Brent Shulman

Mark Liberman

LING 001

Final Project

Amazon Review Semantic Analysis

OBSERVATIONS

While looking to buy a product on amazon a few weeks ago, I scrolled down and started reading the reviews. People are now more than ever writing reviews in online message boards, such as Amazon, after they used certain products or services. Thus, places like this have become forums to communicate with other people about similar interests. Reviews can have huge impacts on business, as well as many other fields. For a business, they can gain useful knowledge about how their end users feel about their products. Furthermore, these review boards are also social, they can influence other people's decisions and opinions regarding certain product. Thus there is power in knowing and understand the language of these communities.

Each review I read had its own personal user's story detailing why they did or did not like the product. These reviews also have a star number associated with them. Its seems trivial and obvious to say that there is a correlation between the star level of a review and the words it contains, but this notion actually carries with it something truly interesting about language. The words that we use carry certain weights. "I love it!" is stronger than "I like this product but it could be better". Again, a seemingly obvious statement but one that clearly illustrates that words such as "love" and "like" with similar definitions have very different semantic or pragmatic meanings.

All code can be found at: https://github.com/shulmanbrent/Amazon-Reviews-Semantic-Analysis

HYPOTHESIS / QUESTIONS

The question I then raise is, can we quantify this weight? Is there a way to definitely show that "love" shows that a user is more positive towards a product than "like"? (And vice-versa is "hate" relatively stronger than "dislike") Can we then use the words in an article to predict the sentiment of a review, or the score that a user gives? These are the questions that I have investigated in this project. To do this, I have analyzed the words of product reviews and attempting to categorize them mathematically.

There is a wealth of information to be gained from classifying reviews according to their sentiment. There are many ways that this information can be used. Not only would it be useful for business to accrue such knowledge about thier products, but it could also make it easier for people to get a baseline feeling about the quality of a product. Sites like Rotten Tomatoes are already using these types of semantic analysis, and their success is proof of its relevance for consumers as well. Furthermore, a one could easily be extend this to gain some extremely insightful knowledge about the behavior of specific groups of users and their opinions. This type of information about users is every company's (and more specifically their advertising department's) dream.

METHODOLOGY / DATA

I will uses methods and techniques I have learned in computational linguistic, natural language processing and machine learning techniques to understand these user reviews. I used Python for this project, as it allows me to use the Natural Language Tool Kit (NLTK) to help me analyze the data. After searching the web I found a very large dataset (11GB) of Amazon reviews across many different topics (from a Stanford group called SNAP – Stanford Network Analysis Project)[1]. After contacting a member of SNAP, I was granted access to their corpus of data and downloaded it. However, I decided that this set will be too large for me to manage by myself. It

would have made testing much too long and not allow for possible iteration and development of the algorithm if there are errors. Instead, I decided to focus on only one of the subtopics provided in the source page. This allowed me to set up a successful model of predication that would allow for scale to larger datasets in the future.

With this in mind, I decide to focus on just reviews of "Baby" products. After talking with many other people, I determined that a topic that many people are interested in hearing other's opinions on in "Baby" products. The transition from to motherhood can be very stressful for many women. I and others have observed anecdotally that mother almost always want "what is best" for their children. Often this in reference to the different types of product they will be using with their child. Many mothers are interested in the opinion of other mothers. This gives them a sense of whether they should use a product on their own baby or not.

