Problem Chosen	2020 MCM/ICM Summary Sheet	Team Control Number
C		2012050

Data Analysis on Product Review System: A Comprehensive Evaluation Model

Summary

E-commerce is growing at an unprecedented rate all over the world. In the process of purchasing goods, ratings and reviews play a vital reference role. Companies are pursuing a comprehensive and understandable analysis of online market data to craft greater success.

In this paper, we seek to devise several approaches to analyze the product evaluation by exploring ratings, text-based reviews and other related indicators.

We first perform exploratory data analysis by generating the data quality report and studying the distribution of the most critical indicators. Then we preprocess reviews by a series of steps, including removing punctuations, converting abbreviations, etc. Besides, we extract the frequent features of each product using **WordCloud**.

We then build PRMP, a framework that defines the patterns, relationships, measures, and parameters within and between ratings and reviews. We use **SentiWordNet** to obtain the sentiment of a review and normalize star ratings and helpfulness. Through **association analysis**, we visualize the relationship using a set of heat maps and draw conclusions from them.

After that, we propose a new approach to find a traceable measure for the product. We use **Entropy Weight Method** to obtain weights of the indicators. To identify reputation trends over time, we use the **ARIMA Model** to fit the reputation score. We select one of the hair dryer products as our main study object, calculate its score over time and give its most likely trending result. We use the **Non-linear Programming Model** to find the best combination of text-based and rating-based measures to indicate potential success and failure. We apply the **Hovland Persuasion Model** to build a decision model that describes the indicators that influence customer decisions, achieving the combination of the theory of social psychology and real-life context. And to analyze specific quality descriptors, we classify words into eight categories using the **NRC Emotion Lexicon**. Then we match the emotional intensity with star ratings and find the relationship between them.

Finally, based on the established model and detailed analysis, we put forward practical suggestions for Sunshine Company's marketing plan to improve its product competitiveness.

Contents

1 Introduction

1.1 Background

Globally, more than 50% of e-commerce sales were made through online marketplaces in 2019, contributing $1.7 trillion to the economy each year.[1] E-commerce is growing at an unprecedented rate. Amazon, as the top online marketplace, provides customers with an opportunity to review products and express their level of satisfaction by rating. At the same time, ratings and reviews can reduce potential annoyance on the buying experience.

Companies use these data to analyze the market demand, the timing of market participation, and product optimization. The description and evaluations of products are one of the most valuable references to customers when shopping online. An analysis of ratings and reviews for similar products from Amazon is critical to the company's sales strategies and can help increase product competitiveness.

1.2 Problem Restatement and Our Work

Sunshine Company is planning to introduce and sell three new products in the online marketplace. Therefore, online sales strategy, analysis of product reviews are needed to craft success in the future.

We are provided with data on ratings, reviews for microwave ovens, baby pacifiers and hairdryers sold in Amazon in recent years.

In this paper, we broke our work into sections as follows.

1. We analyze the three data sets thoroughly and generate the data quality report.

2. We preprocess text-based reviews to fit our natural language processing model better.

3. We use SentiWordNet to obtain the sentiment of the review and then find the relationships between and within star ratings, reviews, and helpfulness ratings.

4. Based on the product evaluation on Amazon, we create measures to analyze the reputation of a product and used the ARIMA model to see how it changes over time.

5. We use a non-linear programming model to explore the combinations of dimensions that best indicate a potentially successful or failing product.

6. Applying Hovland Persuasion Model, we perform a decision analysis on motivation for a person to change a review.

7. To analyze specific quality descriptors, we use the NRC Emotion Lexicon to obtain the intensity of eight categories of word sentiment, including anger, fear, trust, etc. Then we match the emotional intensity with star ratings to see if they were somehow related.

8. Finally, based on the established model and detailed analysis, we put forward practical suggestions for Sunshine Company's marketing director to improve their product competitiveness.

2 Assumption and Data Exploration

2.1 Assumption

- The product evaluations are all reasonable. That is, there is no extreme situation like low star rating and great review. We found it rare when checking the dataset.

2.2 Data Quality Report

The data quality report is the most important tool of the data exploration process. The tabular reports are accompanied by data visualizations that illustrate the distribution of the values in each feature in an **ABT**(analytical base table) [1].

Data quality report of **hair_dryer.tsv** is shown as **Table 1**.

Table 1: Analytical Base Table

	customer_id	product_parent	star_rating	helpful_votes	total_votes
Count	11470.00	11470.00	11470	11470	11470
Mean	28151220.00	484633800.00	4.116042	2.179076	2.563296
Std	152387700.00	287324000.00	1.300333	14.241304	15.38258
Min	12464.00	42396.00	1	0	0
Lower Quartile	14914410.00	235106000.00	4	0	0
Media	27071230.00	486774000.00	4	0	0
Upper Quartile	42336440.00	732252300.00	5	1	1
Max	53096370.00	999436600.00	5	499	575

From this report, it is clear that there is no continuous values are missing, so there is no need to fit any missing values. Star_rating is quite skewed and mostly concentrated on the 5. Meanwhile, there are also some extreme values in *'helpful_votes'* and *'total_votes'* that are worth exploring.

