AAKASH AGRAWAL, RYAN HAN, UMANG SEHGAL (GROUP3)

FAKE REVIEW ANALYSIS

Final Project Write Up

Introduction

When making purchasing decisions, customers increasingly rely on opinions posted on the Internet. Businesses therefore have an incentive to promote their own products or demote competitors' products by creating positive or negative spam reviews on platforms like Amazon.com. Although labeling individual fake reviews and reviewers is very hard, our work investigates to test a simple hypothesis which would help understand the relationship between product description and the review. We also note that the proposed technique departs from the traditional supervised learning approach for spam detection because of the inherent nature of our problem that seeks to find a simplistic conclusion on understanding the tendency of the type of reviews the fake review groups generate.

One Sentence Summary

In our experiment we aim to see if there exists a correlation between the product description and the 'fakeness' of a product review.

Building a Dataset

The Amazon Product Dataset was used for the research analysis presented in this subsection. The dataset was obtained from SNAP - Stanford Network Analysis Platform, a general purpose network analysis and graph mining library [1].

This dataset contains reviews of products – which includes text, ratings and helpfulness votes, links, and also provides information about the product metadata for 142.8 million reviews spanning May 1996 – July 2014. The reviews ranged from wide variety of products, such as books, movies, home and kitchen, video games, health and personal care, digital music, etc. but for our analysis we stuck only to reviews from 'ELECTRONICS'. The format of

the dataset included the Reviewer ID, Product ID, Rating of product, and Unix time of review. This dataset had 1689188 product reviews for about 63001 products. The total count of unique reviewers were 192403

We encountered a stumbling point while attempting to scrape FakeSpot.com as it did not allow us to generate a scrapable URL from the ASIN. The tool had an internal mapping of ASIN to product name which it appended in the URL and thus leaving no particular way to automate the hitting of the FakeSpot webpage for a set of ASINs.

One significant challenge was to scrape data from the the amazon website for every ASIN in the above list. Particularly to get the product description and other product details which could potentially impact the quality of a review and eventually help us classify it as fake or not-fake. Some tools that helped us achieve this throughout the way are described below:

Tools for pulling the data:

- BeautifulSoup
- Selenium
- Webdriver

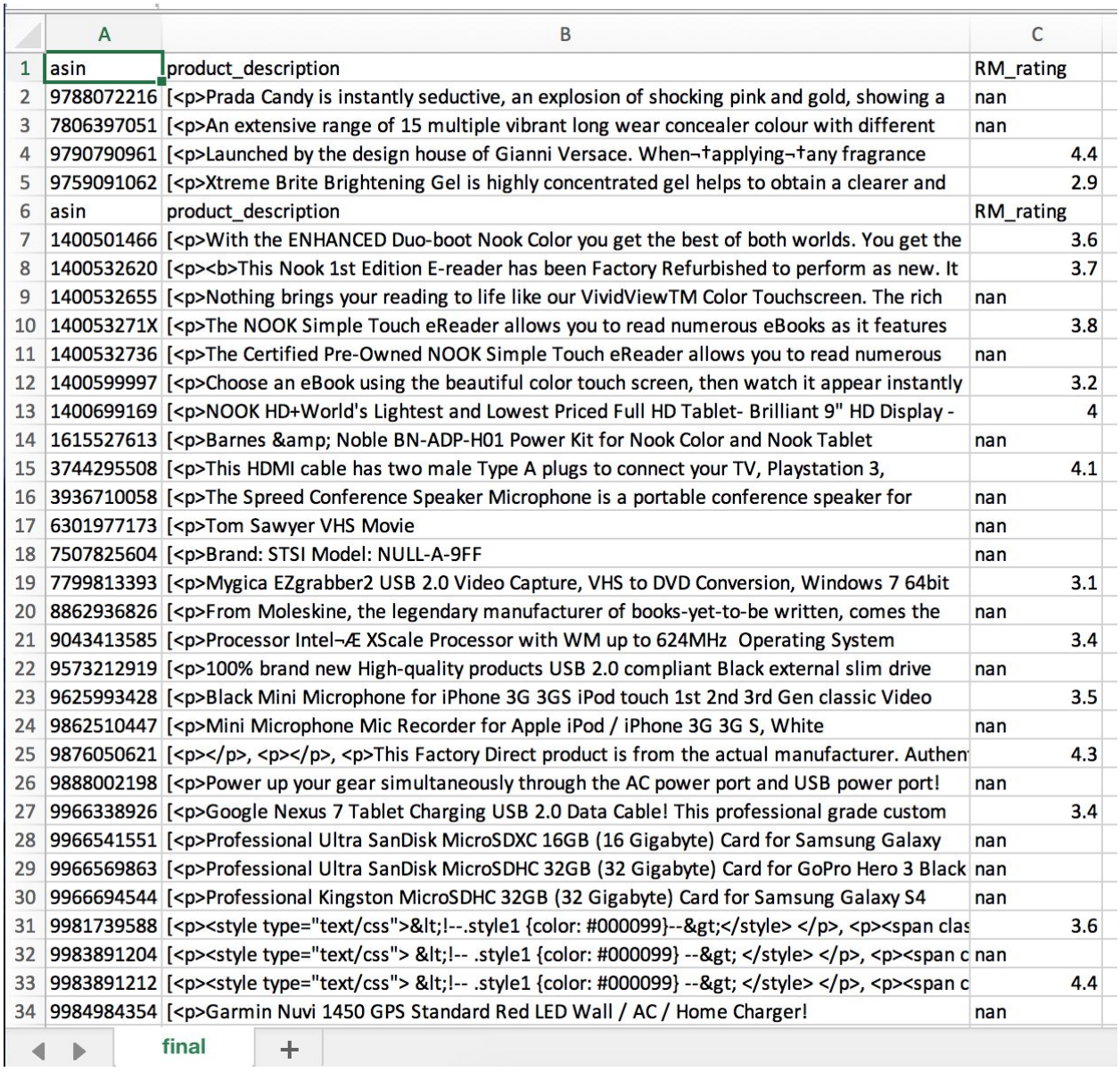
Initial Logic for Data Aggregation:

