Problem Chosen	2020 MCM/ICM Summary Sheet	Team Control Number
C		2000936

Summary

In our paper, we mainly made efforts to find and identify key patterns, relationships and measures in past customer supplied ratings and reviews to help Sunshine Company to gain insight into online markets and make market decisions. Models we constructed include one based on entropy theorem and one combining Doc2Vec and T-SNE method. We were able to display text-based information in a two-dimension scale.

First, we used **exploratory data analysis** to test our hypotheses between and among numerical variables and information that can be quantified. We mainly focused on 6 dimensions: star-rating, cumulative rating, length of reviews, helpful votes of reviews, helpful votes/total votes, and popularity (total votes). We got 4 statistically significant conclusions:

I. People tend to write longer views when they are less satisfactory and giving lower stars.

II. The probability of being voted as helpful is greatest at star-rating 1 and lowest at star-rating.

III. Number of usefulness votes increases only when length of review is long enough.

IV. Helpfulness (= helpful votes/total votes) increases at an approximately exponential rate with logarithm of review length.

Second, we continued to introduced time because we noticed that customers' decisions strongly depend on the star and reviews status the time when they made their purchases. Our results suggest a positive correlation between new star-rating and cumulative rating; a higher agreement among customers when cumulative rating being high; and none correlation between popularity and new star-rating.

Next, we established a recommendation model base on modified **entropy method** to extract most informative and readable comments. We chose helpful votes, total votes, verified purchase, vine and logarithm of review length as variables and successfully got the information we wanted. We then conducted sentiment analysis using word frequency, and receive a clear evidence that high-rating reviews are associated with positive quality descriptors, and vice versa.

Finally, we **combined Doc2Vec and T-SNE** (T-distributed stochastic neighbor embedding) to create a model that allows us to represent our text-based information in 2 dimensions, while not losing information such as similarity. Our model is capable of displaying reviews about 3 different products separately. We used our model to display reviews of different stars and saw a clear separation between 5-star reviews and 1-star reviews. Additionally, we saw different distributions on reviews with higher helpful votes and those with lower or no votes. These results represent that customers have a consensus (i.e. they talk about similar things) toward desirable and bad qualities of a specific type of product. We were excited about our results, but subjected to limited time, we were not able to further explore the information lies behind the dissimilarity of vectors.

Team # 2000936
September 3, 2015

Marketing Director of
Sunshine Company

Dear Marketing Director,

We are honored that you trust our team to analyze the online market on Amazon, now we are confident to show you our research result based on the given data.

- Now it's a golden timing to enter the online market for all three products. As you can see in fig. 1, markets for microwave oven, hair dryer and baby pacifier all expanded dramatically in the past 10 years, and we expected a further expansion.
- Although market share of a particular product fluctuates drastically over time, in general, a product, which won a bigger share in the beginning, tends to gain a greater share in the long run. See in fig. 2 So as a product producer, it's of critical importance to become a big player the moment it comes in. Thus, a corresponding product promotion should also be prioritized.

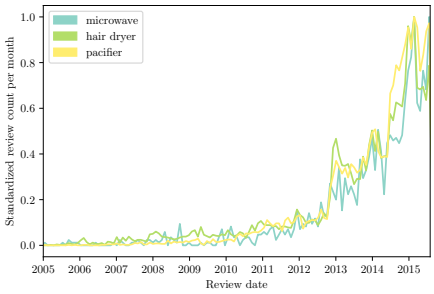


Fig. 1: Total popularity of three product types over time

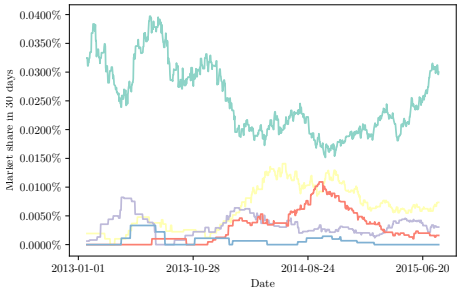


Fig. 2: Different product's market share change over time

- Through our statistical analysis, there is no strong correlation between average star-rating and popularity fig. 3, but the higher your rating is, the bigger the possibility you get another high star-rating (as shown in fig. 4). If we assume that sales volume is in proportion to reputation, which is a combination of popularity (numbers of comments) and rating, then maintaining a high rating is a requirement to win the market.

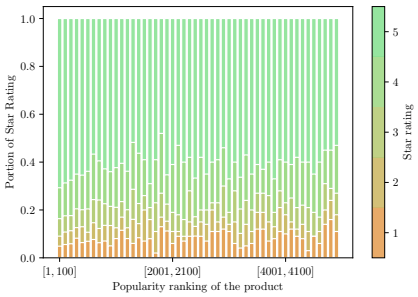


Fig. 3: Cumulative star rating of products with different popularity ranking

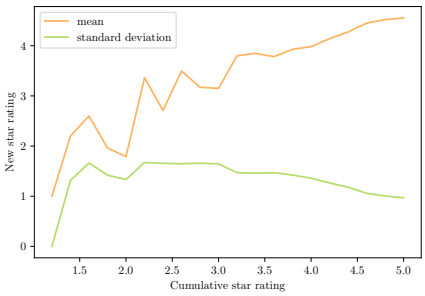


Fig. 4: How cumulative star rating affects a product's new star rating

- While a terrifying volume of reviews might be a strong indication toward success, its impossible to read it by words. We addressed this problem by first established a recommendation model base on entropy theorem. With help of the model, we successfully extracted most informative and readable comments, and through analysis of these texted-based information, we got most desirable features of each type of products as shown in tab. 1.

Microwave	**Hair dryer**	**Pacifier**
simple operation / easy to use	removable filter screen	recognizable
Well-built /constructed	Design	Cute Pattern / entertaining
Quiet / less noise	Easy dry	Last(durable)
Turntable	Buttons are well placed	Not a pacifier alone
Multiple functions / toaster	Work well	easy to wash/sanitation
New buttons (delay, reheat)	Quiet	baby love
Easy installation	Light / small	affordable worth
Smart manual	suitable size	grabbable / don't roll

Tab. 1: Table of desirable features selected through entropy method

- We tested words frequency in reviews, and got a clear evidence that high-rating reviews are associated with positive quality descriptors, and vice versa.

	f_5	f_1	Bias
wonderful	3.25	0.14	23.42
favorite	2.67	0.14	19.26
dries	23.98	1.25	19.19
satisfied	2.52	0.14	18.17
handy	2.37	0.14	17.07
smoother	2.22	0.14	15.98
gives	2.16	0.14	15.54
smooth	6.38	0.42	15.32
silky	2.07	0.14	14.88
love	58.27	4.16	13.99

	f_5	f_1	Bias
refund	0.06	5.00	82.20
dangerous	0.06	4.72	77.63
flames	0.03	2.36	77.63
junk	0.12	5.69	46.81
sparks	0.18	8.33	45.67
fire	0.30	11.24	36.99
2014	0.06	2.08	34.25
return	0.61	20.13	33.11
worst	0.15	4.16	27.40
contacted	0.21	5.00	23.48

Tab. 2: Table of words with biased frequency

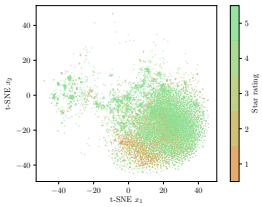


Fig. 5: T-SNE Sentiments of hair dryer reviews

- We combined Doc2Vec and T-SNE (T-distributed stochastic neighbor embedding) to create a model that allows us to represent our text-based information in 2 dimensions, while not losing information such as similarity. Fig. 5 is what we've got regarding hair dryer. We saw a clear separation between 5-star reviews and 1-star reviews, suggesting a strong connection between reviews and their corresponding star-rating. This also represents that customers have a consensus (i.e. they talk about similar things) toward desirable and bad qualities of a specific type of product.

- Once your products are placed online, our modified recommendation model can help track the most informative and readable reviews, thus enabling you to understand the market feedback more efficiently.

We hope that the above information can help you make the decisions.