All data reports from the three data sets are similar, so we won't go into details here.

2.3 Review Preprocessing

For the categorical features, we selected four main features which are 'vine', 'verified_purchase', 'review_headline' and 'review_headline' as our main targets. The first two are essentially boolean values but 'review_headline' and 'review_body' are long strings and need to be modified.

Here are our preprocessing steps for 'review_headline' and 'review_body':

1. Begin by removing dirty strings in the reviews, such as html tag '
'.

2. Convert English abbreviations to full expressions, such as changing 'isn't' into 'is not'.

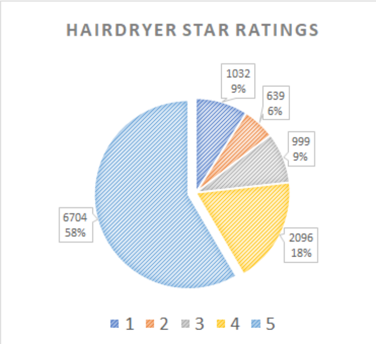
3. Remove any punctuations, alpha-numeric and any other words in the limited set of special characters like '', or '.' or '#' etc.

4. Cut a sentense into seperate words, remove stopwords and convert the remain words to lowercase.

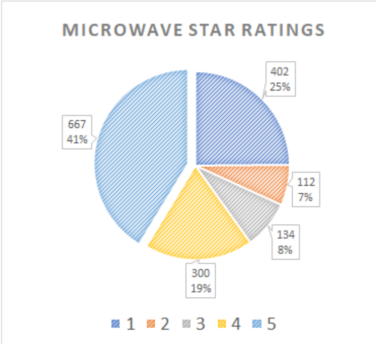
5. When processing review headlines, add a noun after each word in case the part of speech is misjudge .

2.4 Preliminary Insights Into the Data

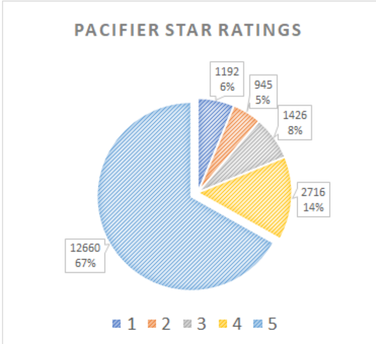
Star Rating. After our statistical analysis, we can find that the 5-star ratings of the three categories of products are all in the majority. But the difference is that the star rating of the microwave oven product is polarized to 1 star and 5 stars. To avoid the potential extremes that may occur in a small number of evaluations, we selected MICROWAVE as a test target for our models.



(a) Star Distribution of Hair dryer



(b) Star Distribution of Microwave



(c) Star Distribution of Pacifier

Figure 1: Star Distribution of Each Product

Review. From the reviews of the three products, we can find the adjective words sifted by WordCloud. The first words that catch the eye are 'great', 'good', 'little' and 'new', then the adjective words related to the specific product. We also generate the feature word cloud of each product in the letter to Sunshine Company.

(a) Review Wordcloud of Hair dryer

(b) Review Worcloud of Microwave

(c) Review Wordcloud of Pacifier

Figure 2: Review Wordcloud of Each Product

3 PRMP Model

The evaluation of a product has a significant impact on whether other customers purchase the product. Star ratings, reviews, and helpfulness ratings are the most valuable references to customers when shopping online, so we consider them as our primary measure factors.

3.1 Review

3.1.1 Sentiment

To analyze customer reviews, we mainly focused on analyzing opinions in the reviews and categorizing the reviews into positive or negative based on customer sentiment.

SentiWordNet [2] is the result of the automatic annotation of all the synsets of WORD-NET according to the notions of “positivity”, “negativity”, and “neutralit”. Each synset s is associated to three numerical scores Pos(s), Neg(s), and Obj(s) which indicate how positive, negative, and “objective” (i.e., neutral) the terms contained in the synset are.[baccianella-etal-2010-sentiwordnet]

We found whether a review is positive or negative in the following steps:

1.) Preprocessing reviews using methods shown in section 2.3 and extract adjectives and coordinating conjunctions, which reveal more emotions and opinions of review.

2.) Using the SentiWordNet to find the positive and negative values related to each word in a review and consider the impact of its neighboring words.

3.) Summing up the obtained positive and negative values to calculate a net positive and net negative values related to a review.