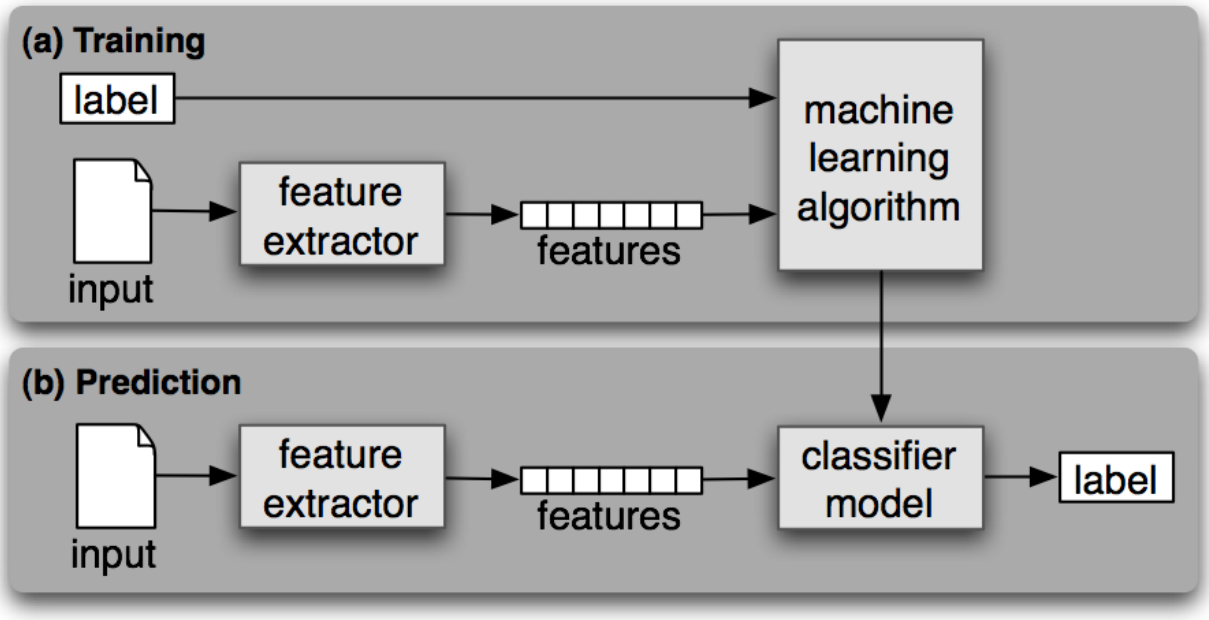
I decided go about researching these questions in two different ways. First I used only the given information in the dataset. Specifically, the review score and the review text given in the JSON object. An example of these objects can be see below:


```
{"review/profileName": "C. COURVAL \"CINDY C.\"",
"product/price": "unknown",
"review/time": "1200528000",
"product/productId": "B000C9DZ4W",
"review/helpfulness": "2/2",
"review/summary": "RUFF",
"review/userId": "A1IGLU07FR5WG2",
"product/title": "Sumersault Polka Dots Sheet",
"review/score": "1.0",
"review/text": "THIS SHEET IS A CRIB SHEET FOR A BABIES BED. IT IS VERY COARSE, HARSH AND NOT POLKA DOTTED."}
```


(Clearly not happy)

All code can be found at: https://github.com/shulmanbrent/Amazon-Reviews-Semantic-Analysis

Then secondly I used a crowdsourcing method to help me categorize the text. The basic model for how I will be tackling this research is best illustrated by the picture on the next page:



For the first part of the research project, the inputs were the reviews of the products contained in the Baby.gz file. Using a method called `parse` given by SNAP, I am able to access a `list` of the reviews and their properties in the form of a `dictionary`. This allowed me to pull out only specific and pertinent information about each of the reviews. In this case, only review *score* and the review *text* were needed. We can then use this to help define out data. The first step is to choose our labels. These are essentially the buckets into which I am classifying the reviews. During the first iteration I chose the five possible review values to be the labels 1.0 → 5.0. As you will see below this was a bit ambitious.

I then needed to create a "*feature extractor*" through which I will take the given reviews text and denote which information is used to determine category of the review. For this, I will initially use the top 2000 most common words throughout all of the 187,000 reviews. This a decent

method for determining pertinent information about the reviews because there is more data about these words. It will be easier to build a model that represents how these words affect the sentiment of the review. However, this oversimplifies many factors. For example, some infrequently used words may be extremely telling as to the sentiment of a review (ex: "f**king" is probably very likely to denote that a user did not like a product; however, it is likely used infrequently on message boards). Despite this, it is still a decent and logical assumption.

I then created *featureExtractor.py* to parse the given review text into just its constituent words. I make sure not to include digits, as well as remove any punctuation from the strings. This method however also removes a feature that is possible semantically telling. Often increase punctuation (ex: exclamation points) could make it more likely that a review fits into one category or another. While I have simply chosen to look at each word individually, often the syntax and of sentences and phrases if often important. This can be analyzed, but I have chosen not to do this for this project because it can be very complicated.

