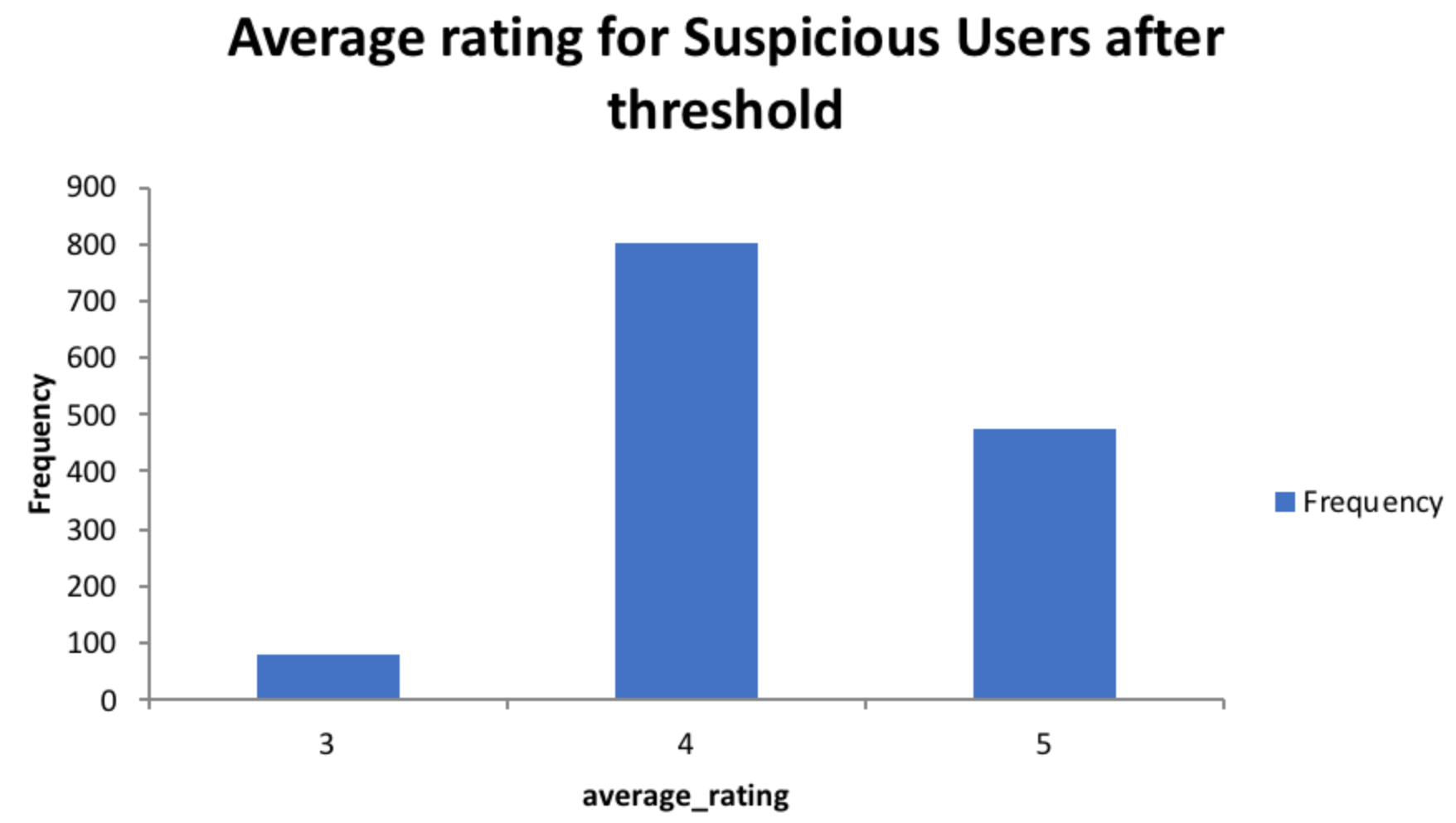


**Figure [3]. Reviews per Cluster for the clustering model**

The next interesting observation was that of average ratings for suspicious users our model tagged. Most of these suspicious users had a similar username pattern (Name followed by a letter, for example, Alex P). As you can see in the chart below, majority of the suspicious users rated a business 5 out of 5, which strengthens the theory that these reviewers might have been giving fake reviews. There have been stories online about people getting hired just for the purpose of giving fake reviews for users.[7] and the chart below tends to agree with that.