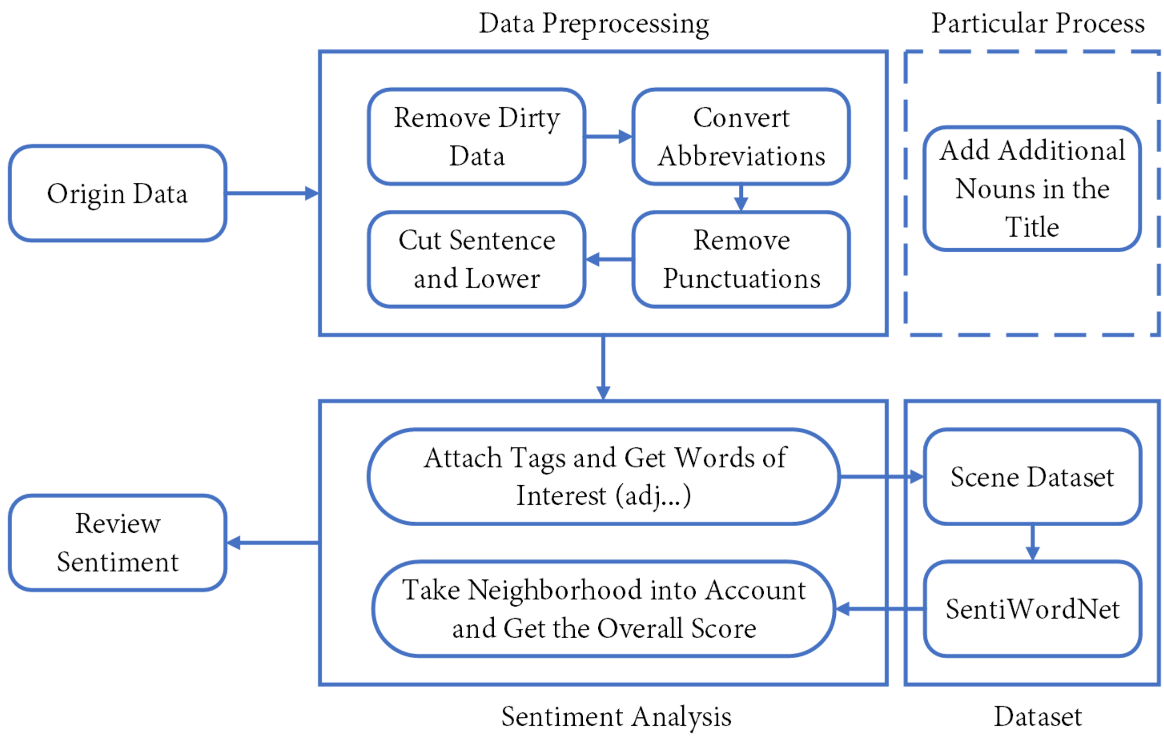


Figure 3: Flowchart of the Sentiment Processing

The sentiment of a review results in [-1, 1], and we normalize it so that the result is between 0 and 1.

3.1.2 Readability and Length

Readability and length of the text are a direct expression of the intelligibility of the review text. In other words, intelligibility is judged by the difficulty of reading and understanding the relevant text.

The more understandable the review is, the more valuable it is. Therefore, intelligibility can be theoreticalized at the cognitive level, based on the cognitive adaptability of the review text to ordinary consumers.

Readability tests have been used to study the quality characteristics of several types of text in different areas of information science, and a number of readability indicators have been developed over the years. We cite the following two quantitative criteria: The Fog index measure complexity, and the ARI index measure reading ease [3].

$$FOG = 0.4 \times (\frac{Words}{Sentence} + 100 \times (\frac{N(complex\_words)}{N(words)})) \tag{1}$$

$$ARI = 4.71 \times (\frac{characters}{words}) + 0.5 \times (\frac{words}{sentence}) - 21.43 \tag{2}$$

Based on these, a quantitative study on the readability of text can be carried out. Due to the limited space of the article, we will not expand the description.

3.2 Star Rating

Star ratings tell people the quality of the product in an intuitive way, which may have an impact on customers' decisions. In general, a moderate star rating(like three stars) shows customers' neutral attitude and gives little information. What customers pay attention to are the extreme star ratings, which can convey more information.

Star ratings are originally a set of integers from 1 to 5. To better fit our model, we performed normalization on it.

Table 2: Star Normalized Value Table

star_ratings	1	2	3	4	5
after normalization	0	0.25	0.5	0.75	1

3.3 Helpfulness

Counting helpful votes is a measure of the quality of a review. With a certain total number of votes, the more helpful votes a review receives from other customers, the more valuable the review is.

We define the **Helpful Votes Ratio** of a review (**HVR**) as the number of helpful votes a review gets divided by the total number of votes.

$$HVR = \frac{helpful\ votes}{total\ votes} \tag{3}$$

If a review doesn't receive any votes at all, we define its HVR as 0.5, which means that this review was neither helpful nor unhelpful.

3.4 PRMP Model and Result

We propose a **PRMP** model that defines the patterns, relationships, measures, and parameters within and between star ratings, reviews, and helpfulness ratings.