Now that I had the feature and the label matched up appropriately, we have something of the following form:

*** *This is only an example feature label match; order is not the same for all feature cases****

```
[ { Contains(baby): True; Contains(hate): False; … Contains(like):
True}, 1.0]
```

Once this is done for all 187,000 items in the review dataset, I can then use a machine learning algorithm to help build a model to predict the sentiment of a previously unseen review. For this project I have chosen to use Naïve Bayes because of the ease by which I can build a Naïve Bayes Classifier using NLTK. The basic formula for this method can be seen below:

$$p(C|F_1,\ldots,F_n) = \frac{p(C)\ p(F_1,\ldots,F_n|C)}{p(F_1,\ldots,F_n)}.$$

All code can be found at: https://github.com/shulmanbrent/Amazon-Reviews-Semantic-Analysis

The naivety of the Naïve Bayes comes from the assumption that the probability of the any particular feature being true is independent of any other particular feature being true. This is obviously not inherently logical, as it would make sense that the words “love” and “like” appear often together.

Following this, I partition the feature set into 3 separate groups – the train_set, dev_set, and test_set. I will use train set to train the classifier and test its accuracy on the dev_set. This allows me to continue to tweak which features I am using to predict and classify the sentiment of the reviews. Ultimately by improving the accuracy on the dev_set I can test the classifier of the test_set for accuracy. This avoids me simply adjusting my methods to solely fit the test set.

Now, we can finally train the algorithm and test for accuracy. These were my initial results

Accuracy: 0.5429

Most Informative Features

Feature: presence	*Label 1:Label 2*	*Pr[Label 1]: Pr[Label 2]*
highly = True	5.0 : 3.0	= 14.1 : 1.0
loves = True	5.0 : 1.0	= 9.9 : 1.0
love = True	5.0 : 1.0	= 5.1 : 1.0
sturdy = True	4.0 : 1.0	= 4.6 : 1.0
toys = True	3.0 : 1.0	= 4.3 : 1.0
however = True	3.0 : 5.0	= 4.3 : 1.0
perfect = True	5.0 : 1.0	= 3.7 : 1.0
easy = True	5.0 : 1.0	= 3.7 : 1.0
toy = True	4.0 : 1.0	= 3.6 : 1.0
fits = True	5.0 : 1.0	= 3.6 : 1.0
best = True	5.0 : 1.0	= 3.4 : 1.0
amazon = True	1.0 : 3.0	= 3.4 : 1.0
monitor = True	1.0 : 5.0	= 3.2 : 1.0
thought = True	2.0 : 5.0	= 3.2 : 1.0
bit = True	4.0 : 1.0	= 3.2 : 1.0
money = True	1.0 : 4.0	= 3.1 : 1.0
work = True	2.0 : 5.0	= 2.8 : 1.0
seems = True	3.0 : 1.0	= 2.7 : 1.0
though = True	3.0 : 1.0	= 2.6 : 1.0
plastic = True	3.0 : 5.0	= 2.6 : 1.0

All code can be found at: https://github.com/shulmanbrent/Amazon-Reviews-Semantic-Analysis

Initially, this was very unpromising. An accuracy score of .2 would mean the algorithm does not better than chance, but this still means it only gets the answer right 50% of the time. However, after looking over that data, I decided that part of the issue was that I was taking on too many categories. To solve this problem. I let a 4.0 and a 5.0 score be positive, and a 2.0 or 1.0 be negative. I ignored all reviews with 3.0 score for simplicity. Doing this then yielded the following data

Accuracy: 0.826
Most Informative Features

Feature: presence	*Label 1:Label 2*	*Pr[Label 1]: Pr[Label 2]*
perfect = True	pos : neg	= 4.4 : 1.0
love = True	pos : neg	= 3.4 : 1.0
pampers = True	neg : pos	= 3.3 : 1.0
works = True	pos : neg	= 3.3 : 1.0
amazon = True	neg : pos	= 3.3 : 1.0
toy = True	pos : neg	= 3.3 : 1.0
monitor = True	neg : pos	= 3.2 : 1.0
though = True	pos : neg	= 3.1 : 1.0
since = True	pos : neg	= 3.0 : 1.0
plastic = True	neg : pos	= 2.8 : 1.0
item = True	neg : pos	= 2.8 : 1.0
stroller = True	pos : neg	= 2.8 : 1.0
sturdy = True	pos : neg	= 2.7 : 1.0
easily = True	pos : neg	= 2.6 : 1.0
say = True	neg : pos	= 2.6 : 1.0
easy = True	pos : neg	= 2.6 : 1.0
price = True	pos : neg	= 2.5 : 1.0
house = True	pos : neg	= 2.5 : 1.0
reviews = True	neg : pos	= 2.4 : 1.0
toys = True	pos : neg	= 2.3 : 1.0

This yielded dramatically better results than the previous method on the *dev_set* data. Here the algorithm correctly predicts the label almost 80% of the time - not paid for a seemingly simple method. I now apply this method to the *test_set*. The results are below:

Accuracy: 0.8109

All code can be found at: https://github.com/shulmanbrent/Amazon-Reviews-Semantic-Analysis

Thus we can see that what is a naïve method seemingly actually works very well in predicting the results.