We looked at the number of reviews for these suspicious users and they had way more reviews than the average number of reviews per user. We considered a threshold and looked at the reviews of users above the threshold. They do not have any negative reviews, all reviews are positive. These reviews and users definitely need further probing.



# Implementation Performance

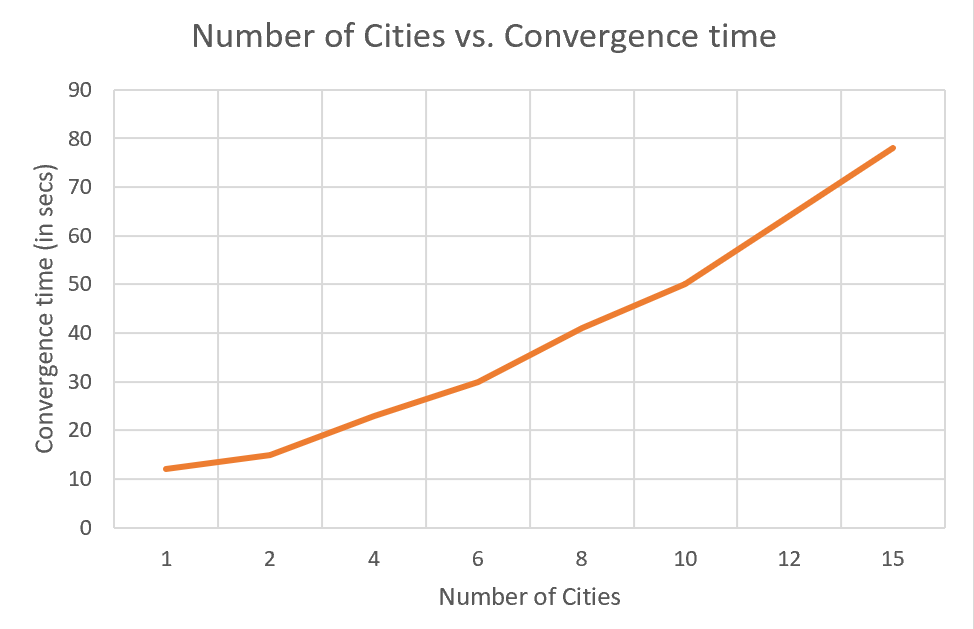
In this section we talk about the performance of our algorithm in terms of time taken for the convergence.

Firstly, we will discuss about the time taken for the K-Means algorithm to converge across different scenarios. We initially started with a prototype with two cities running on a single box and later extended it to 15 cities running on an AWS cluster with PySpark with and without redis caches and the run time is mentioned in the section below.

|  |  |  |
| --- | --- | --- |
| Scenario | Two Cities | All Cities |
| Local Machine | 30 sec | N/A |
| Aws Cluster / No Caching | 19 sec | 128 sec |
| AWS Cluster / Redis | 12 sec | 78 sec |

As expected its much slower when we run locally(mac) due to lower number of cpu’s. When we onboarded to AWS , the clustering algorithm’s convergence time reduced by 50%. We then introduced the redis cache in our AWS cluster and that further improved our convergence time by another 50%. Once we went through the protoype for two cities in the cluster along with the caching, our next step was to extend our algorithm to all the cities. It wasn’t feasible to perform this operation on a local machine as its too much data to handle. So we performed this on our AWS cluster directly and convergence times are mentioned in the table above.

The chart below shows the number of cities vs the convergence times for the k means clustering algorithm that we used on AWS Cluster along with Redis Cache installed.