Best regards,
Team # 2000936

Contents

1 Introdcution **1**
1.1 Problem Statement . . . 1
1.2 Our Goals . . . 1
1.3 Our Thinking . . . 1
1.4 Assumptions . . . 2

2 Data Processing **2**
2.1 Data Explanations . . . 2
2.2 Remove irrelevant Entries . . . 2
2.3 Compute Cumulative Information . . . 3
2.4 Remove Invalid Entries . . . 3
2.5 Preprocess of Text-based data . . . 3

3 Exploratory Data Analysis **3**
3.1 Qualitative Observations on Each Review . . . 3
3.2 Time Pattern Analysis . . . 6
3.3 Entropy Method . . . 9

4 Analysis on Review Texts **10**
4.1 Word Frequency and Sentiments . . . 10
4.2 Document Vectorizing . . . 11
4.3 T-SNE Visualization . . . 11

5 Strengths and Weaknesses **14**
5.1 Strengths . . . 14
5.2 Weaknesses . . . 14

6 Conclusion **15**

1 Introdcution

1.1 Problem Statement

Electronic commerce has become more and more popular in American and around world. Online marketplaces, such as Amazon, allow customers to submit their ratings and reviews towards purchases, which makes it possible for companies to obtain and refine both markets' and customers' demand and take actions toward product requirement management and product design. Therefore, in order to enhance product desirability and gain more market share, many companies think of using these data. One particular exciting angle is to use mathematical and statistical models to look into this booming market.

Now, Sunshine Company is setting to sell three new types of products online: microwave oven, baby pacifier, and hair dryer. Getting customers' remarks toward these three types of products provided by Sunshine's future competitors, we determine to identify the relationships, measures and some patterns in customers' evaluations of the products, and to see whether they will help the company design potentially successful products and track market response once the products are lunched.

1.2 Our Goals

Based on our understanding of the problem, we set the following goals:

- Explore the potential relationships within and between customers' reviews, star ratings and helpfulness votes of their reviews.
- Identify time-based patterns among existing data sets to predict future reputation of a specific product with only months of customers' rating.
- Develop a model based on past data to identify whether a newly coming review will or will not win helpfulness votes in the future.
- Test whether there is a strong connection between sentiment of reviews and star rating, i.e. how they interact with each other in both directions.
- Distinguish most popular features of microwaves, hair dryers and pacifiers based on reviews and ratings.

1.3 Our Thinking

This is a typical big data problem, involving both numerical and text-based information. We relied on basic statistics models and borrow some thoughts from machine learning algorithms to find insights into this problem. Here is our thinking:

First, we preprocessed the data as well as doing exploratory data analysis to identify if there are interesting qualitative correlations between numerical variables and information that can be quantified. Next, we modified our models according to findings in the previous process.

At the same time, after scanning the reviews, we discovered that not all reviews make sense, even those with extremely high votes. So, we decided to view the text-based data as a whole. We

thought of Nature language processing, a branch of machine learning, that can help us to deal with big amount of text-based information. To be specific, we chose a model called **Doc2Vec Model**, which turns each preprocessed review into a fixed-length vector of integers without losing order of words. Through modification of existing process procedures, we were able to extract features of a specific product that customers care about.

In a nut shell, we believe that based on these results, we can help identify potentially valuable information for our client.

1.4 Assumptions

- Reviews of vine customers are more trustworthy then normal customers', and verified purchases are more reliable than unverified purchases.
- We use number of entries to represent popularity of purchases, since there should be a positive correlation between them.
- More than one entry of a same customer with a same date regarding a same product is viewed as invalid. However, when computing the cumulated star rating, we assume the results are based on all existing entries.
- We assume votes of helpfulness reveals the helpful or informative level of a review directly.
- We assume reputation of a product is the combination of the product's cumulative star-rating and it's popularity.

2 Data Processing

2.1 Data Explanations

- Data set definitions and expressions are consistent with the problem set.
- We use **three types of** product to represent microwave, hair dryer and pacifier, while **each product** means products with different product id.
- We use **Entries** to represent each row of data.
- We use **Cumulative Star-rating (Cumulative rating)** to represent the mean star-rating a product had up to but not including the day this entry took place.

2.2 Remove irrelevant Entries

After scanning the raw data, we observed that not all reviews are related to microwaves or pacifiers or hair dryers. So, we immediately removed such entries.

2.3 Compute Cumulative Information

We assume that Amazon's real time star-rating for a particular product is based on all star-ratings in the past, including accidental repetition and intentional repetition. So, before we remove these repetition in further data analysis, we first computed the accumulative star-rating for every product.

2.4 Remove Invalid Entries

Through preliminary observation of our raw data sets, we noticed mainly two type of problematic information. The first one is obvious repetition. We dealt with it by simply deleting entries with same "customer id", "product id" and "review date". The second type is review: "None available." For some reason, there seems to be a lot of reviews being blocked or lost. We delete them when we use content or length information of "review body".

2.5 Preprocess of Text-based data

We deal with text-based data in two different ways. Reviews are meant to be read, so instead of fully use them as computer -reading-data, we process them manually and via computer.

To preprocess the data, we first replaced `"<[^>]*>"` with `""`, then found all emotional expressions, such as `":)"` ,`":D"` and reserved them. We want to delete all meaningless and interfering information while retaining emotions. Next, we replace all abbreviations with original words. Now we have our text in order.

3 Exploratory Data Analysis

3.1 Qualitative Observations on Each Review

The best and quickest way to identify inter-correlation between multidimensional data is to visualize them in certain ways. In this part, we mainly focused on 6 dimensions: star rating, cumulative rating, length of reviews, helpful votes of reviews, helpful votes/total votes, and popularity (total votes).

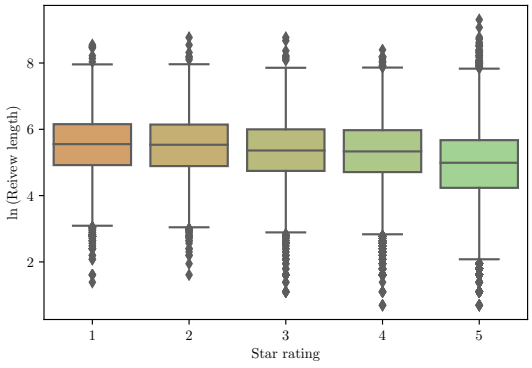


Figure 1: Review lengths of review by star rating

Star-ratings and review length To seek the relationship between the star-ratings and the logarithm of reviews' length, we get fig. 1 on the preceding page. Compare the median of log(length of reviews), we can conclude that people tend to write longer views when they are less satisfactory and giving lower stars. We can also see from the figure that log(length of reviews) can be both the biggest and the smallest when the star-rating is 5. On one hand, it is easier for customers to write longer reviews to show their satisfaction and give a 5 star-rating. On the other hand, it is also easier for customers to give shorter reviews as they do not think it is necessary to type such many words if they give higher ratings(4 or 5 stars).

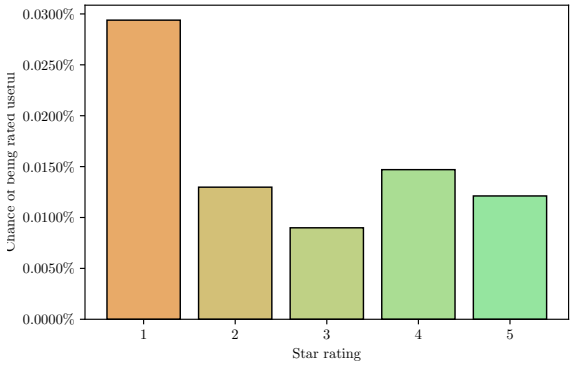


Figure 2: Helpful review frequency by star rating

Helpful votes and star-rating We try to find the relationship between the star-rating and helpful votes. First, we defined helpfulness as proportion of the helpful votes in total votes:

$$\text{helpfulness} = \frac{\text{helpful votes}}{\text{total votes}}.$$

Then, choosing the data which satisfies A total votes > 10 and B helpfulness > 0.9, we calculated the proportion of the data above-mentioned in total votes:

$$P = \frac{\text{votes}(A \wedge B)}{\text{total votes}}.$$

We can get the relationship between star-rating and the probability P.