- Check for the product page on Amazon.com through the ASIN
- If exist, pull the date and get the review score from ReviewMeta.com
- If RM page not exist, put “Product Not Found” and update the score as “N/A”

In order for the above to run, it uses BeautifulSoup and the Selenium WebDriver for Google Chrome, which was needed to install separately and point to in our script or else install to either usr/local/bin/chromedriver for OSx/Linux or C:\chromedriver\chromedriver\ for Windows.

Similar scraping efforts were directed from ReviewMeta[2] towards reviewmeta.com/ASIN and the tags ‘adjusted rating’ was scraped and aggregated to our dataframe. Thus completing our data scraping efforts for all ASIN values in the Stanford Dataset.

Here’s a screenshot of our final scraped dataset:



	A	B	C
1	asin	product_description	RM_rating
2	9788072216	[Prada Candy is instantly seductive, an explosion of shocking pink and gold, showing a	nan
3	7806397051	[An extensive range of 15 multiple vibrant long wear concealer colour with different	nan
4	9790790961	[Launched by the design house of Gianni Versace. When¬†applying¬†any fragrance	4.4
5	9759091062	[Xtreme Brite Brightening Gel is highly concentrated gel helps to obtain a clearer and	2.9
6	asin	product_description	RM_rating
7	1400501466	[With the ENHANCED Duo-boot Nook Color you get the best of both worlds. You get the	3.6
8	1400532620	[<b>This Nook 1st Edition E-reader has been Factory Refurbished to perform as new. It	3.7
9	1400532655	[Nothing brings your reading to life like our VividViewTM Color Touchscreen. The rich	nan
10	140053271X	[The NOOK Simple Touch eReader allows you to read numerous eBooks as it features	3.8
11	1400532736	[The Certified Pre-Owned NOOK Simple Touch eReader allows you to read numerous	nan
12	1400599997	[Choose an eBook using the beautiful color touch screen, then watch it appear instantly	3.2
13	1400699169	[NOOK HD+World's Lightest and Lowest Priced Full HD Tablet- Brilliant 9" HD Display -	4
14	1615527613	[Barnes & Noble BN-ADP-H01 Power Kit for Nook Color and Nook Tablet	nan
15	3744295508	[This HDMI cable has two male Type A plugs to connect your TV, Playstation 3,	4.1
16	3936710058	[The Spreed Conference Speaker Microphone is a portable conference speaker for	nan
17	6301977173	[Tom Sawyer VHS Movie	nan
18	7507825604	[Brand: STSI Model: NULL-A-9FF	nan
19	7799813393	[Mygica EZgrabber2 USB 2.0 Video Capture, VHS to DVD Conversion, Windows 7 64bit	3.1
20	8862936826	[From Moleskine, the legendary manufacturer of books-yet-to-be written, comes the	nan
21	9043413585	[Processor Intel¬Æ XScale Processor with WM up to 624MHz Operating System	3.4
22	9573212919	[100% brand new High-quality products USB 2.0 compliant Black external slim drive	nan
23	9625993428	[Black Mini Microphone for iPhone 3G 3GS iPod touch 1st 2nd 3rd Gen classic Video	3.5
24	9862510447	[Mini Microphone Mic Recorder for Apple iPod / iPhone 3G 3G S, White	nan
25	9876050621	[, , This Factory Direct product is from the actual manufacturer. Authen	4.3
26	9888002198	[Power up your gear simultaneously through the AC power port and USB power port!	nan
27	9966338926	[Google Nexus 7 Tablet Charging USB 2.0 Data Cable! This professional grade custom	3.4
28	9966541551	[Professional Ultra SanDisk MicroSDXC 16GB (16 Gigabyte) Card for Samsung Galaxy	nan
29	9966569863	[Professional Ultra SanDisk MicroSDHC 32GB (32 Gigabyte) Card for GoPro Hero 3 Black	nan
30	9966694544	[Professional Kingston MicroSDHC 32GB (32 Gigabyte) Card for Samsung Galaxy S4	nan
31	9981739588	[<style type="text/css"><!--.style1 {color: #000099}--></style> , <span clas	3.6
32	9983891204	[<style type="text/css"> <!-- .style1 {color: #000099} --> </style> , <span c	nan
33	9983891212	[<style type="text/css"> <!-- .style1 {color: #000099} --> </style> , <span c	4.4
34	9984984354	[Garmin Nuvi 1450 GPS Standard Red LED Wall / AC / Home Charger!	nan

