Neil Nguyen

COEN 353

Liu

November 28, 2021

Term Project Report

Problem Significance:

With the speed and size of media these days, how can one tell the reliability of a product review, more specifically something as subjective as a movie review? Whenever a movie is released it's often plastered with countless 4 or 5-star reviews. You see them at the end of the commercials, the front page of magazines, and the headlines of articles all the time. And then when you want to go and actually read these reviews they'll only give you one sentence or even just a snippet of that review that tells you nothing about the movie quality. Something like: "I loved it!" or "Fantastic!". This is a significant problem for the average movie consumer who has to decide what to pay for and watch. How can they truly trust what they're reading when it's so overwhelmingly positive without any concrete evidence?

Problem Challenges:

The largest challenge with judging movie reviews is the subjectivity of them. Movies are a very special form of media that are meant to hit very many different people in very different ways. Very often you have people with complete polar opposite feelings of the same movie. How exactly can you tell who's "right" or "wrong"? The solution is not to discredit one or the other but to instead judge the quality of that person's review. Both reviewers can be reliable! What makes a review "dishonest" or "unreliable" are two general things: whether they're real or not and whether they're helpful or not. (These two qualities are not mutually exclusive) We'll further dive into the difference between these two qualities in the articles I researched.

Current Literature (1):

Really when it comes to dishonest online reviews, the two articles I read addressed two different types you may encounter. In “Amazon Hits Chinese Sellers With Crackdown on Fake Reviews” by Bloomberg this article addressed the issue of massive amounts of fake reviews. Apparently Amazon initially encouraged this type of behavior by offering free products to people willing to review them. As a result sellers wrote thousands of quick, fake reviews to accrue these benefits while also helping Amazon generate traffic. It’s important to note that while the article specifically addresses the issue of Chinese super sellers doing this, it’s very possible this is also happening on a global scale outside of China. These are the types of reviews the human eye could usually be able to tell apart from the real ones. They were very short, indescriptive, and possibly littered with grammatical errors. The purpose of these reviews are to quickly bolster up a product or movie’s score immediately. When looked at by the average user they would see that a hypothetical product has thousands of positive reviews but upon further inspection you could tell that most of the reviews are very robotic. These reviews are neither real nor helpful.

Current Literature (2):

The other article titled “Remembering David Manning, Sony’s Fictional Film Critic” by MentalFloss detailed a specific case yet not uncommon where the director of creative advertising at Sony created an alter ego known as David Manning. David Manning sent in movie reviews to local newspapers that were extremely well-written and detailed. The only issue is that “Manning” only reviewed Sony movies that were always in an extremely positive light. Fortunately some reporters caught on this pattern and were able to rightfully expose Sony for it. These types of reviews are meant to attract the attention of a more dedicated movie audience, the ones that only trust the opinion of reputable reviewers (instead of the masses at Amazon for example). While there exists much less of these reviews they still present the same problem but in a different form: fewer yet more powerful.

Current Literature (3):

Finally the 3rd article I read was the “Complete Guide to Identify Fake Reviews On Amazon” by geekflare. This article detailed simple yet effective strategies to identify fake reviews on Amazon. Seeing just one tell may not be enough to immediately write off a review but here are some that in conjunction might be very telling: generic tone, dead words, short review length, broken english, etc.” This article also described the origin of fake reviews. Fake reviews can easily be purchased online and for all major product sites, not just Amazon. There is clearly a market for it! Additionally it’s not always AI that writes these reviews, very often reputable product/movie reviewers are also willing to sell their reputation and write fake reviews as a side hustle. As a result they often fall into similar patterns in their purchased reviews as mentioned prior. If we’re aware of these patterns we will more easily be able to catch them with a human eye and better yet, code them into an algorithm for the experiment!

Proposed Solution:

Mainly based on the criterias detailed in *Current Literature (3)* I will code out an algorithm that gives every review a “validity score”. When reading a review, users are expected to both use their own judgement and also consider the provided validity score when coming to a decision regarding the movie.