From fig. 2, we can clearly see that the probability P is the biggest at star-rating 1, which means that the helpful votes satisfying the two conditions outcompete other situations. And P is the smallest at star-rating 3, which means that the helpful votes satisfying the two conditions are the least. We can interpret this result as moderate-level of reviews are much less helpful than specific comments.

Helpful votes and review length Figure 3 on the following page shows the relationship between the review length and helpful votes. We took logarithm scale on both axis to compact the data. We found that it is more likely to get more helpful votes if the review length is longer. But again, review

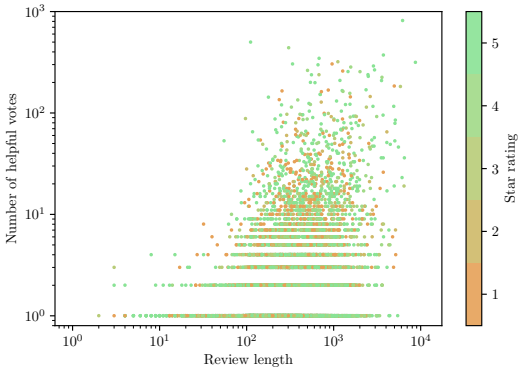


Figure 3: Relation between helpful votes and review length

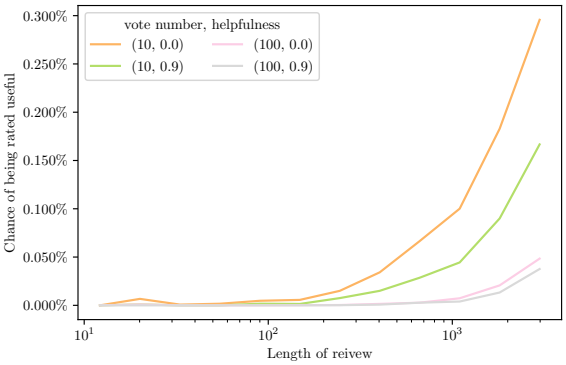


Figure 4: Chance of being "helpful" for different review length

helpful votes are quite random, to make clear this relation, we continued to use the possibility of being helpful, a.k.a. p in section 3.1 on the previous page.

In this way, we displayed in fig. 4 the probability of being recognized as helpful of different length of reviews. We learned that "helpfulness" increases at an approximately exponential rate, regardless of different choice of the standard for being "helpful" (depicted by different colors). Together with fig. 3, we know that there is a strong positive correlation between helpfulness and length of reviews.

3.2 Time Pattern Analysis

Factors influencing change in star rating We know that customers' decisions strongly depend on the star and reviews status the moment they made their purchase. So, in order to dig deep into this probably psychological process, we made fig. 5.

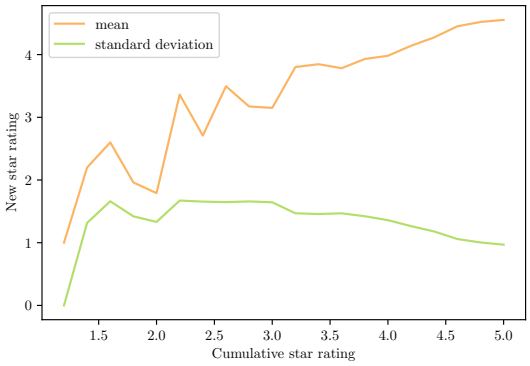


Figure 5: How cumulative star rating affects a product's new star rating

In fig. 5, we compute the expectation and variance of new coming rates on each level of cumulative star-rating. We can see from the figure that, the higher the cumulative star-rating is, the higher the expectation of the newly coming rate. Also, the higher the cumulative star-rating is, the lower the variance of new data is, indicating that when the star-rating of a product is around 5, customers tend to reach a consensus that the product is indeed good, while customers' opinions tend to very when the cumulative rating is relatively low. Note that the sample size of cumulative rating around 1 is quite small, so it make sense that it fluctuates.

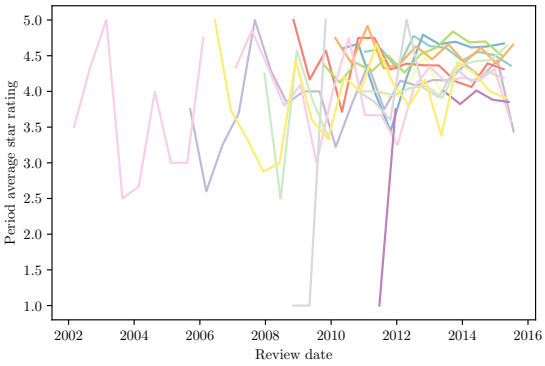


Figure 6: Star rating of different product over time

In another view, fig. 6 shows the fluctuation of an average of 90 days of new star-rating and each color represents one of the 10 most popular products in our data sets. Together with fig. 5, it seems to us that, the new star-rating is a Markova process, a stochastic process whose probability is only related to the real time star rate.

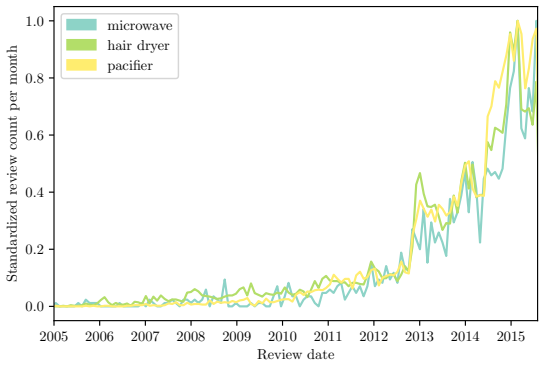


Figure 7: Total popularity of three product types over time

Evaluating popularity with time As stated, we assume that the frequency of being reviewed reflects the popularity of a product. First we checked the star rating's influence on this representation of popularity in fig. 9 on the following page.

So we first plotted the "popularity" of all products by counting the reviews every month. We are surprised to discover that products in three different catagories have really similar overall trend as shown in fig. 7, where each category's data was standardized to illustrate the similarity.

However, we can still see that the monthly review count varies a lot, which is the normal case for a discrete random process. So we tried to smooth data by taking a longer period (90 days) around every review, so that we plotted the popularity more smoothly in fig. 8.

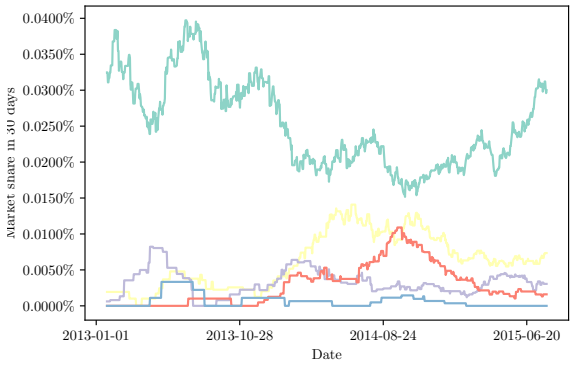


Figure 8: Different product's market share change over time

Moreover, since the overall popularity is increasing and varying greatly, so we introduced market share, which is estimated by $\frac{n_i(T)}{\sum n(T)}$, where n_i is the product's review over some time period T, and $\sum n(T)$ stands for the total review of all products over this period T.

Through plotting market share in fig. 8, we can tell that although a product's market share changes, but the order of magnitude persists.

Factors of popularity change over time We further explore the concept of market share and combines it with star rating, as we show in fig. 9 on the following page.