final

Some considerations while performing the scraping were as follows:

- Web scraping is not being liked by site owners
    - Amazon does not allow site scraping
    - Lots of detection mechanism in place.
- Too many ways to do it

    - Too many outdated guides in the wild
    - Script vs Headless
- Considerations
    - Asking for the permission to scrape
    - Don't DDoS
    - Plan ahead of time.

Analysis and Methods Used

Our primary approach involved using semantic similarity approaches to make predictions in understanding fake review generation. To enunciate on the above, outlined are two approaches:

Approach 1: Similarity between reviews

- Each review has an associated review score
- If reviews are similar and scores are high, can we infer they are likely to be written by paid reviewers?
- Cluster similar reviews together and compare scores/stars between them
- If they hover around high scores, we may be able to use as a feature

Approach 2: Using product descriptions as feature set

- Each review has an associated product description
- If reviews for a particular product have more similarity to the product description, it could be paid review
- Normalize product descriptions and extract features from each description
- Process the reviews corpus and vectorize the documents
- Compare reviews with the extracted features from product description
- Assign weightage or score to the reviews based on the similarity.

Once our dataset had been aggregated, our first step was to pre-process the data and the steps involved were:

- **Stemming and Lemmatization** :
  Worked by cutting off the end or the beginning of the word, taking into account a list of common prefixes and suffixes that can be found in an inflected word. This indiscriminate cutting can be successful in some occasions, but not always, and we affirm that this approach worked well in our case owing to similarity measures.

- **Tokenization**:
  Tokenization is taking a text or set of text and breaking it up into its individual words and was performed by simply splitting the string up into relevant tokens.

- **Vectorization**:
  We chose the 'Bag of Words' technique to define a fixed length vector where each entry corresponded to a word in our predefined dictionary of words. The size of the vector equaled the size of the dictionary. Then, for representing a text using this vector, we counted how many times each word of our dictionary appears in the text and we put this number in the corresponding vector entry.

Before we had pre-processed the product descriptions and the corresponding reviews for each ASIN we wanted to ensure consistency in our dataset across the three disparate sources of data we had. In order to achieve that we then proceeded with the following:

- **Filtering from ReviewMeta.com**:
  We filtered out all those entries that did not have a corresponding ReviewMeta score available from the scraped webpage. Our scraping ensured validation to handle timeout cases and made sure that no cases were missed due to scraping complexities.
- **Filtering from Amazon.com:**
  We marked all the products that did not exist on Amazon anymore as 'N/A' and also

went on to filter those results from the Stanford Dataset.

Next, we wanted to extract important features after the preprocessing was complete. We focus on extracting certain features from the documents (or reviews in our case). Thus, **term frequency – inverse document frequency** (also called tf-idf), is a well know method to evaluate how important is a word in a document and was used to convert the textual representation of information into a Vector Space Model (VSM).

Once we had selected all the processed terms from the document and converted it to a dimension in the vector space we created a index dictionary of the words of the review set, using the review and the product description from the each set. The tf-idf weight was composed by the number of times a word appears in a review, divided by the total number of words in that review and the logarithm of the number of the review in the corpus divided by the number of review where the specific term appears.