Experiment Preparation:

The coding language I will be using is Python. I’m most comfortable with this language and it’s able to handle data in the most flexible ways. The dataset contains millions of online movie reviews from Amazon between 2000 to 2012. While these movie reviews contained many parameters: product ID, user ID, profile name, helpfulness, score, time, summary, text. Here’s what the Python object for a movie review looked like on the left:

```python
class review:
    def __init__(self, product_id):
        self.product_id = product_id
        self.user_id = ''
        self.profile_name = ''
        self.helpfulness = 1.0
        self.score = 0.0
        self.time = 0
        self.summary = ''
        self.text = ''

        self.dead_flag = 0
        self.short_flag = 0
        self.unhelp_flag = 0
        self.perf_imperf_flag = 0

    def set_user_id(self, user_id):
        self.user_id = user_id
    def set_profile_name(self, profile_name):
        self.profile_name = profile_name
    def set_helpfulness(self, helpfulness):
        self.helpfulness = helpfulness
    def set_score(self, score):
        self.score = score
    def set_time(self, time):
        self.time = time
    def set_summary(self, summary):
        self.summary = summary
    def set_text(self, text):
        self.text = text
```

```python
def get_data(cap):
    txt = open("movies.txt", errors="ignore")
    data = []
    counter = 0
    for line in txt:
        # print(line)
        if counter == 0:
            temp = review(line[19:]) #19:
        elif counter == 1:
            temp.set_user_id(line[15:]) #15:
        elif counter == 2:
            temp.set_profile_name(line[20:]) #20:
        elif counter == 3:
            temp.set_helpfulness(line[20:]) #20:
        elif counter == 4:
            temp.set_score(line[14:]) #14:
        elif counter == 5:
            temp.set_time(line[13:]) #13:
        elif counter == 6:
            temp.set_summary(line[16:]) #16:
        elif counter == 7:
            temp.set_text(line[13:]) #13:
            data.append(temp)
        elif counter == 8:
            counter = -1
        counter += 1
        if len(data) > cap:
            break
    return data
```

The original dataset was a text file that was over 15 gigs. Each line represented a parameter of the review, with 8 parameters total each review contained 8 lines of text. The code I wrote to store that data into python objects is on the right. One important thing to note is that I had to put in a “cap” variable to manually stop collecting data after a certain amount of movie reviews. This is for two reasons: the object insertion was very time consuming so I could only wait so long and after a certain number of reviews (100000) the text file became inconsistent and glitched out the reviews. You can see that in my get_data function I had to be very specific with my string slicing to properly collect the data. Fortunately, 100000 reviews is more than enough reviews to work with.

Experiment Step 1 - Flagging:

I came up with 4 discrete criterias that were considered unreliable when in a review. I’ll describe these in order as well as show the code. The code I wrote simply took every review that matched that criteria and stored it in a list. They were more of a “filter” if anything, except

you filtered out everything *else*. Additionally if a review matched a criteria, its “flag” was flipped for it. Those are the variables shown above in the object definition that weren’t explained yet. The first discrete criteria were “dead words”. Each flag is initially set at 0 but once tripped is set to 1.

```
self.dead_flag = 0
self.short_flag = 0
self.unhelp_flag = 0
self.perf_imperf_flag = 0
```

Dead words were essentially the type of words you were taught to *not* put in your essays as a kid. There’s no concrete definition on what words should be considered dead so I simply looked up some online graphics that showed some common ones. Here are the dead words I chose to use followed by the code for the dead word filter:

```
dead_words = ["awesome", "cool", "awful", "fun", "funny", "good", "great", "mad", "nice", "pretty", "scared", "very", "really", "quite"]
dead_reviews = dead_word_filter(dead_words, data)
```

```
def dead_word_filter(deadwords, dataset):
    ret = []
    dead = set(deadwords)
    for data in dataset:
        temp = set(word.lower() for word in re.sub(r'[^\w]', ' ', data.text).split())
        if dead.intersection(temp):
            data.dead_flag = 1
            ret.append(data)
    print("There are {} reviews with words from {}.".format(len(ret), dead))
    return ret
```

The next criteria was based on review length. If the description of the review had less than a certain amount of words it was deemed “too short” and flagged for it. In the experiment I settled on 20 for an acceptable word count. Here’s the code:

```
short_reviews = len_filter(20, data)
```

```
def len_filter(length, dataset):
    ret = []
    for data in dataset:
        temp = re.sub(r'[^\w]', ' ', data.text).split()
        if len(temp) < length:
            data.short_flag = 1
            ret.append(data)
    print("There are {} reviews with less than {} words.".format(len(ret), length))
    return ret
```