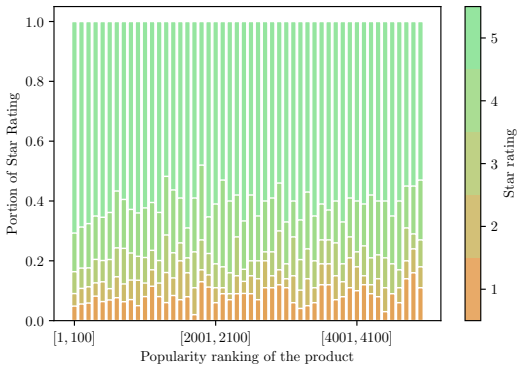


Figure 9: Cumulative star rating of products with different popularity ranking

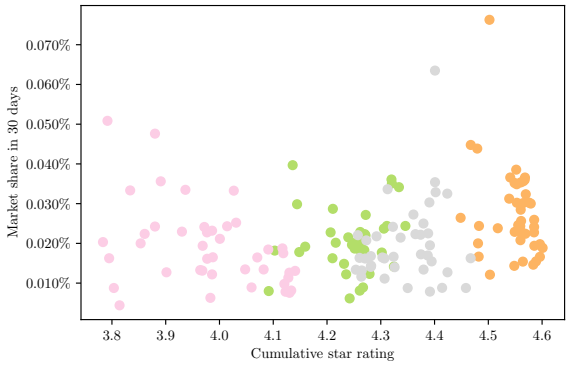


Figure 10: Different product's market share and cumulative star rating in every month

However, surprisingly, the distribution of star rating is clearly random and isn't correlated with the popularity of the product. As verification, we can also see in fig. 10 that different products' market shares vary greatly regardless of their star rating.

3.3 Entropy Method

The Entropy method can help measure the uncertainty of the big data, and entropy can help calculate the randomness and estimate the measures of dispersion. So, we used the modified entropy method to evaluate different indexes that influence the validity of the reviews.

First, we built an original data matrix. We chose different reviews (assume the numbers of the reviews is m), and selected 5 indexes: helpful votes, total votes, vine, verified purchase (assign yes as 1, no as 0), length of review (taken logarithm). Then, we built the matrix $X = \left(x_{ij}\right)_{m\times n}$, where $i = 1 \ldots m$; $j = 1 \ldots n$.

Second, we need to standardize the source data as different indexes reflect different contents. We got the direct indicator by

$$x_{ij}^{\alpha} = \frac{x_{ij} - x_{\min j}}{x_{\max j} - x_{\min j}},$$

where $x_{\max j}$ is the maximum of the jth index and $x_{\min j}$ is the minimum. And then, we calculated the index weight in evaluation index by

$$p_{ij} = \frac{x_{ij}^{\alpha}}{\sum_{i=1}^{m} x_{ij}^{\alpha}}.$$

Third, we tried to calculate the information entropy to measure the disorder of the indexes. The higher the disorder degree is, the smaller the information entropy is. The lower the disorder degree is, the bigger the information entropy is. We got the information entropy of the j-th index by

$$e_j = -k \sum_{i=1}^{m} p_{ij} \ln p_{ij},$$

where $k = 1/\ln m$.

Fourth, we could determine the weight of each index by $\omega_j = \frac{1-e_j}{\sum_{j=1} n\left(1-e_j\right)}$.

Finally, we can calculate the comprehensive evaluation score to see whether the review is valid. We used the formula $s_i = \sum_{j=1}^{n} \omega_j p_{ij}$ and s_i is the final score.

Using this method, we found that the reviews that score high in entropy method are usually more informative than other reviews as shown in table 1.

Helpful V.	Total V.	Verified	Vine	Length	Text	Score
499	575	1	0	111	I didn't know what the attachments were for so...	376.211979
439	451	1	0	304	This is a good dryer while it lasts. Quiet, g...	313.017129
320	332	1	0	1118	This is my only experience with an ionic hair ...	229.513558
315	325	1	0	8701	Let me address the title of the review. From t...	225.519507
304	321	1	0	337	My old Conair died so I read all the reviews a...	219.816417
298	325	1	0	661	I love this dryer. I used a similar one at Pl...	218.959594
304	315	1	0	959	I bought this dryer at Amazon in order to get ...	217.927464
290	293	0	0	2866	I bought the T3 featherweight for my wife as a...	205.459354
266	281	0	0	2951	As a child of the Sixties, I still wear my hai...	192.558397
259	288	0	0	1271	I eagerly snapped up this ionic hair dryer abo...	192.186666

Table 1: Entropy parameters and score on reviews of hair dryers

We have already gotten the score to measure the validity of the existed reviews. Then, we want to know whether the review is valid if it was submitted just now. We replaced "helpful votes" and

"total votes" with their possibility to be voted as helpful based on their corresponding star-rating according figures we get in fig. 2 on page 4.

Then, we repeat the same steps with new indexes. This way, we can quickly find reviews that are most informative and readable the moment they are published online, thus enabling the company to understand the market feedback more efficiently.

4 Analysis on Review Texts

4.1 Word Frequency and Sentiments

We started with calculating and comparing some word frequency $f = n_i/n$, where n_i is the words' times of occurrence, and n is the total word count.

We calculated the frequency for every word separately in 5-star reviews and 1-star reviews. To find words that is more likely to occur in one type of review, we found the quotient of the frequency f in two cases.

	f_5	f_1	Bias
wonderful	3.25	0.14	23.42
favorite	2.67	0.14	19.26
dries	23.98	1.25	19.19
satisfied	2.52	0.14	18.17
handy	2.37	0.14	17.07
smoother	2.22	0.14	15.98
gives	2.16	0.14	15.54
smooth	6.38	0.42	15.32
silky	2.07	0.14	14.88
love	58.27	4.16	13.99

Table 2: Words in review that are highly biased to be positive

The results are shown in table 2 and table 3 on the next page. Good news is, there are quite a few "highly biased" words (where f_1/f_2 are larger). To match for results that make sense, we limited the result to words with $f > 0.002$. Viewing the results we got, we can draw the conclusion that higher rate reviews are related to positive quality descriptors such as wonderful, favorite, satisfied and smoother, silky and etc. in this specific case of hair dryers. But, we can also see that, even in the one-star cases, the word frequency of these descriptors is not zero. Their appearance is certainly not sufficient to say that this review is a five-star review. This correlation is also correct in the opposite situation, where negative descriptor appears much more times in one-star scenarios.

Moreover, some of the words in the reviews can tell us about features of successful products. Therefore, we also decided to spend more time to look into the texts.

	f_5	f_1	Bias
refund	0.06	5.00	82.20
dangerous	0.06	4.72	77.63
flames	0.03	2.36	77.63
junk	0.12	5.69	46.81
sparks	0.18	8.33	45.67
fire	0.30	11.24	36.99
2014	0.06	2.08	34.25
return	0.61	20.13	33.11
worst	0.15	4.16	27.40
contacted	0.21	5.00	23.48

Table 3: Words in review that are highly biased to be negative

4.2 Document Vectorizing

To better understand how reviews influenced customers' behavior, we tried to vectorize the reviews. We combined two existing models, which allow to compact text-based information into a relatively small dimension, in our case 50, and further reduce to 2 dimensions.

We started with mapping documents to a vector space[1]. First, we mapped every stem word to a vector ω_i ($i = 1 \dots m$), and built them into a matrix W.

Second, we took the vectors into the formula:

$$\frac{1}{m}\sum_{t=k}^{m-k} \log p\left(\omega_t \mid \omega_{t-k}, \dots, \omega_{t+k}\right),$$

and then, using softmax to calculate p with the formula

$$p\left(\omega_t \mid \omega_{t-k}, \dots, \omega_{t+k}\right) = \frac{e^{y_{\omega_t}}}{\sum_i e^{y_i}},$$

where y_i is the probability of each word that can be predicted. To get y_i, we have the equation:

$$y = b + uh\left(\omega_t \mid \omega_{t-k}, \dots, \omega_{t+k}; W\right),$$

where b, u are parameters and h is decided by the mean value of $(\omega_t \mid \omega_{t-k}, \dots, \omega_{t+k})$.

After these two steps, we can put the words with similar meanings to similar positions. However, we can not know clearly about the word order.

Third, if there are paragraphs, we need to map each paragraph to a unique vector and build the matrix D, which is used to memorize the topic of the paragraph. Then, repeat the steps above, but we need to take care that h is decided by W and D. Using paragraph vectors and word vectors, we can clearly predict the next word and make the order of the word more accurate.

