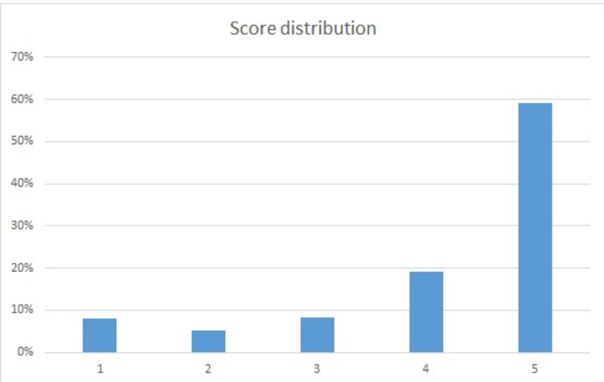
Dan Rozenberg
darozenb@eng.ucsd.edu
A53071493

CSE 255 Assignment 1

1. IDENTIFY DATASET

I will use the Amazon Review database, aquired from http://snap.stanford.edu/data/web-Amazon-links.html. I have downloaded the individual files. On aggregate, there are 35,358,900 reviews. To reduce processing time and need for tighter memory management, I have decided to use only a fraction of this data. More specifically, I am using 0.5% of the total, resulting in over 170 thousand reviews. I will keep the category proportion the same, however, forming the datum with the first 0.5% reviews of each category. During modeling and testing, this reduced data will be separated into training, and validation.

Let's start by looking at the highest level of data. There are 176,779 reviews. The star rating (or star score) is distributed as on



We can see a clear bias towards high scores, specifically, a 5 star rating. This is consistent with Potts' [1] findings. Let's open this up by category and see if the pattern holds.

We can immediately see that the bias holds for all categories, except for office products. Overall, categories do not seem to matter much to the average score distribution.

Product Category	1	2	3	4	5
Amazon_Instant_Video	12%	7%	11%	20%	50%
Arts	9%	6%	8%	13%	64%
Automotive	14%	5%	8%	17%	56%
Baby	13%	6%	7%	16%	57%
Beauty	10%	6%	7%	12%	65%
Books	7%	5%	8%	20%	60%
Cell_Phones_&_Accessories	16%	7%	10%	24%	44%
Clothing_&_Accessories	11%	8%	11%	19%	52%
Electronics	13%	6%	7%	19%	54%
Gourmet_Foods	10%	4%	5%	15%	66%
Health	11%	7%	7%	14%	60%
Home_&_Kitchen	9%	6%	8%	17%	61%
Industrial_&_Scientific	17%	7%	9%	18%	50%
Jewelry	9%	5%	11%	20%	56%
Kindle_Store	11%	10%	14%	21%	43%
Movies_&_TV	10%	6%	9%	20%	55%
Musical_Instruments	6%	2%	6%	22%	65%
Music	5%	4%	7%	18%	67%
Office_Products	35%	10%	8%	14%	34%
Patio	20%	7%	9%	16%	48%
Pet_Supplies	9%	6%	10%	16%	59%
Shoes	3%	3%	8%	22%	64%
Software	22%	10%	9%	18%	41%
Sports_&_Outdoors	9%	5%	6%	21%	59%
Tools_&_Home_Improvement	14%	4%	7%	17%	57%
Toys_&_Games	11%	7%	9%	20%	54%
Video_Games	12%	6%	10%	21%	51%
Watches	9%	5%	7%	24%	55%
Average	12%	6%	8%	18%	55%

If we look at average length per score, we see that scores at the both extremes of the scale are significantly smaller than the ones in the middle. One possible explanation is that such products could have been great or terrible, were it not for a series of reasons, which are explained at greater lengths by the reviewer.

Score	1	2	3	4	5
Characters	660	836	902	871	672

Opening this data up by categories, we see that some types of products have rather lengthier reviews than other, in particular books, movies, videogames and musics, which can be considered more subjective. The distribution of length throughout scores, however, remains mostly unchanged.