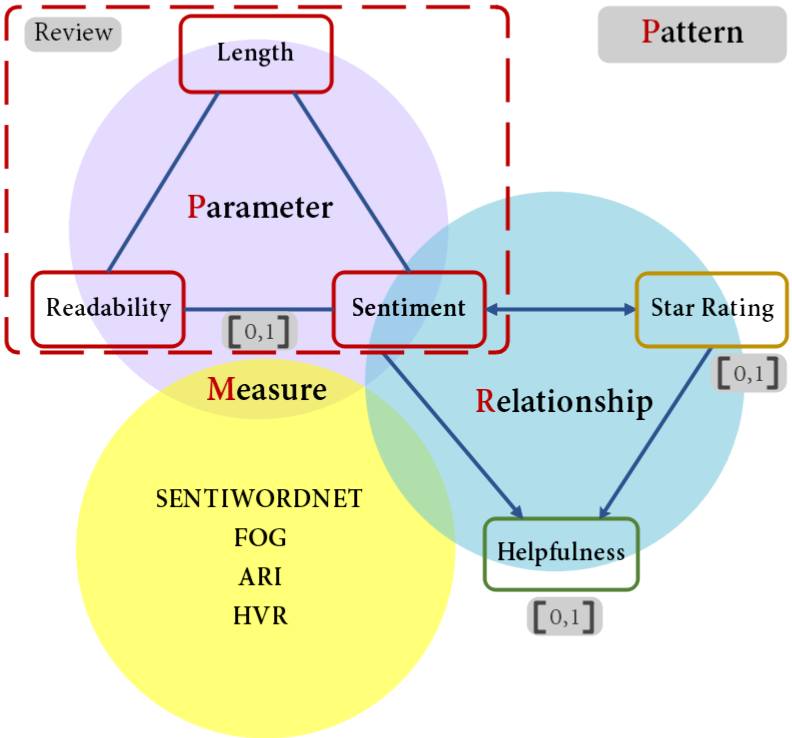


Figure 4: The Concept Graph of PRMP Model

Through the association analysis, we drew the following heat map. Star rating is closely related to the sentiment of a review and has an almost linear correlation.

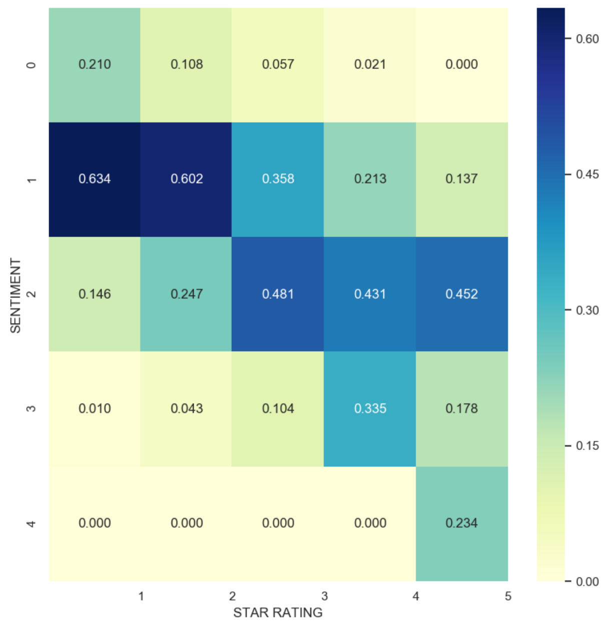
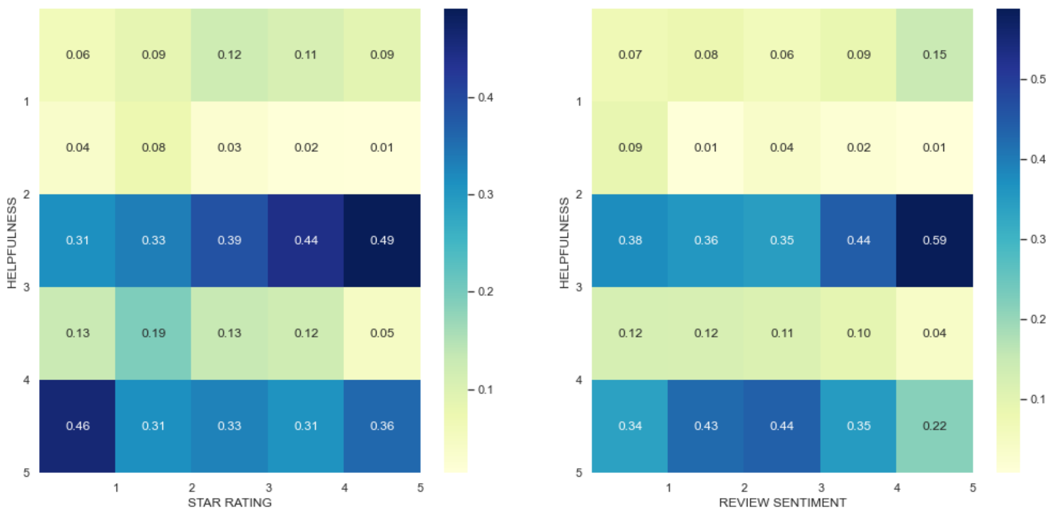


Figure 5: The Correlation Between Sentiment and Star Rating

The dark areas other than the diagonal in the heat map may have something to do with parameter adjustment.

When we performed the association analysis with helpfulness, we got different results. From the two heat maps below, the helpfulness and the other two parameters do not seem to be directly related. However, it is worth noting that in the helpfulness and star rating heat map, 1-star reviews seem to get more support from others than 5-star reviews. Low star reviews may be more helpful because of illustrating the actual problems with the product.