4.3 T-SNE Visualization

To visualize the high dimensional data, we used another method called T-SNE(T-distributed stochastic neighbor embedding), which allows us to display high dimensional vector $(x_1, x_2, \dots x_n)$ in

a low dimensional space, in particular, a 2-dimension Euclidean plane, while preserving similarity between points. In detail, we first convert the high-dimensional Euclidean distances between datapoints into joint probabilities that represent similarities:

$$p_{ij} = \begin{cases} \frac{\exp -\left\|x_i - x_j\right\|^2 / 2\omega^2}{\sum_{k \neq l} \exp -\|x_k - x_l\|^2 / 2\omega^2}, & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases}$$

While using a Gaussian distribution in high dimension, In the low-dimensional map, we can use t-distribution, a probability distribution that has much heavier tails than a Gaussian to convert distances into probabilities

$$q_{ij} = \frac{\left(1 + \left\|y_i - y_j\right\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}.$$

In this way, we are able to mitigate crowding problem. The gradient function is given in equation

$$\frac{\delta C}{\delta y_i} = 4 \sum_j \left(p_{ij} - q_{ij}\right)\left(y_i - y_j\right)\left(1 + \left\|y_i - y_j\right\|^2\right)^{-1}.$$

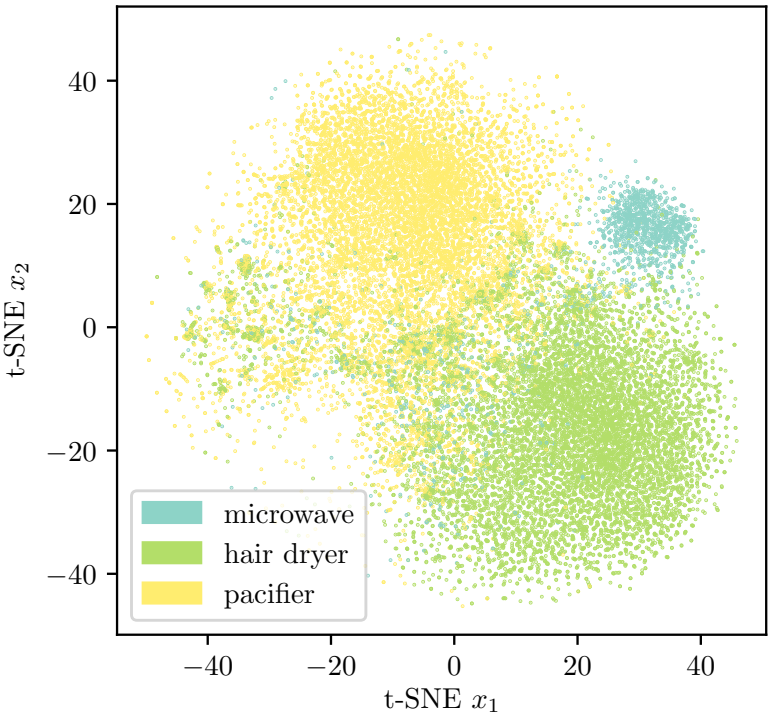
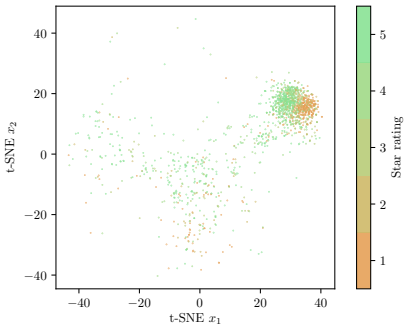


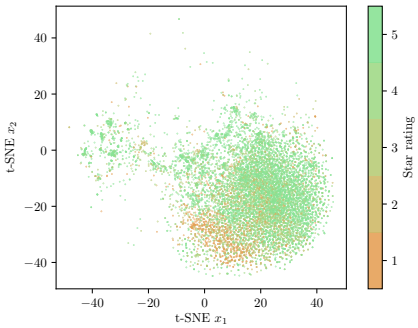
Figure 11: T-SNE visualization of reviewed product types

To test whether this dimension reduction of text-based information makes sense, we tested how reviews regarding the three types of products are displayed in a two-dimensional plane, which is shown in fig. 11 on the preceding page.

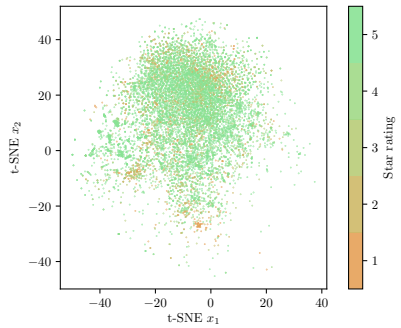
Each color represents one type of products, and despite some outliers, it's clear that each color clustered closely. This result proved that our model is reasonable to some extent. Subject to limited time, we were unable to further optimize our model.



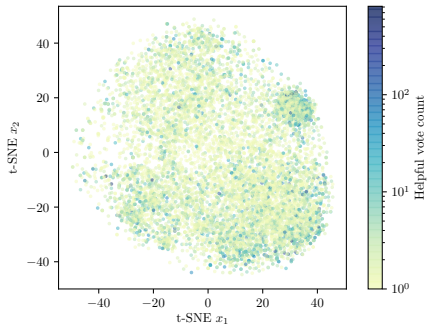
(a) Sentiments of microwave



(b) Sentiments of hair dryer reviews



(c) Sentiments of pacifier reviews



(d) Helpfulness of reviews

Figure 12: T-SNE visualization of review sentiment and helpfulness

Having this model, we tested the relationship between reviews and their corresponding star-rating. Our results on three types of products are shown in fig. 12.

In subfigure (a) and (b), you can easily identify high-rating reviews and low rating reviews. Although the third figure is not as clear as the others, we think this is because most pacifiers received a relatively high star rating.

In subfigure (d), we distinguished reviews that are recognized as "helpful". What is interesting is that it looks like that reviews outside clusters are more likely to be helpful, which may require some more refinement to the model to prove or disprove.

5 Strengths and Weaknesses

5.1 Strengths

- We preprocessed our data throughout our research process, removing all the irrelevant data and invalid data whenever needed. This flexibility helps us to protect the completeness of our data sets while reducing possible disturbances. Our data processing makes it more efficient and convenient for us to sort out different types of the text-based and rating-based measures.
- We used plenty of figures to find and show relationships among indexes directly. All of the figures help us to identify relationships that may exists between two or more indexes and also provide ideas about how to build our models in further analysis.
- We used the entropy method to evaluate how different indexes influence the validity of reviews, which can help us to digitalize the reviews and screen the useful data. Using this method, we gather all the related indexes and choose the data that may contribute to the analysis for further study.
- Using the same method, we can quickly find reviews that are most informative and readable the moment they are published online, thus enabling the company to understand the market feedback more efficiently.
- We combined Doc2Vec and T-SNE to create a model that allows us to represent our text-based information as a 2-dimension vector, while not losing information such as similarity. Getting the results of the model, we can easily test the relationship between reviews and other indexes and draw the figures directly.

5.2 Weaknesses

- Our models and analysis were subjected to limited information. We can only get sufficient information about products with high sales. However, as for those products with low sales, we hardly knew anything about them.
- Because of limited time, we only tested 2 parameters when combining Doc2Vec and T-SNE models, and we were unable to further improve our models.

6 Conclusion

Electronic commerce has been experiencing a rapid growth, so it is important for companies to look inside the markets and decide how they can expand their business and improve products. We receive and analyze the data of three new products: microwave, hair dryer and baby pacifier.

Through the analysis of data, we found there is no strong correlation between average star-rating and popularity, but you are more likely to get another high star-rating if you have higher ratings. Additionally, sales volume could represent reputation to some extent, which is related to popularity (numbers of comments) and ratings. Therefore, gaining a high star-rating is necessary to get more market share. Moreover, we got 4 statistically significant conclusions: I. People tend to write longer views when they are less satisfactory and giving lower stars. II. The probability of being voted as helpful is greatest at star-rating 1 and lowest at star-rating III. Number of usefulness votes increases only when length of review is long enough. IV. Helpfulness increases at an approximately exponential rate with log of review length. As for the time patterns, we can see that the rating tends to be stable when the product has enough reviews.

Also, we made great efforts toward text-based information. By calculating the word frequency, we came to the conclusion that high-rating reviews are associated with positive quality descriptors, and vice versa. Using the entropy method and combining Doc2Vec and T-SNE, we analyzed the content of the reviews and got some figures about the relationships between reviews and other indexes. Our results indicate that customers have a consensus toward desirable and bad qualities of a specific type of product.