# Comparison With Existing Models

Since, we use unsupervised learning, we won’t have test data set, and also given the scale of the data we do not have labelled data unlike the reference study mentioned[5] (manually labeled data after hiring a group of people to write fake reviews). Our model is intended to identify the characteristics for probable fake reviews that need to be screened further. The output would be a set of rules (identified from characteristics of suspicious clusters) to flag the reviews for screening.

Although an apples to apples comparison is not possible with the related work mentioned, we tried to prove that adding user information will improve the accuracy of the classification by doing the following: We started with our New York dataset one with the user information and one without. While we noticed ~425 reviews in the ‘suspicious’ cluster(discussed in the results & Observations section) without the user information, we saw only ~375 with the user information included. We did a manual check on the false positives and found that in fact reviews with high number of upvotes with a users who are ranked higher were tagged as fake as the user information wasn’t considered.

Not surprisingly this theory did hold for all the reviews across the 15 cities we included for this project. Although we had ~2760 reviews in the suspicious cluster when we ran the model with the user information included we saw around 2450 reviews. We randomly spot checked 5% of the reviews and majority of them confirmed with our proposal that without the

User information being included in that model will lead to false positives. The table below illustrates the improvements in the accuracy rates as a result of including the user information.

|  |  |  |  |
| --- | --- | --- | --- |
| Scenario | Without User Information | With User Information | Improvement  % |
| New York | 425 | 375 | 11.7 |
| All Cities | 2760 | 2450 | 11.2 |

**Figure [5]. Suspicious Review Cluster Size**

# Conclusion

Online shopping is a huge industry and its only growing bigger. And the reliance on reviews for trusting he businesses has become ever so important. While there have been attempts to classify the reviews as fake or not, they missed the most important information when classifying the reviews ie., the user information. In our solution, we scraped the reviews from the famous travel website Tripadvisor.com along with the user information. By performing the clustering algorithm on the reviews we found that there is an improvement of accuracy of ~11% when we include the user information when we manually tested on multiple smaller sample review sets. Hence we can conclude that including the reviewer information is as important as having the review comment itself in order to improve the accuracy of our prediction model.

##### References

1. *Product reviews boost revenue per online visit 62%*. [Online]. Available: https://www.digitalcommerce360.com/2015/01/22/product-reviews-boost-revenue-online-visit-62/. [Accessed: 2-Feb-2018].
2. *Why Positive reviews are so valuable to small businesses*. [Online]. Available: https://smallbiztrends.com/2017/04/importance-of-online-reviews.html/. [Accessed: 20-Feb-2018].
3. [*I created a fake business and bought it an amazing online reputation*](https://splinternews.com/i-created-a-fake-business-and-bought-it-an-amazing-onli-1793850918). [Online]. Available: <https://splinternews.com/i-created-a-fake-business-and-bought-it-an-amazing-onli-1793850918> [Accessed: 20- Feb- 2018].
4. [*Investigation reveals an entire industry is dedicated to generating bogus appraisal*](https://splinternews.com/i-created-a-fake-business-and-bought-it-an-amazing-onli-1793850918) [Online]. Available: http://www.dailymail.co.uk/news/article-3276617/Disturbing-proof-online-review-book-holiday-FAKE-Investigation-reveals-entire-industry-dedicated-generating-bogus-appraisals-cash.html [Accessed: 20- Feb- 2018].
5. [*Finding Deceptive Opinion Spam by Any Stretch of the Imagination*](https://splinternews.com/i-created-a-fake-business-and-bought-it-an-amazing-onli-1793850918)  [Online]. Available: http://myleott.com/op-spam.html [Accessed: 20- Feb- 2018]
6. [*Fake Review Detector - Github*](https://splinternews.com/i-created-a-fake-business-and-bought-it-an-amazing-onli-1793850918)  [Online]. Available: http://github.com/psravanthi/fake-reviews-detector [Accessed: 1- Apr- 2018]
7. [*Writing Fake Reviews for companies and getting paid for them*](https://splinternews.com/i-created-a-fake-business-and-bought-it-an-amazing-onli-1793850918)  [Online]. Available: hhttp://www.cracked.com/personal-experiences-2376-i-get-paid-to-write-fake-reviews-amazon.html [Accessed: 5-May- 2018]