Now that we have TF-IDF calculated already, as the next step, we employed '**Cosine Similarity**' algorithm to calculate the similarity among the documents. Let's dive a little into what Cosine Similarity is before we jump ahead. We can calculate the similarity between pairs of the documents using 'cosine similarity' algorithm, which measures the **cosine of the angle between two vectors. In this case, each document can be presented as a vector whose direction is determined on a set of the TF-IDF values in the space.**

To understand this concept further, let's assume we have only two terms (or words), "supply" and "transparency" as text data to simplify this example, and we have TF-IDF values calculated per term for Document A and Document B like below.

	supply	transparency
Document A	40	60
Document B	50	20

If we visualize their positions and if the two vectors are pointing to a similar directions then we can say these two documents are similar. So in order to measure and compare the similarity we want to calculate the cosine of the angle between the two vectors. Now, of course we have much more terms than just the two for these documents we are working with. This means that we have 'high' dimensional space rather than the two dimensional space. But the concept is still the same.

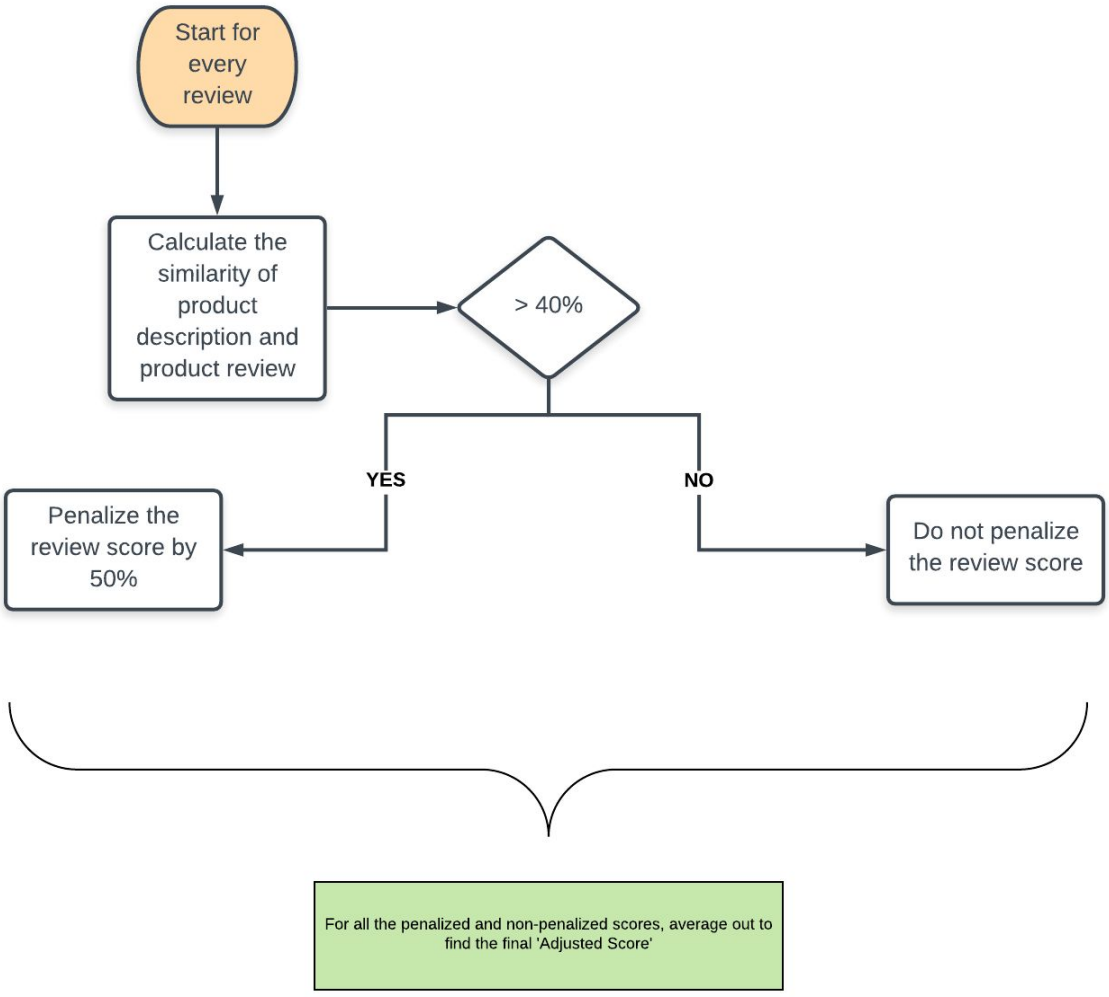
This helped us obtain a similarity score for each review with respect to the product description. A look at it is as follows:

```
In [89]: asin = "B00001P4ZH"
         sim_scores = get_sim_score(asin, df, electronics)

In [90]: sim_scores

Out[90]: [0.16837835023333778,
          0.3289518118342984,
          0.292946588275165,
          0.32119754722481797,
          0.39022578726973123,
          0.418037148696135,
          0.33897231222805013,
          0.18645910847127656,
          0.3316137004223539,
          0.46517611846835777,
          0.39418133587117804,
          0.19799634851011316,
          0.4203892259149084,
          0.30388010713163427,
          0.3092789496181269,
          0.29122572148514086,
```