In future research, we hope that we can further optimize our models and try to find the information lies behind the dissimilarity of vectorized reviews and get important product features directly from our models. We can also build a more precious model to analyze how reviews with positive (or negative).

References

[1] Le, Q., & Mikolov, T. (2014, January). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188-1196).

[2] Maaten, L. V. D., & Hinton, G. (2008). Visualizing data using T-SNE. *Journal of machine learning research*, 9(Nov), 2579-2605.

List of Figures

1 Review lengths of review by star rating . . . 3
2 Helpful review frequency by star rating . . . 4
3 Relation between helpful votes and review length . . . 5
4 Chance of being "helpful" for different review length . . . 5
5 How cumulative star rating affects a product's new star rating . . . 6
6 Star rating of different product over time . . . 6
7 Total popularity of three product types over time . . . 7
8 Different product's market share change over time . . . 7
9 Cumulative star rating of products with different popularity ranking . . . 8
10 Different product's market share and cumulative star rating in every month . . . 8
11 T-SNE visualization of reviewed product types . . . 12
12 T-SNE visualization of review sentiment and helpfulness . . . 13

List of Tables

1 Entropy parameters and score on reviews of hair dryers . . . 9
2 Words in review that are highly biased to be positive . . . 10
3 Words in review that are highly biased to be negative . . . 11