(a) The Correlation Between Helpfullness and Star Rating (b) The Correlation Between Helpfullness and Star Sentiment

Figure 6: The Correlation Between Analysis

Note: the reason why moderate helpfulness is the majority is that most reviews don't have any votes, so HVR equals 0.5.

4 Comprehensive Product Evaluation Model

4.1 Product Measurement

We believe that ratings and reviews give different amounts of information. By calculating the respective information entropy and assigning different weights, we get a score of a single product review.

But are all reviews equally important? Our conclusion is no. The importance of a review is also related to its helpfulness and authority. We used HVR to define the helpfulness and assign different weights for various combinations of 'vine' and 'verified', thus setting the authority.

Considering a star rating and a review can be potentially extreme, we cannot use one evaluation to judge the quality of a product. A short term overall score is needed for Sunshine Company to track.

The figure 7 shows our product measure model .

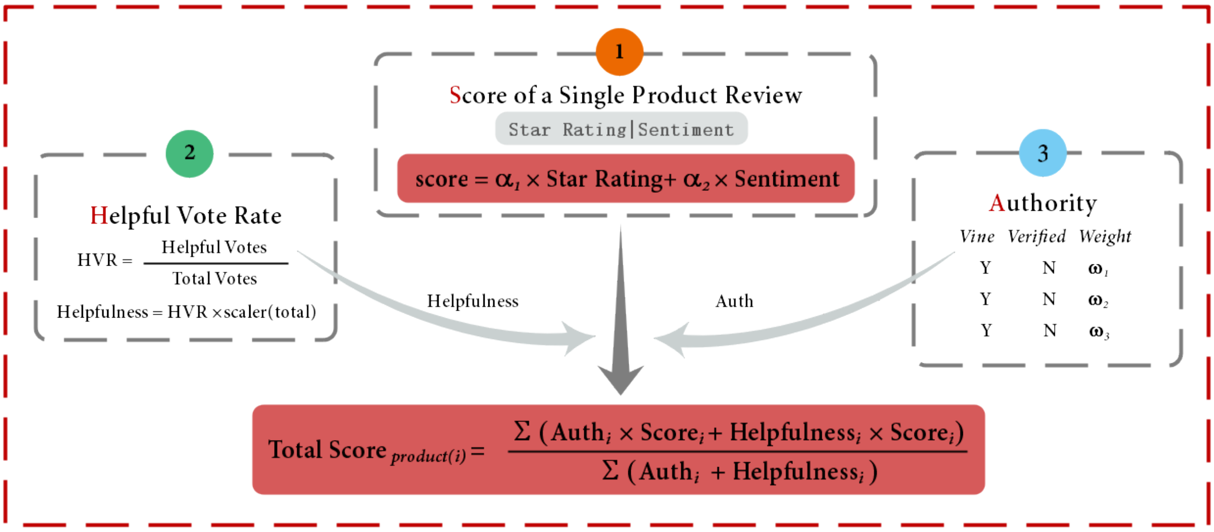


Figure 7: The Concept Graph of Product Measure Model

Since we used indicators with different units, normalization is needed to scale all values in the range [0,1]. Equation4 gives the form of data normalization.

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4}$$

Where x_{max} and x_{min} are the maximum and minimum values of the indicator in the same unit.

4.1.1 Entropy Weight Method to Obtain α_1 and α_2

The weights of star rating and review sentiment are given using the entropy weight method.

According to the definition of information entropy in information theory, the information entropy E_j is equal to:

$$E_j = -\frac{1}{ln(x)} \sum_{i=1}^{n} p_{ij} ln(p_{ij}) \tag{5}$$

According to the calculation formula of information entropy, the information entropy of each indicator is calculated as $E_1, E_2, \cdots, E_k$. From this we can get weights of each indicator:

$$W_j = \frac{1 - E_j}{k - \sum E_j} (i = 1, 2, \cdots, k) \tag{6}$$

4.1.2 Defining ω_1, ω_2, ω_3

Amazon Vine invites the most trusted reviewers on Amazon to post opinions about new and pre-release items to help their fellow customers make informed purchase decisions [4]. Vine Customers by Amazon are credible and professional reviewers, so such reviews are more convincing, and their evaluations are highly authentic. In addition, the rest of the reviewers are divided into two groups, depending on whether they have bought this product on Amazon. We have reason to believe that reviews written by costumers who make a purchase on Amazon are more persuasive than those who don't. So we subjectively consider $\omega 1$ to be 1, ω_2 to be 0.6 and ω_3 to be 0.4.

4.1.3 Scaling HVR

There are cases where there is only one vote and, at the same time being helpfulness vote. It results in a small number of total votes but a high support rate. So we adjust the **HVR** of a review to reduce the extreme impact of a small sample with high support.

We will divide the total number of votes in the short term as the denominator and divide each vote as the scaling number.

Based on the model mentioned above, we have given the following calculation formula to get a scoring formula for measuring a product in the short term.

$$TotalScore_{product(i)} = \frac{\sum(Auth_i \times Score_i + Helpfullness_i \times Score_i)}{\sum(Auth_i + Helpfulness_i)} \tag{7}$$

We calculate the overall score for andis 1875-watt hair dryer, using one month as a short term.