Word Length Per Score

Product Category	1	2	3	4	5
Amazon_Instant_Video	749	920	1,048	926	531
Arts	234	656	328	348	415
Automotive	386	375	450	339	337
Baby	437	533	386	355	262
Beauty	396	432	386	498	390
Books	787	921	980	965	791
Cell_Phones_&_Accessories	370	314	405	340	288
Clothing_&_Accessories	384	352	383	390	322
Electronics	576	705	732	632	484
Gourmet_Foods	407	405	438	431	337
Health	345	397	400	447	445
Home_&_Kitchen	469	447	480	452	426
Industrial_&_Scientific	507	672	525	706	474
Jewelry	271	313	273	253	261
Kindle_Store	727	861	1,028	940	652
Movies_&_TV	760	1,020	1,042	1,004	719
Musical_Instruments	732	510	439	745	664
Music	541	793	934	884	664
Office_Products	451	507	697	611	416
Patio	412	385	365	385	366
Pet_Supplies	387	466	458	460	422
Shoes	517	728	509	425	413
Software	619	796	712	959	500
Sports_&_Outdoors	498	411	453	506	389
Tools_&_Home_Improvement	486	607	502	452	423
Toys_&_Games	462	450	443	477	393
Video_Games	648	860	975	841	540
Watches	303	354	343	416	372
Average	495	578	575	578	453

Next, let us look at individual words that compose all of reviews. All together we have 215,494 different words

If we look at the top 5 words, we will see that these are extremely common, neutral words which do not add much information. Looking at the least common words also reveals another issue, many words appear with very small frequency: these are either proper nouns, rare words, or misspelled words. It is clear that this data will need cleaning if we are to use words as predictors.

aggregated top 5	
the	1,285,505
and	696,667
a	604,414
of	592,079

It is reasonable to assume that different scoring reviews will have different phrasing and word usage. As a crude first test I got the 1000 most common words in each score category. For each group of 1000 words, I removed any words that were not exclusive to that group. Looking at the top 20 words yields some interesting and sometimes amusing results:

1-star reviews have monopoly on words such as ”avoid”, ”garbage”, and ”returned”. 2-star reviews have words such as ”confusing”, ”potential” and ”sadly”. 3-star reviews have words like ”fair” and 5-star, words like ”incredible” and ”masterpiece”.

Top Unique Words

1	2	3	4	5
printer	**confusing**	match	451	rice
service	**potential**	constantine	kit	**incredible**
sent	japanese	approach	empire	**thank**
keanu	recent	marriage	plenty	**thanks**
hp	image	**fair**	fahrenheit	1984
phone	thus	personally	fantasy	**masterpiece**
avoid	**considering**	german	witch	**beauty**
crap	asimov	biggest	opening	artist
garbage	sit	terms	crime	sing
junk	suppose	**standard**	introduction	tivo
catholic	telling	presented	members	**favorites**
trash	**sadly**	eye	changes	created
send	let's	places	p	**superb**
weisz	**premise**	odd	stage	jazz
returned	picked	jane	minor	la
total	examples	following	likes	pure
month	**fails**		surprise	bruce
refund	market			hands
customer	alot			**beautifully**

At the same time, we see problems with the methodology: while strongly positive or negative reviews have more unique influential words, things get fuzzy when they get closer to neutrality, since the language gets more nuanced. Some scores did not even have enough unique words (out of 1000) to make a top 20.

Finally, wording seems very dependent on category: printers are the main focus of hatred at Amazon, securing the top unique word for 1-star reviews, (which would explain why office products have such higher incidence of that score). Some very positive words seem to be related to specific books or actors, like ”Bruce” and ”jazz”. Highly praised books, such as George Orwell’s 1984 appear very frequently as 5-star reviews. It could be that a review is an opinion on that book, or is comparing another book to it, as a form of praise.

It is reasonable to assume, therefore, that words have different effects, depending on the product category.

2. IDENTIFY PREDICTIVE TASK

In their recent ”New Avenues in Opinion Mining and Sentiment Analysis” article, Cambria [2] and the others make a distinction between opinion mining and sentiment analysis. According to them, the first relates to polarity detection (positive vs. negative) and the second deals with emotion recognition (such as love, hate, desire, etc...).

Since it is a much simpler and easier problem, I will attempt to perform opinion mining on the Amazon reviews. Although simpler, this problem still could be divided into two categories. The first is simply classifying an opinion as positive/negative/neutral (polarity), while the somewhat more ambitious alternative is to detect the intensity of this opinion—in a scale of 1 to 5, for example.