Python Code

```
1
2 ######################################################################
3
4 import pandas as pd
5 import numpy as np
6 import matplotlib.pyplot as plt
7 import math
8 import seaborn as sns
9 from sklearn.feature_extraction.text import CountVectorizer
10 microwave = pd.read_csv("D:\Studio\mcm2020\Problem_C_Data\microwave.tsv", sep=
    "\t")
11 hair_dryer = pd.read_csv("D:\Studio\mcm2020\Problem_C_Data\hair_dryer.tsv",
    sep="\t")
```

```
pacifier = pd.read_csv("D:\Studio\mcm2020\Problem_C_Data\pacifier.tsv", sep="\t")
tsne = pd.read_csv("D:\Studio\mcm2020\\t_sne.csv")
microwave['product_type'] = microwave['product_title'].str.contains('microwave')*1
hair_dryer['product_type'] = hair_dryer['product_title'].str.contains('hair dryer')*2
pacifier['product_type'] = pacifier['product_title'].str.contains('pacifier')*3
data_all = pd.concat([microwave,hair_dryer,pacifier]).reset_index(drop=True)
data_all = pd.concat([data_all, tsne[['tsne1', 'tsne2']]], axis=1)
data_all['review_date'] = data_all['review_date'].apply(pd.to_datetime)
data_all = data_all.drop_duplicates(subset=['review_date', 'customer_id', 'review_body']);
data_all = data_all.sort_values(by = 'review_date')
data_all['vine']=(data_all['vine'].apply(str.lower)=='y')*1
data_all['verified_purchase']=(data_all['verified_purchase'].apply(str.lower)=='y')*1

data_all.index = pd.Index(data_all['review_id'])
data_all['review_length'] = data_all['review_body'].apply(str).apply(len)
data_all['review_length_log'] = data_all['review_length'].apply(lambda x: math.log(x+1))
data_all['helpful_votes_log'] = data_all['helpful_votes'].apply(lambda x: math.log(x+1))
data_all['total_votes_log'] = data_all['total_votes'].apply(lambda x: math.log(x+1))
data_all['helpfulness']=(data_all['helpful_votes'])/(data_all['total_votes'])
data_all['had_reviews'] = data_all.groupby('product_parent')['star_rating'].transform(lambda s: [i for i in range(len(s))])

def useful_review_count(s):
  ans = [None] * len(s)
  cnt = 0;
  for i in range(len(s)):
      ans[i]=cnt
      if s[i]>10:
          cnt+=1
  return ans

data_all['had_useful_reviews'] = data_all.groupby('product_parent')['helpful_votes'].transform(useful_review_count)

def cumulative_average(s):
    ans = [None] * len(s)
    ans[0] = math.nan
    if len(s) <= 1:
        return ans
    for i in range(len(s)-1):
        ans[i+1]=s[0:i+1].mean()
    return ans

data_all['history_rating'] = data_all.groupby('product_parent')['star_rating'].transform(cumulative_average)
```

```

def recent_average(s):
    n = 10;
    ans = [math.nan] * len(s)
    for i in range(n, len(s)):
        ans[i]=s[i-n:i].mean()
    return ans

data_all['recent_rating'] = data_all.groupby('product_parent')['star_rating'].
    transform(recent_average)

data_all['rating_diff'] = data_all.groupby('product_parent')['history_rating'
    ].transform(lambda x: x.diff().rolling(25).mean())

top_products = data_all.groupby('product_parent').size().sort_values(ascending
    =False)

######################################################################

import matplotlib.colors as colors
import matplotlib.cm as cm
import matplotlib.ticker as mtick
import matplotlib.patches as mpatches

def truncate_colormap9(cmap, minval=0.0, maxval=1.0, n=100):
    new_cmap = colors.LinearSegmentedColormap.from_list(
        'trunc({n},{a:.2f},{b:.2f})'.format(n=cmap.name, a=minval, b=maxval),
        cmap(np.flip(np.linspace(minval, maxval, n)))*0.9)
    return new_cmap

def truncate_colormap(cmap, minval=0.0, maxval=1.0, n=100):
    new_cmap = colors.LinearSegmentedColormap.from_list(
        'trunc({n},{a:.2f},{b:.2f})'.format(n=cmap.name, a=minval, b=maxval),
        cmap(np.linspace(minval, maxval, n)))
    return new_cmap

star_cmap = truncate_colormap9(plt.get_cmap('rainbow'), 0.55, 0.75)
vote_cmap = truncate_colormap(plt.get_cmap('YlGnBu'), 0.1, 1)

from matplotlib import rcParams
rcParams['font.family'] = 'serif'
rcParams['font.serif'] = 'Computer Modern Roman'
rcParams['text.usetex'] = True

def cbar_star():
    fig = plt.gcf()
    bounds = np.arange(.5,6.5)
    ticks = range(1,6)
    norm = colors.BoundaryNorm(bounds, star_cmap.N)
    plt.colorbar(
        cm.ScalarMappable(cmap=star_cmap, norm=norm),
        ticks=ticks,
        label='Star rating',
    )
```

```

product_cmap = plt.get_cmap('Set3')
product_colors = [product_cmap(i/2) for i in range(3)]

######################################################################

sns.boxplot('star_rating', 'review_length_log', data=data_all, palette=
    star_palette(5))
plt.xlabel('Star rating')
plt.ylabel('$\ln\left(\\textrm{Reivew length}\\right)$')
plt.savefig('plots\\length_vs_star.pdf', format='pdf')

######################################################################

d = data_all.groupby('star_rating').apply(lambda df: sum((df['total_votes'
    ]>10) & (df['helpfulness']>.9))/len(df))
plt.bar(d.index, d, color=((pd.Series(d.index)-1)/4).apply(star_cmap),
    edgecolor='black')
plt.xlabel('Star rating')
plt.ylabel('Chance of being rated useful')
plt.gca().yaxis.set_major_formatter(mtick.PercentFormatter())
plt.savefig('plots\\useful_vs_star.pdf', format='pdf')

######################################################################

cmap=star_cmap, data=data_all[~(data_all['review_body']=='None available.')],
    alpha=1, s=2)
plt.xscale('log')
plt.yscale('log')
plt.ylim(.8,1000)
plt.xlabel('Review length')
plt.ylabel('Number of helpful votes')
cbar_star()
plt.savefig('plots\\useful_vs_length.pdf', format='pdf')

######################################################################

step = .5
cut_by = np.arange(2, 8.5, step)
h = [0, .9, 0, .9]
v = [10, 10, 100, 100]
cat_center = np.exp(cut_by[0:len(cut_by)-1] + step)
plt.xscale('log')
d=data_all
for i in range(4):
    g = d.groupby(pd.cut(d["review_length_log"], cut_by)).apply(lambda df: sum
    ((df['total_votes']>v[i]) & (df['helpfulness']>h[i]))/len(df))
    plt.plot(cat_center, g, color=plt.get_cmap('Set3')(i+5))
leg = plt.legend(['(%.0f, %.1f)' % (v[i], h[i]) for i in range(4)], ncol=2)
plt.xlabel('Length of reivew')
plt.ylabel('Chance of being rated useful')
leg.set_title('vote number, helpfulness')
plt.gca().yaxis.set_major_formatter(mtick.PercentFormatter())
plt.savefig('plots\\useful_freq_vs_length.pdf', format='pdf')
```

```
154
155 ######################################################################
156
157 freq = '30D'
158 cut_by = pd.date_range(start='1/1/2012', end='8/1/2015', freq=freq)
159 cut_by = cut_by[0:len(cut_by)-1]
160 for i in range(4):
161     data_p = data_all[data_all['product_parent']==top_products.index[i]]
162     g = data_p.groupby([pd.cut(data_p["review_date"], cut_by), 'product_parent
    ']) 
163     d = pd.merge(g.size().rename('product_count'), data_all.groupby(pd.cut(
    data_all["review_date"], cut_by)).size().rename('total_count'), on='
    review_date')
164     d = pd.merge((d['product_count']/d['total_count']).rename('market_share'),
     g['history_rating'].mean().rename('rating'), on='review_date')
165     plt.scatter('rating', 'market_share', data=d, alpha=1, color=product_cmap(
    i+5))
166 plt.xlabel('Cumulative star rating')
167 plt.ylabel('Market share in 30 days')
168 plt.gca().yaxis.set_major_formatter(mtick.PercentFormatter())
169 plt.savefig('plots\\market_share_vs_cumu_star.pdf', format='pdf')
170
171 ######################################################################
172
173 freq = '1D'
174 cut_by = pd.date_range(start='1/1/2012', end='8/1/2015', freq=freq)
175 dates = cut_by[0:len(cut_by)-1]+pd.Timedelta(freq)/2
176 # cut_by = cut_by[0:len(cut_by)-1]
177 for i in range(5):
178     data_p = data_all[data_all['product_parent']==top_products.index[i*30+1]]
179     g = data_p.groupby([pd.cut(data_p["review_date"], cut_by), 'product_parent
    ']) 
180     d = pd.merge(g.size().rename('product_count'), data_all.groupby(pd.cut(
    data_all["review_date"], cut_by)).size().rename('total_count'), on='
    review_date')
181     d = pd.merge((d['product_count']/d['total_count']).rename('market_share'),
     g['star_rating'].mean().rename('rating'), on='review_date')
182     plt.plot(dates, d['market_share'].rolling(90).mean(), c=product_cmap(i))
183 plt.xlabel('Date')
184 plt.ylabel('Market share in 30 days')
185 plt.gca().yaxis.set_major_formatter(mtick.PercentFormatter())
186 plt.savefig('plots\\market_share_vs_time.pdf', format='pdf')
187
188 ######################################################################
189
190 def standardize(s):
191     return (s-s.min())/(s.max()-s.min())
192
193 top_products = data_all.groupby('product_parent').size().sort_values(ascending
    =False)
194 freq = '30D'
195 for i in range(3):
196     data_p = data_all[data_all['product_type']==i+1]
197     resampled = data_p.resample(freq, on='review_date')
```

```
    plt.plot(resampled.size().index, standardize(resampled.size()), c=
    product_colors[i])
plt.xlim(pd.datetime(2005, 1, 1), pd.datetime(2015, 8, 1))
plt.locator_params(axis='x', nbins=6)
plt.legend(handles=[mpatches.Patch(color=product_colors[i], label=
    product_types[i]) for i in range(3)])
plt.xlabel('Review date')
plt.ylabel('Standardized review count per month')
plt.savefig('plots\\standardized_popularity_vs_time.pdf', format='pdf')

########################################################################

top_pacifiers = pacifier.groupby('product_parent').size().sort_values(
    ascending=False)
def data_product(x):
    return data_all[data_all['product_parent']==top_pacifiers.index[x]]

rng = 50;
step = 100;
x = pd.Series(range(rng));
n =  x.apply(lambda x: sum([len(data_product(i))                    for i in
    range(x*step,(x+1)*step)]))
n4 = x.apply(lambda x: sum([sum(data_product(i)['star_rating']<=4) for i in
    range(x*step,(x+1)*step)]))/n
n3 = x.apply(lambda x: sum([sum(data_product(i)['star_rating']<=3) for i in
    range(x*step,(x+1)*step)]))/n
n2 = x.apply(lambda x: sum([sum(data_product(i)['star_rating']<=2) for i in
    range(x*step,(x+1)*step)]))/n
n1 = x.apply(lambda x: sum([sum(data_product(i)['star_rating']<=1) for i in
    range(x*step,(x+1)*step)]))/n
plt.bar(x, [1]*rng, edgecolor='white', color=star_cmap(4/4))
plt.bar(x, n4     , edgecolor='white', color=star_cmap(3/4))
plt.bar(x, n3     , edgecolor='white', color=star_cmap(2/4))
plt.bar(x, n2     , edgecolor='white', color=star_cmap(1/4))
plt.bar(x, n1     , edgecolor='white', color=star_cmap(0/4))
ax = plt.gca()
plt.locator_params(axis='x', nbins=5)
ax.set_xticklabels(['$[%i,%i]$' % (x*step+1, (x+1)*step) for x in ax.
    get_xticks()])
plt.xlabel('Popularity ranking of the product')
plt.ylabel('Portion of Star Rating')
cbar_star()
plt.savefig('plots\\star_vs_popularity.pdf', format='pdf')

########################################################################

def star_palette(x):
    return sns.color_palette([star_cmap(i/x) for i in range(x)])

product_types = ['microwave', 'hair dryer', 'pacifier']

########################################################################

d=data_all[~(data_all['product_type']==0)]
```

```
plt.scatter(d['tsne1'], d['tsne2'], c=d['product_type'], alpha=1, s=.1, cmap='Set3')
plt.gca().set_aspect(1)
plt.xlabel('T-SNE $x_1$')
plt.ylabel('T-SNE $x_2$')
plt.legend(handles=[mpatches.Patch(color=product_colors[i], label=product_types[i]) for i in range(3)])
plt.savefig('plots\\tsne_product_type.pdf', format='pdf')

######################################################################

d=data_all[(data_all['product_type']==1)]
plt.scatter(d['tsne1'], d['tsne2'], c=d['star_rating'], alpha=1, s=.1, cmap=star_cmap)
plt.gca().set_aspect(1)
plt.xlabel('T-SNE $x_1$')
plt.ylabel('T-SNE $x_2$')
cbar_star()
plt.savefig('plots\\tsne_star1.pdf', format='pdf')

######################################################################

d=data_all[(data_all['product_type']==2)]
plt.scatter(d['tsne1'], d['tsne2'], c=d['star_rating'], alpha=1, s=.1, cmap=star_cmap)
plt.gca().set_aspect(1)
plt.xlabel('T-SNE $x_1$')
plt.ylabel('T-SNE $x_2$')
cbar_star()
plt.savefig('plots\\tsne_star2.pdf', format='pdf')

######################################################################

d=data_all[(data_all['product_type']==3)]
plt.scatter(d['tsne1'], d['tsne2'], c=d['star_rating'], alpha=1, s=.1, cmap=star_cmap)
plt.gca().set_aspect(1)
plt.xlabel('T-SNE $x_1$')
plt.ylabel('T-SNE $x_2$')
cbar_star()
plt.savefig('plots\\tsne_star3.pdf', format='pdf')

######################################################################

d=data_all
plt.scatter(d['tsne1'], d['tsne2'], c=d['helpful_votes'], alpha=.5, s=3, cmap=vote_cmap, norm=colors.LogNorm())
plt.gca().set_aspect(1)
plt.xlabel('T-SNE $x_1$')
plt.ylabel('T-SNE $x_2$')
cbar=plt.colorbar()
cbar.set_label('Helpful vote count')
plt.savefig('plots\\tsne_helpful.pdf', format='pdf')