Table 3: The Information of the Specific Product Tested in This Paper

dataset	parent_id	product title
hairdryer.tsv	127343313	andis 1875-watt tourmaline ceramic ionic styling hair dryer

Table 4: The Total Score of Product.127343313

Date	14-Sep	14-Oct	14-Nov	14-Dec	15-Jan	15-Feb	15-Mar
Score	0.777895	0.583396	0.636676	0.559139	0.559205	0.731894	0.692061

4.2 Reputation Trend Model using ARIMA

Definition of the reputation of a company's products: Companies or consumers share various multimedia and comment on information about a product on the Internet. These discussions will affect the credibility of this product. This is the same as 'Internet word of mouth.' We believed that the reputation of a product related to many factors such as star rating, review and helpfulness rating, etc.

And we are looking for data related to the reputation of the product for Sunshine Company. In Section 4.1, we obtained the score of a product over a short period. We believe that the change of its score is related to the rise and fall of reputation, and the dimensions related to reputation are all mentioned in the 4.1 model. We correlate scores with time and build an ARIMA model.

We smooth the original time series. Here we notice that there are zero points in the obtained scores. We use cubic spline interpolation to complete the data, as the figure 8 shows.

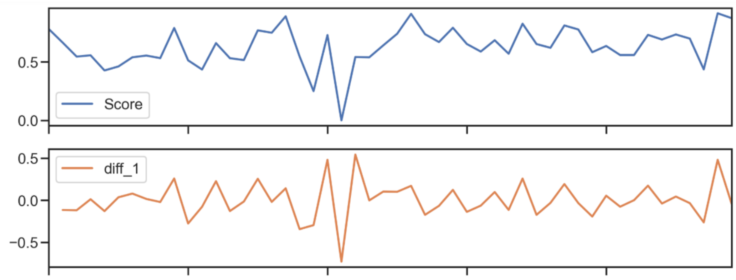


Figure 8: The Sequence of First Difference

The original line chart of the score shows that this is an unstable sequence. So we process first difference method on the sequence. Then we get $p < 0.05$, which means the data passes the test.

We then plot the ACF and PACF of this sequence to select the appropriate p and q of the ARIMA(p, q, d) model. Although there exits an autocorrelation value that exceeds the bounds, we may have p equals 1 due to accidentally exceeding the 95% confidence interval. Using the same method, according to Figure 9, we can get q equals 1.

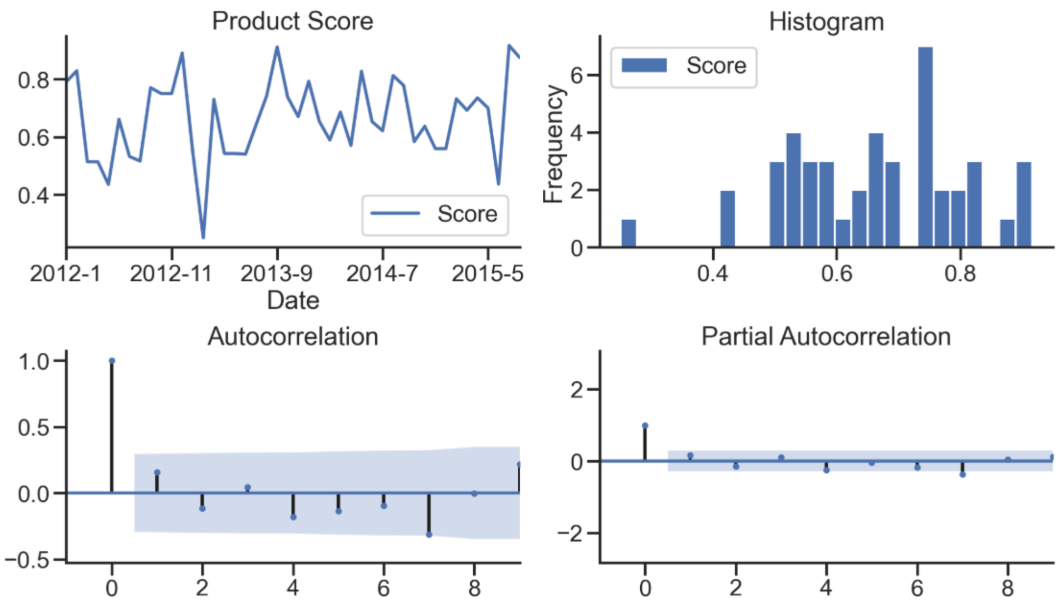


Figure 9: ACF and PACF

We use D-W test method to test this model and find it performs well.