My main goal in this assignment will be to find the polarity of reviews. The model used does have the ”side-effect” of predicting the of 1 to 5 intensity as well. The evaluation, however, will consider only the polarity for counting successes and failures.

On the matter of evaluation, it will be done in Amazon reviews not seen by the model during training. Whenever the model and the actual data agrees on polarity, this will count as a "hit". When they disagree, it will count as a "miss".

When it comes to comparison baselines, the simplest one would be random chance and naive classification. The first just guesses at random what the polarity should be. The second always chooses the polarity that is majority in the training corpus.

Finally, we can look at the literature for a seeing how the model implemented by me, and other's implementation of this and similar models compare.

3. DESCRIBE LITERATURE

I am using the Amazon review data from SNAP. Previously this data composed a portion of the data used by McAuley and Leskovec [3] in order to find latent product and user dimensions. While my assignment will try to predict a user's rating based on his review text, the aforementioned article tries to predict a user's rating of not yet purchased items. Nonetheless, analysis of the review text is fundamental for McAuley and Leskovec's paper.

Using reviews for opinion mining is nothing new, however. In 2002 Turney [4] used an unsupervised learning algorithm to determine polarity. The main idea was to use adjectives and verbs to estimate semantic orientation (the author offers "romantic ambience" and "horrific events" as examples). The algorithm was run on 440 reviews from Epinion.

Pang [5] used movie reviews from IMDB to train supervised machine learning algorithms in order to predict polarity. They used 2053 reviews from 144 users and tested Naive Bayes, Maximum Entropy classification and Support Vector Machines. The techniques had similar results, but were all better than random.

These early examples will serve as good comparison for the model being implemented in this assignment, due to the similarity of goals and use of relatively simpler models. More recent attempts have more sophisticated goals and techniques.

Jurafsky [6] investigates how to extract the underlying emotions and narratives from Yelp restaurant reviews. One prominent tool was the use of specialized lexicons—domain specific dictionaries that attribute values to common words. They also used logistic regression and, on the statistics side of things, the Monroe [Monroe 2008] method for accounting for variance in words frequencies.

Jo and Oh [7] investigated ways to automatically find out which aspects of an item were being evaluated (and how each aspect was evaluated). Examples of aspects for a digital camera would be its photo quality, brightness of lens, shutter speed and price. One technique used was a modified supervised latent Dirichlet allocation, a probabilistic generative model, which take word positioning into account.

Sooner et al. [8] on the other hand, use a deep learning method (semantic vector spaces), which resembles a tree structure, to try to capture nuances in the phrases, by assigning a positive or negative status to each of the sentence segments. A live demo can be found online at http://nlp.stanford.edu/sentiment/.

As for the future, Multimodal Sentiment Analysis [9] might be one hot area. It involves sentiment analysis aided not only by text, but by audio and video footage of people as well, for example, by analysing Youtube videos.

4. FEATURE SELECTION AND CLEANING

For this task we shall consider the review text to be the main predictor. This is because we expect the user's textual explanation to justify his star-score. However, text itself is not automatically understood by the computer as information, and thus our predictive task has a high degree of natural language processing (NLP).

The world of NLP is bigger than the word of sentiment analysis, and offers a variety of tools. For this assignment, however, I will stick the the more simpler ones, such as the bag-of-words. this was accomplished by transforming each review into a dictionary of word counts indexed by the word itself.

However, for this step we must think of how to split sentences into words. A good candidate is to split at spaces, but sentence ending words are finished with a punctuation sign. If we do not remove punctuations, we will get different tokens that would otherwise mean the same in the bag-of-words model, such as "good", "good." and "good!".

Therefore the first cleaning step was to transform all punctuation (except for the apostrophe) into spaces. Secondly, all words were put into lower case, to avoid duplicate tokens when a token begins a sentence and is capitalized, such as "Good" and "good".

Additionally, I have removed words that, although very common in English, do not add much information by themselves. Since the model I will use mostly ignore semantic structure, removing these words should remove "noise" from the data, while making it smaller at the same time. The list of stop words is in the appendix, and was found at http://www.ranks.nl/stopwords.