Now that we have a bunch of similarity scores - we want to derive an analysis of the ‘fakeness’ of a product. A simple logic behind it was as follows:



In essence:

- Iterate through all reviews for every ASIN to create a list of similarity scores that are penalised as follows:

  if (similarity >40%):

  penalizing 50%

  else

  no penalization
- Aggregate by averaging out the weights for each ASIN
- Create two comparison vectors of our generated score vs RM score

Finally, here's what our calculated score looks like when kept right aside the scores delivered by ReviewMeta with **penalization point as 40%**:

```
In [81]: comparisons_df.head(20)
```

Out[81]:

	asin	our_score	review_meta_score
0	B00000J061	4.446428571428571	3.7
1	B00000J1EQ	4.6571428571428575	4.2
2	B00000JPPI	4.0	4.2
3	9876050621	3.2	4.3
4	B00000K13L	4.725	4.5
5	B00004SYKO	5.0	3.2
6	B00000J1TX	4.738095238095238	4.0
7	B00001P505	4.2894736842105265	4.2
8	7799813393	3.5	3.1
9	B000023VW2	4.1891891891891895	4.1
10	9043413585	3.4285714285714284	3.4
11	B00004SY4H	4.019607843137255	4.7
12	9625993428	3.8	3.5
13	B00004SB92	3.6352459016393444	3.3
14	B00001P4ZH	3.9553941908713695	4.4
15	1400699169	4.291666666666667	4.0
16	B00001W0H1	4.4	4.4
17	B00003CWDH	4.48062015503876	4.1

Here's another result, when we changed the **penalization point to be 70%:**

```
In [105]: comparisons_df.head(20)
```

Out[105]:

	asin	our_score	review_meta_score
0	B00000J061	4.535714285714286	3.7
1	B00000J1EQ	4.6571428571428575	4.2
2	B00000JPPI	4.0	4.2
3	9876050621	3.2	4.3
4	B00000K13L	4.725	4.5
5	B00004SYKO	5.0	3.2
6	B00000J1TX	4.738095238095238	4.0
7	B00001P505	4.434210526315789	4.2
8	7799813393	3.5	3.1
9	B000023VW2	4.243243243243243	4.1
10	9043413585	3.4285714285714284	3.4
11	B00004SY4H	4.7254901960784315	4.7
12	9625993428	3.8	3.5
13	B00004SB92	4.131147540983607	3.3

Some Considerations:

We would want to find the Euclidean distance between our score vs reviewmeta score to better understand the correctness of our results and would also be interested to explore optimality with comparing different results using different penalizing factors & similarity percentage. As an additional effort, we could look into incorporating sentiment analysis and diving into the negative end of the spectrum.

Challenges Faced:

- Longer than expected time taken with scraping web data and working with Selenium webdriver to hit pages as a human simulation.
- No easy way to build prebuilt URL to get the result from Fakespots.com and thus unable to gather results from a standard website.
- No baseline exists and no-predefined accuracy measure to match up and look to compete performance against.
- Due to lack of time we could not investigate further into features like 'helpfulness of a review'.

Conclusion

With a growing trend in online shopping there is a need currently to provide an authentic environment for analyzing product reviews and ratings. This paper expounds a methodology to identify fake ratings among genuine ones over time. This can be used to implicitly identify fake reviewers in cyberspace. Amazon Product Datasets were used for the analysis presented in this paper based on which a hypothesis was proposed to understand the generation of fake reviews. Our results conclude that semantic similarity between product description and review is a strong indicator of a review being fake or genuine.

References

[1] Jure Leskovec and Andrej Krevl, "SNAP Datasets: Stanford Large Network Dataset Collection", June 2014 http://snap.stanord.edu/data

[2] ReviewMeta.com analyzes millions of reviews and helps you decide which ones to trust, scraped from: https://reviewmeta.com/