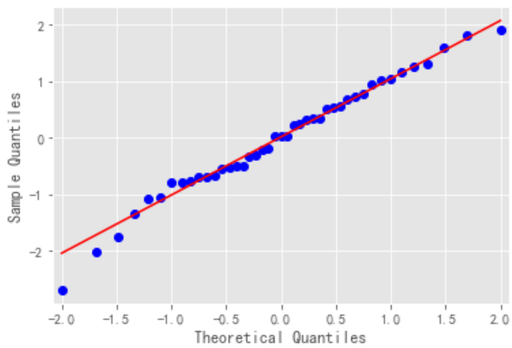


Figure 10: The Result D-W Test

Then we can observe the obvious changes in the data from the figure, and the corresponding changes in the reputation can also be clearly observed because the reputation and the score mentioned above are linked. From the figure shown below, it can be seen that reputation has a greater possibility of increasing.

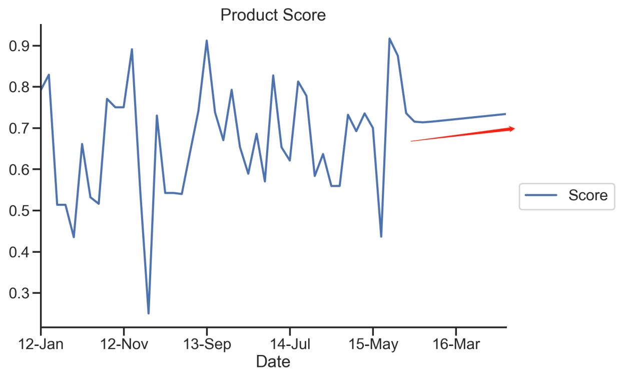


Figure 11: The Trend Generated by ARIMA

4.3 Non-linear Programming Model

As we explained in section 4.2, reputation affects the sales of online products. It means that reputation and its change over time conceal information about the quality of the product sales. Therefore, we map the success or failure of the product to the space of reputation. The base vector of this vector space contains star ratings, review score, helpfulness, etc., and their values range from zero to one.

We use a non-linear programming model to explore the combinations of dimensions that best indicate a potentially successful or failing product.

The model can be described as,

$$\begin{cases} max \quad Total\ Score_{product(i)} = f(star\ rating, review\ score, helpfulness, auth, \alpha) \\ min \quad Total\ Score_{product(i)} = f(star\ rating, review\ score, helpfulness, auth, \alpha) \end{cases}$$

$$\begin{cases} 0 \leq star\ rating \leq 1 \\ 0 \leq sentiment \leq 1 \\ 0 \leq helpfulness \leq 1 \\ auth \in A,\ A = \{\omega_1, \omega_2, \omega_3\} \\ \sum_{j=1}^{2} \alpha_j = 1 \end{cases} \tag{8}$$

The objective function is iterated through the particle swarm optimization algorithm. The pseudo-code of the algorithm is as follows.

Algorithm 1: PSO in NLP

Input: $f(\mathscr{P}), \mathscr{P} \in \mathbb{D}^{star\ rating \times review\ score \times helpfulness \times auth \times \alpha}, it\_max, , c_1, c_2, v$

Output: $Combination(star\ rating, review\ score, helpfulness, auth, \alpha)$

Initialize p=Particle_Initialiazation()

for $i = 1$ to it_max **do**

  for each particle p in $\mathscr{P}$ **do**

    $f_p = f(p)$

  if f_p is better thatf$(pBest)$ **then**

    $pBest = p$

$gBest = best\ pin\mathscr{P}$

for each $particlepin\mathscr{P}$ **do**

  $v = v + c_1 \times rand \times (pBest - p) + c_2 \times rand \times (gBest - p)$ $p = p + v$

After several experiments, we get a pair of combinations.{star rate : 1,review sentiment : 0.75,helpfulness : 1} will best indicate the potentially successful product. While {star rate : 0,review sentiment : 0.5,helpfulness: 0.75} will best indicate the potentially failing product.

4.4 Customer Decision Model

To investigate the causal relationship between star ratings and reviews, we seek inspiration from social psychology. Hovland Persuasion Model is one of the most classic models in this field. The basic model of this approach can be described as "who said what to whom": the source of the communication, the nature of the communication and the nature of the audience [5].

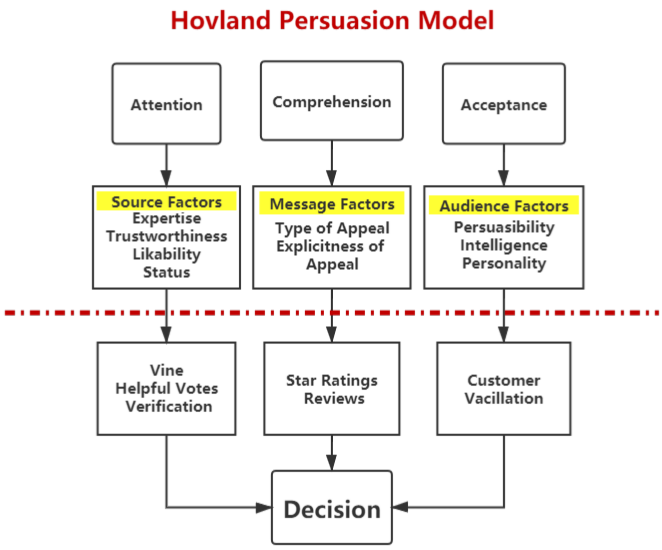


Figure 12: Hovland Persuasion Model Into Practice

4.5 Correlation Between Emotional Descriptors and Star Rating

To analyze specific quality descriptors, we classify all into eight categories using the NRC Emotion Lexicon, including anger, fear, trust, etc. The Sentiment and Emotion Lexicons is a collection of lexicons that was entirely created by the experts of the National Research Council of Canada [6].