Lastly, even though I will use a unigram model, negations could be too important to just throw away. To capture the negative of words, I replaced all instances of negation tokens such as "not ", "isn't ", and "doesn't "(notice the empty space at the end) with "not_". This creates tokens such as "not_good", which, if appear enough time, will have statistical significance in the model. The list of negations is in the appendix.

Further data oddities will be explained in the model section of this report. They weren't removed, but they should not affect the predictions too much.

When we looked at the score distribution over the product category, we saw that, although generally the same, there are certainly some differences. We also saw that top negative and positive words seem to belong to a specific product type. Given this, I will also test a version of the model where the product category is taken into account.

5. MODEL

I will use the Naive Bayes Classifier for this predictive task. This classifier is attractive for its simplicity and intuitive appeal. Even so, it manages to do quite well on this sort of task. Naive Bayes correctness for Turney [4] was on average 74%, vs 59% from guessing the majority class. For Socher et. al [8], Naive Bayes managed 82.6% on the polarity test.

This model is called Naive Bayes because we are naive enough to suppose that a certain word's appearance does not depend on any of the other words in the sentence. In this implementation of the model, each word in the training vocabulary is a separate feature.

If we want to calculate $P(C_k|\vec{x})$, where C is the "document" with class k and

$$\vec{x} = [x_1, x_2, x_3, \cdots, x_n]$$

is the vector of words that appear in C. then we can use Bayes rule to find that it is equal to

$$P(C_k|\vec{x})\frac{P(C_k)P(\vec{x}|C_k)}{P(\vec{x})}$$

. In particular, we can think of each star-score as being a different class $k \in \{1,2,3,4,5\}$. For opinion mining, we want to find which class k has the highest likelihood of being true. In other words, we want to find

$$argmax_k[\frac{P(C_k)P(\vec{x}|C_k)}{P(\vec{x})}]$$

. Under naive assumptions, we have conditional independencies on the (x_i, x_j) pair for all $i \neq j$. Also, we can remove the $P(x)$ term in the denominator, for it is the same regardless of k this leaves us with

$$argmax_k[P(C_k)\prod_{i=1}^{n} P(x_i|C_k)]$$

Now, we can take the *log* of the expression to facilitate calculation, without affecting the $argmax_k$ operation. This becomes

$$argmax_k[log(P(C_k)) + \sum_{i=1}^{n} log(P(x_i|C_k))]$$

$P(C_k)$ is the prior probability of a review having star-score= k. We only need to count the number of reviews of each score and divide for the total number of reviews.

$$P(C_k) = \frac{\text{\# of reviews with score k}}{\text{total \# of reviews in corpus}}$$

However, for calculating $P(x_i|C_k)$ we have a few options. We could count how frequently a word appears in reviews with score k by counting the number of times a given word shows up and divide it by the total count of words in that star-score. However we can also assume that, for a given review, the repetition of words can safely be ignored, and we deal instead with the presence of words. This transforms our model into a *Bernoulli Naive Bayes.*

$$P(x_i|C_k) = \frac{\text{\# of reviews with score k that contain word i}}{\text{\# of reviews with score k}}$$

When these probabilities are calculated, we can begin classifying reviews based on their text. When it came to consider the category into account, the proccess was similar, except that now our number of classes increases to account for all scorecategory combination.

The major advantages of this method is its simplicity and relatively fast execution. Additionally, the training only consists of calculating prior and posterior probabilities. Lastly, rare tokens (such as misspelled words, proper nouns and similar oddities) receive a small weight in classification. There is also no need for a domain-specific lexicon.

The major disadvantages are the lack of any consideration regarding sentence structure and word positioning.

An alternative to my "unigram" approach would be to use "bigrams" or even "n-grams". However, as seen on Pang [Pang 2002], the differences usually are not big.

6. RESULTS

The classifier was trained with random 2/3 of the data corpus. The test data was set to be the remaining 1/3. Whenever a classification was made, it was compared against the actual score. In this assignment, I am trying to predict polarity. Thus, "hits" and "misses" were assigned according to the following table:

	Predicted	hit/miss			
Actual	1	2	3	4	5
1	hit	hit	miss	miss	miss
2	hit	hit	miss	miss	miss
3	miss	miss	hit	miss	miss
4	miss	miss	miss	hit	hit
5	miss	miss	miss	hit	hit

Thus the following results have a small bias towards making errors, since the "neutral" class was defined as being exclusively with score 3.