```

```
######################################################################

step = .2
cut_by = np.arange(1, 5+step, step)
g = data_all.groupby(pd.cut(data_all["history_rating"], cut_by))['star_rating'
    ].mean()
f = data_all.groupby(pd.cut(data_all["history_rating"], cut_by))['star_rating'
    ].std()
cat_center = cut_by[0:len(cut_by)-1] + step
plt.plot(cat_center, g, color = product_cmap(5))
plt.plot(cat_center, f, color = product_cmap(6))
plt.legend(labels=['mean', 'standard deviation'])
plt.xlabel('Cumulative star rating')
plt.ylabel('New star rating')
plt.savefig('plots\\star_vs_history_star.pdf', format='pdf')

######################################################################

for i in range(12):
    data_p = data_all[data_all['product_parent']==top_products.index[i]]
    plt.plot('review_date', 'history_rating', data=data_p, c=product_cmap(i))
plt.xlabel('Review date')
plt.ylabel('Cumulative star rating')
plt.savefig('plots\\cumu_star_vs_date.pdf', format='pdf')

######################################################################

for i in range(12):
    data_p = data_all[data_all['product_parent']==top_products.index[i]].
    resample('180D', on='review_date').mean()
    plt.plot(data_p.index, data_p['star_rating'], c=product_cmap(i))
plt.xlabel('Review date')
plt.ylabel('Period average star rating')
plt.savefig('plots\\star_vs_date.pdf', format='pdf')

######################################################################

import glob
import numpy as np
import pandas as pd
from sklearn.manifold import TSNE
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction import stop_words
import math
import re
import string
import gensim
import random
import nltk
from nltk import tokenize, sent_tokenize
import matplotlib.pyplot as plt

def process_text(text):
```

```
    text = re.sub(r"\\n", " ", text)
    text = re.sub(r"\n", " ", text)
    text = text.lower()
    text = re.sub(r"[^A-Za-z0-9^\/'+-=]", " ", text)
    text = re.sub(r"what's", "what is ", text)
    text = re.sub(r"\'s", " ", text)
    text = re.sub(r"\'ve", " have ", text)
    text = re.sub(r"can't", "can not ", text)
    text = re.sub(r"n't", " not ", text)
    text = re.sub(r"i'm", "i am ", text)
    text = re.sub(r"\'re", " are ", text)
    text = re.sub(r"\'d", " would ", text)
    text = re.sub(r"\'ll", " will ", text)
    text = re.sub(r" e g ", " eg ", text)
    text = re.sub(r" b g ", " bg ", text)
    text = re.sub(r" u s ", " american ", text)
    text = re.sub(r"\0s", "0", text)
    text = re.sub(r" 9 11 ", "911", text)
    text = re.sub(r"e - mail", "email", text)
    text = re.sub(r"j k", "jk", text)
    text = re.sub(r"\s{2,}", " ", text)
    return text

def read_corpus(df, tokens_only=False):
    for i, line in enumerate(df['text']):
        tokens = gensim.utils.simple_preprocess(line)
        if tokens_only:
            yield tokens
        else:
            # For training data, add tags
            yield gensim.models.doc2vec.TaggedDocument(tokens, [i])


df1 = pd.read_csv('microwave.tsv', sep='\t', header=0)

df2 = pd.read_csv('hair_dryer.tsv', sep='\t', header=0)

df3 = pd.read_csv('pacifier.tsv', sep='\t', header=0)

df1 = df1.append(df2)
df1 = df1.append(df3)
df1.reset_index(inplace=True, drop=True)

file = open("vec_df.arr", "rb")
vec_df = np.load(file)
file.close
df1["verified_purchase"] = df1["verified_purchase"].str.lower()
df1["vine"] = df1["vine"].str.lower()
YN_nums = {"verified_purchase": {"y": 1, "n": 0}, "vine": {"y": 1, "n": 0}}
df1.replace(YN_nums, inplace=True)
Y = pd.DataFrame(df1, columns=['verified_purchase', 'product_category', '
    star_rating'])
X_tmp = pd.DataFrame(df1, columns=['product_category', 'star_rating', '
    helpful_votes', 'total_votes', 'vine'])
```

```

X_tmp["product_category"] = X_tmp["product_category"].str.lower()
X = pd.get_dummies(X_tmp, columns=["product_category"], prefix=["pro_cat"])
X = X.to_numpy()
X = np.hstack((X, vec_df))

tsne = TSNE(n_components=2, init='pca', random_state=501)
X_tsne = tsne.fit_transform(vec_df)
df = pd.DataFrame(X_tsne, columns=['tsne1', 'tsne2'])
df4 = pd.concat([df,Y], axis=1, ignore_index=True)
df4.columns = ['tsne1', 'tsne2', 'verified_purchase', 'product_category', '
    star_rating']
df4.to_csv("t_sne.csv")

plt.figure(figsize=(12, 10))
plt.scatter(df4['tsne1'], df4['tsne2'], c=df4["star_rating"], alpha=0.01)
plt.colorbar()
plt.xlabel("tsne1")
plt.ylabel("tsne2")
plt.legend(loc='upper left')
plt.savefig("latent_2dim.png")
plt.show()

######################################################################

def standardize(s):
    return (s-s.min())/(s.max()-s.min())

def evaluation_index(s):
    a = 0.01
    s = standardize(s)
    return (s*(1-a)+a)/s.sum()

def entropy(s):
    m = 1/math.log(len(s))
    p = evaluation_index(s)
    k = 1/math.log(m)
    return -k*sum(p*p.apply(math.log))

def weight(df):
    e = df.apply(entropy)
    return (1-e)/(1-e).sum()

def valid_score(df):
    o = weight(df)
    return df.apply(lambda x: (o*x).sum(), axis=1)

data_all['valid_score'] = valid_score(data_all[['vine', 'helpful_votes', '
    total_votes', 'verified_purchase','review_length_log']])

######################################################################

r = data_all[data_all['product_type']==2].sort_values(by='valid_score',
    ascending=False)[0:10][['helpful_votes', 'total_votes', 'verified_purchase
```

```
    ','review_length', 'vine', 'review_body', 'valid_score']]
with open('plots\hair_dryer_entropy.tex','w') as tf:
    tf.write(r.to_latex())

######################################################################

import re
def preprocessor(text):
    text = re.sub('<[^>]*>', '', str(text))
    emoticons = re.findall('(?::|;|=)(?:-)?(?:\)|\(|D|P)',
    text)
    text = (re.sub('[\W]+', ' ', text.lower()) +
    ' '.join(emoticons).replace('-', ''))
    return text
hair_dryer['helpfulness']=(hair_dryer['helpful_votes'])/(hair_dryer['
    total_votes'])

######################################################################

hair_dryer['helpfulness']=(hair_dryer['helpful_votes'])/(hair_dryer['
    total_votes'])
hair_dryer['review_body_processed'] = hair_dryer['review_body'].apply(
    preprocessor)
common = pd.Series([element for list_ in hair_dryer['review_body_processed']
    for element in list_.split()]).value_counts()
useful = pd.Series([element for list_ in hair_dryer[(hair_dryer['total_votes'
    ]>10) & (hair_dryer['helpfulness']>.9)]['review_body_processed'] for
    element in list_.split()]).value_counts()
useless = pd.Series([element for list_ in hair_dryer[(hair_dryer['helpfulness'
    ]<.5)]['review_body_processed'] for element in list_.split()]).
    value_counts()
love = pd.Series([element for list_ in hair_dryer[hair_dryer['star_rating'
    ]==5]['review_body_processed'] for element in list_.split()]).value_counts
    ()
hate = pd.Series([element for list_ in hair_dryer[hair_dryer['star_rating'
    ]==1]['review_body_processed'] for element in list_.split()]).value_counts
    ()
lvh=pd.merge(love.rename('love')/love.sum(), hate.rename('hate')/hate.sum(),
    how='outer', left_index=True, right_index=True)
lvh['r']=lvh['love']/lvh['hate']
lvh=lvh.sort_values(by='r', ascending=False)
r=lvh[(lvh['love']>.0002)&(lvh['r']>10)][0:10]
with open('plots\goodwords.tex','w') as tf:
    tf.write(r.to_latex())
lvh['r']=lvh['hate']/lvh['love']
lvh=lvh.sort_values(by='r', ascending=False)
r=lvh[lvh['hate']>.0002][0:10]
with open('plots\\badwords.tex','w') as tf:
    tf.write(r.to_latex())

######################################################################
```