The next criteria was based on an already provided Amazon scoring called “helpfulness”. Users are allowed to vote a review as “helpful” or “unhelpful”. So the “helpfulness” score of a review is how many people voted it as helpful versus how many people voted in total. So it’s always a number between 0 and 1. The helpfulness filter takes every review below a certain helpfulness threshold, the one I chose was 0.5. Here’s how that looks:

```
unhelpful_reviews = helpfulness_filter(0.5, data)
```

```
def helpfulness_filter(helpfulness, dataset):
    ret = []
    for data in dataset:
        temp = re.split('/|\n', data.helpfulness)
        if not int(temp[1]) or int(temp[0])/int(temp[1]) < helpfulness:
            data.unhelp_flag = 1
            ret.append(data)
    print("There are {} reviews with less than {} helpfulness.".format(len(ret), helpfulness))
    return ret
```

The final criteria was whether a user gave a movie a perfect or imperfect score, meaning a 5 or a 1. These were considered potentially “exaggerated” scores and flagged as such. Here’s how that looks:

```
perf_imperf_reviews = perfection_filter(data)
```

```
def perfection_filter(dataset):
    ret = []
    for data in dataset:
        temp = float(data.score[:-1])
        if temp == 1.0 or temp == 5.0:
            data.perf_imperf_flag = 1
            ret.append(data)
    print("There are {} reviews with a score of 1 or 5.".format(len(ret)))
    return ret
```

In short, every review was sorted into bins of each criterias they matched and flagged as such. Here are the counts of each criteria:

```
There are 73634 reviews with words from {'great', 'scared', 'funny', 'pretty', 'quite', 'awesome', 'fun', 'ni
ce', 'very', 'awful', 'mad', 'good', 'really', 'cool'}.
There are 3426 reviews with less than 20 words.
There are 48393 reviews with less than 0.5 helpfulness.
There are 63230 reviews with a score of 1 or 5.
```

Experiment Step 2 - Scoring:

A review was given a ‘validity” score based on how many flags it tripped. The initial validity score of a user is set to 5. For every flag they trip 1 is subtracted from it, this provides us with a validity range of [1,5]. Here is the calculation function followed by the validity spread of the dataset and the validity filter used to count those:

```
def calculate(self):
    return 5 - (self.dead_flag + self.short_flag + self.unhelp_flag + self.perf_imperf_flag)
```

```
There are 3230 reviews with a validity score of 5
There are 27753 reviews with a validity score of 4
There are 47148 reviews with a validity score of 3
There are 20846 reviews with a validity score of 2
There are 1024 reviews with a validity score of 1
```

```
def validity_filter(validity, dataset):
    ret = []
    for data in dataset:
        if data.calculate() == validity:
            ret.append(data)
    print("There are {} reviews with a validity score of {}".format(len(ret), validity))
    return ret
```

Some important things to note here are that each criteria is weighed equally in the scoring function. The validity score of 3 had the highest volume with decreasing volume in either direction.

Experiment Step 3 - Criteria Insights:

Each criteria had its strengths and weaknesses. We will deeply analyze those now. How I analyzed them was I pulled random reviews from each filter and personally read them. Based on this I judged whether or not the criterias were helpful.

The reviews with dead words had by far the largest amount. While in some cases “dead words” were used in “dead” manners in reviews, that was usually not the case. The words considered dead can very well be used in a helpful, descriptive manner. Simply flagging a review for containing these words is unfair and holds reviewers’ vocabularies to too high of a standard. They’re writing movie reviews, not an essay! This criteria was weak.

The short reviews were a somewhat medium strength criteria. That being said, they were a little inconsistent. While short review length could surely be used to flag the worst short reviews, where is it that the line is meant to be drawn? There were many unhelpful reviews with 5 words or less but could that be said about all reviews with 20 words or less? There were both helpful and unhelpful reviews with 15-20 words, so should the line be drawn lower? This is why this criteria was inconsistent.