Accuracy		
	In-Sample	Out-Of-Sample
Simple Bayes	88.3%	73.6%
Category Bayes	94.4%	76.5%

For comparison, the classifier was run both in-sample (on the training data) and out-of-sample. As expected, in-sample predictions were very accurate, with the category-aware classifier doing almost 7% better than the score-only aware classifier. In the actual testing data, the classifier had an accuracy close to 75%. Using the product category increased performance by about 4%. These results are rather better than pure random guesses, and better than naive classifier, which would predicted all scores to be 5, and would yield a 59% of correctness (it would get reviews with score 4 and 5 correctly). This is in line with the aforementioned Turney's [4], but falls short of Socher et. al [8].

	Predicted		Confustion Matrix Without Category		
Actual	1	2	3	4	5
1	975	680	1019	1107	975
2	97	566	799	859	734
3	84	231	1621	1646	1313
4	112	245	1789	6010	3159
5	311	662	3643	15917	14340

Following is the confusion matrix for both classifiers, first score-only, the second is the category-aware matrix.

	Predicted		Confustion Matrix With Category		
Actual	1	2	3	4	5
1	972	437	538	985	1824
2	115	484	466	768	1222
3	101	212	1090	1412	2080
4	118	211	987	4831	5168
5	278	504	2145	11145	20801

Overall the performance was as expected. The boost in performance by using the category information was welcome, but unfortunately rather small. There might have been a gain in performance if bi-grams were used, or if more sophisticated cleaning mechanisms were used.

Finally, the Naive Bayes classifier is a simple, yet effective tool. It's easy to implement, as well as its availability in several programming packages contribute for having this method as a baseline for more complex ones.

APPENDIX

A. REFERENCES

[1] Christopher Potts. On the negativity of negation. In Nan Li and David Lutz, editors, *Proceedings of Semantics and Linguistic Theory 20*, pages 636–659. CLC Publications, Ithaca, NY, 2011.

[2] E. Cambria, B. Schuller, Yunqing Xia, and C. Havasi. New avenues in opinion mining and sentiment analysis. *Intelligent Systems, IEEE*, 28(2):15–21, March 2013.

[3] J. J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Recommender Systems*, 2013.

[4] Peter D. Turney. Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 417–424, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[5] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 79–86, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[6]

[7] Yohan Jo and Alice H. Oh. Aspect and sentiment unification model for online review analysis. In Irwin King, Wolfgang Nejdl, and Hang Li, editors, *WSDM*, pages 815–824. ACM, 2011.

[8] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Stroudsburg, PA, October 2013. Association for Computational Linguistics.

[9] Louis-Philippe Morency, Rada Mihalcea, and Payal Doshi. Towards Multimodal Sentiment Analysis: Harvesting Opinions from The Web. In *International Conference on Multimodal Interfaces (ICMI 2011)*, Alicante, Spain, November 2011.

B. LIST OF STOP WORDS

List of Stop Words					
a	does	how's	other	they	who
about	doesn't	i	ought	they'd	who's
above	doing	i'd	our	they'll	whom
after	don't	i'll	ours	they're	why
again	down	i'm	ourselves	they've	why's
against	during	i've	out	this	with
all	each	if	over	those	won't
am	few	in	own	through	would
an	for	into	same	to	wouldn't
and	from	is	shan't	too	you
any	further	isn't	she	under	you'd
are	had	it	she'd	until	you'll
aren't	hadn't	it's	she'll	up	you're
as	has	its	she's	very	you've
at	hasn't	itself	should	was	your
be	have	let's	shouldn't	wasn't	yours
because	haven't	me	so	we	yourself
been	having	more	some	we'd	yourselves
before	he	most	such	we'll	
being	he'd	mustn't	than	we're	
below	he'll	my	that	we've	
between	he's	myself	that's	were	
both	her	no	the	weren't	
but	here	nor	their	what	
by	here's	not	theirs	what's	
can't	hers	of	them	when	
cannot	herself	off	themselves	when's	
could	him	on	then	where	
couldn't	himself	once	there	where's	
did	his	only	there's	which	
didn't	how	or	these	while	
do					