With that accomplished, I then worked on the other task, using gather human response and using that as a classifications scheme. I was pointed by a colleague to Mechanical Turk. This site, run by Amazon, allows you to get data for Human Intelligence Tasks, like deciding the sentiment of a review. I paid for over 1000 reviews to be analyzed by this service and was then able to utilize the data in a similar way as above. Here is had to also use neutral as a category as that is what I specified when I gave Mechanical Turk the data.

accuracy: 0.693386773547
Most Informative Features

Feature: presence	*Label 1:Label 2*	*Pr[Label 1]: Pr[Label 2]*
guess = True	Negati : Positi =	17.1 : 1.0
obviously = True	Neutra : Positi =	11.7 : 1.0
lack = True	Neutra : Positi =	11.7 : 1.0
maybe = True	Neutra : Positi =	11.7 : 1.0
finally = True	Neutra : Positi =	11.7 : 1.0
later = True	Neutra : Positi =	11.7 : 1.0
okay = True	Neutra : Positi =	11.7 : 1.0
solution = True	Negati : Positi =	11.0 : 1.0
practical = True	Negati : Positi =	11.0 : 1.0
returned = True	Negati : Positi =	11.0 : 1.0
start = True	Neutra : Positi =	9.1 : 1.0
oz = True	Neutra : Positi =	9.1 : 1.0
pieces = True	Negati : Positi =	9.0 : 1.0
return = True	Negati : Positi =	9.0 : 1.0
washed = True	Negati : Positi =	9.0 : 1.0
regret = True	Negati : Positi =	9.0 : 1.0
disappointed = True	Negati : Positi =	9.0 : 1.0
loud = True	Negati : Positi =	9.0 : 1.0
paying = True	Negati : Positi =	9.0 : 1.0
terrible = True	Negati : Positi =	9.0 : 1.0

As we can see we have a lower rate of accuracy, as well as different most informative features. However, these features are more divisive in their probabilities of the outcomes. This is

likely due to the smaller sample size. This data could then be applied to the rest of the set of reviews. In my attempt to do this, my computer shut down multiple times so I was unable to yield any results.

ANALYSIS / CONCLUSIONS

From these results we can see that there are clearly linguistic trends in these reviews. There is a clear correlation between the words that a person writes in the review and the perceived sentiment of it. Without any knowledge of the definition of words, we can partition them into sets based on user's uses of them. If we look at the results of the data above we see some interesting developments. For example predicting the exact review score is a much harder task than just simple defining positive and negative sets. That being said, by grouping the exact review scores, we reveal that there is in fact a greater overall trends. Furthermore, by looking at the most informative features of each of the naïve bayes classifications, we glean some interesting and potential useful information about the relative sentiment of English words. For example, as one would predict "maybe" is very correlated with neutral reviews, while "perfect" is very correlated with positive reviews. However, some new information garnered from this is that the word "pampers" is ¾ of the time associated with negative reviews. Each of the words relative probabilities can be seen as the weights that each word places on the sentiment of a review, thus showing in a way we can in fact quantify how positive or negative certain words actually are. This could also be expa

All of this information raises a larger question. Do we understand the emotions behind a sentence in the same way as a computer does? With each of the words carrying a certain weight, pulling how we feel one way or another? Obviously we can read emotions without words, but when are simply just hearing language, it could be argued that human understanding of the

All code can be found at: https://github.com/shulmanbrent/Amazon-Reviews-Semantic-Analysis

meaning of a phrase, the semantics and pragmatics, is don't word by word. Ultimately, we may never know.

All code can be found at: https://github.com/shulmanbrent/Amazon-Reviews-Semantic-Analysis

Bibliography

[1] http://snap.stanford.edu/data/web-Amazon.html

[2] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. RecSys, 2013.

[3] Steven Bird, Ewan Klein. *Natural Language Processing with Python.* Oreilly Media, 2009

All code can be found at: https://github.com/shulmanbrent/Amazon-Reviews-Semantic-Analysis