The strongest criteria was the one provided by Amazon already: the helpfulness score. These are voted by actual users and as a result were mostly helpful. Movie reviews that were unhelpful or unrelated to actual movie content were usually voted “unhelpful” by readers. This

implies that in general the more participating users in this voting the more useful the voting becomes. However, there are some drawbacks where users might have different understandings of what they're meant to vote for helpful or unhelpful. For example, there was a review I read that was unrelated to the movie entirely and instead based on the DVD player compatibility of a particular movie's DVD release. Many Amazon voters considered this a helpful *product* review because it convinced them to *not* purchase the DVD release but this isn't a very helpful *movie* review. For the most part however, this criteria is strong.

The weakest review by far was the perfect/imperfect criteria. We learned that you can somewhat tell when a reviewer is exaggerating when they put the maximum or minimum score available. However, in this particular dataset the scoring range was simply too small to use this criteria. It's very possible for users to genuinely give movies scores of 1 or 5, so they shouldn't be written off for simply having that opinion. However, this criteria may be more helpful in a dataset where ratings could contain decimals or maybe ranging from 1 to 100.

Experiment Step 4 - Validity Insights:

I will give the insights on each group of validity scores 1 through 5 individually. Each one varies in reliability for different reasons.

The reviews with a validity score of 5 were mostly strong. They were descriptive in length, contained only descriptive words, gave more on-the-fence scores, and were considered helpful by readers. That being said there were some unhelpful reviews that managed to sneak by every criteria check and be scored a 5. So this grouping is not the end-all-be-all when reading a review but it is helpful.

The reviews with a validity score of 4 were also strong but now they contained many reviews that had some dead words, didn't reach the review length, weren't voted as helpful (or voted at all), or had a score of 1 or 5. This group of reviews may even be considered *more* helpful than the score of 5. That is because reviews of score 1 and 5 are permissible here and

users could read a much larger variety of opinions here. Also there were dead words allowed that could be descriptive as well.

The next group was validity scores of 3. This was by far the largest grouping. This was probably the *least* telling group because it contained both strong and weak reviews. This is to be expected because it is the middle ground. It's fortunate that this is where we really start to see a higher volume of unhelpful reviews because now users should be able to generally tell what exactly a validity score they're looking for is. While 3 may not necessarily mean a review is unhelpful it means it has potential to be and users are responsible for giving the final verdict.

The next group was validity scores of 2. Most of these reviews were unhelpful that I read and there were some unfortunate helpful reviews that fell into here for coincidentally hitting every criteria check. When looking at a validity score of 2 review, users should read the review and then come to their own decision about trusting the review even if most of the time the validity score is accurate.

The final group had validity scores of 1. There's not much to say about this group, most of the reviews I looked at from here were genuinely unhelpful.

Final Insights:

Based on the reviews and scorings I looked at I definitely think that validity scores have potential to be useful in judging movie reviews. One issue is that my criterias chosen might have been *too* discrete. That being said, the more discrete the criteria is the more helpful it may be in scooping up bot reviews. However, this is a double-edged sword. If they're stronger for hunting bots it might be weaker in judging actual human reviews. Another fallback of my experiment is that I did not use enough criterias. Only 4 is not enough because it leaves the validity scores range too small, we saw previously how the score of 3 was not very helpful in helping a user come to a decision. Some other criterias to be considered might be time of review relative to movie release date or user reputation of each reviewer. Finally, perhaps this experiment may not be best when working with movies. That is because (as stated in the beginning) movies are

extremely subjective and appeal to everybody in different ways. Maybe this project could be better used towards scoring product reviews or app reviews.

Resources:

1. “Amazon Hits Chinese Sellers With Crackdown on Fake Reviews.” *Bloomberg.com*, Bloomberg, https://www.bloomberg.com/news/articles/2021-08-18/amazon-amzn-cracks-down-on-fake-reviews-hitting-chinese-retailers.
2. “Complete Guide to Identify Fake Reviews on Amazon.” *Geekflare*, 8 Sept. 2021, https://geekflare.com/identify-amazon-fake-reviews/.
3. “Remembering David Manning, Sony's Fictional Film Critic.” *Mental Floss*, 15 Apr. 2021, https://www.mentalfloss.com/article/645297/david-manning-sony-fake-film-critic.