We then use the microwave dataset and find the sentiment distribution of all reviews. As the picture shows, the distribution of review headlines and review bodies is similar across all reviews. Words with emotions like anticipation, trust, and joy, make up the majority.

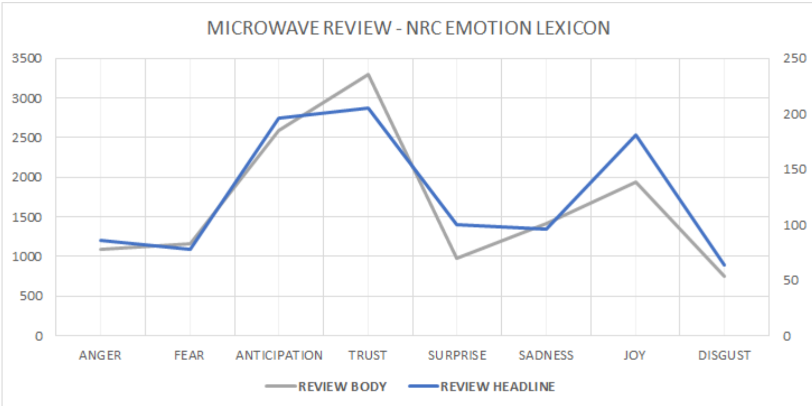
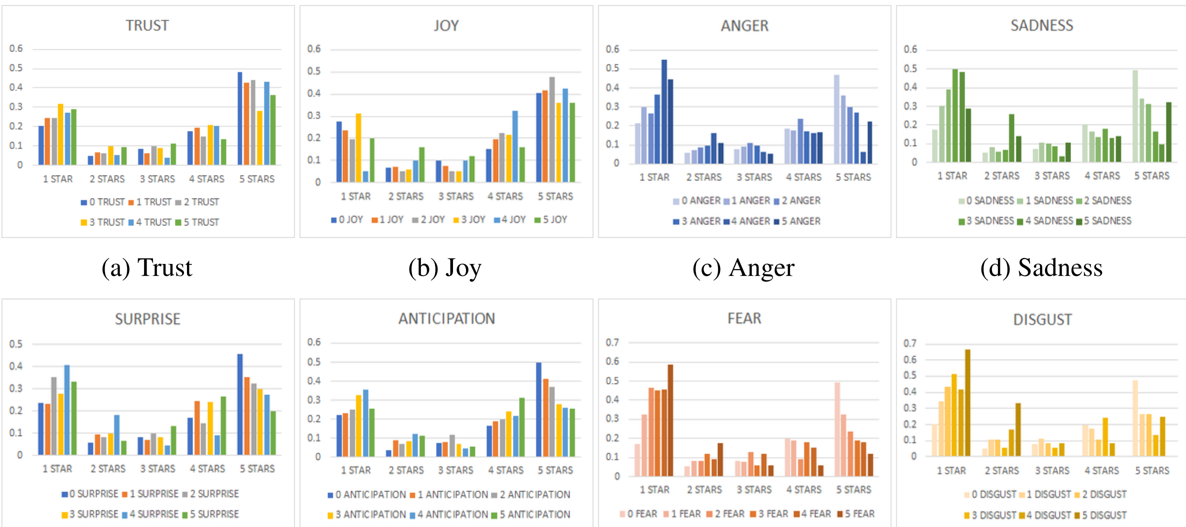


Figure 13: Microwave Review - NRC Emotion Lexicon

Generally speaking, in the bad reviews, customers can describe the expectations of the product, but there will not be too many negative descriptions in the good reviews. Therefore, negative descriptions were the focus of our attention.

We match the emotional intensity of eight categories with star ratings to see if they were somehow related. In the eight tables below, the left and right four show the relationship between the intensity of positive emotions and negative emotions and star ratings, respectively. The bars of each colour represent the frequency of that emotional intensity in each star rating level.



(a) Trust (b) Joy (c) Anger (d) Sadness

(e) Surprise (f) Anticipation (g) Fear (h) Disgust

Figure 14: Emotional Intensity with Each Star Rating Level

In a specific star rating, the darker the color of the histogram is, the stronger the emotion of that star rating is.

In negative emotions, taken ’anger’ as an example, reviews with angry words are more found in low stars ratings. Conversely, reviews that did not or rarely contain angry words were more observed in high star ratings. The same goes for the other three negative emotions.

In positive emotions, the distribution ratio of emotional intensity in star ratings is more uniform, especially in low star rating cases.

5 Sensitivity Analysis

An essential part of our model is the product measure in section 4.1. In our model, the weight (ω_i) is artificially judged based on experience. Changes in those weights may bring different results. We performed the sensitivity analysis by setting various combinations of weights to see the robustness of our model.

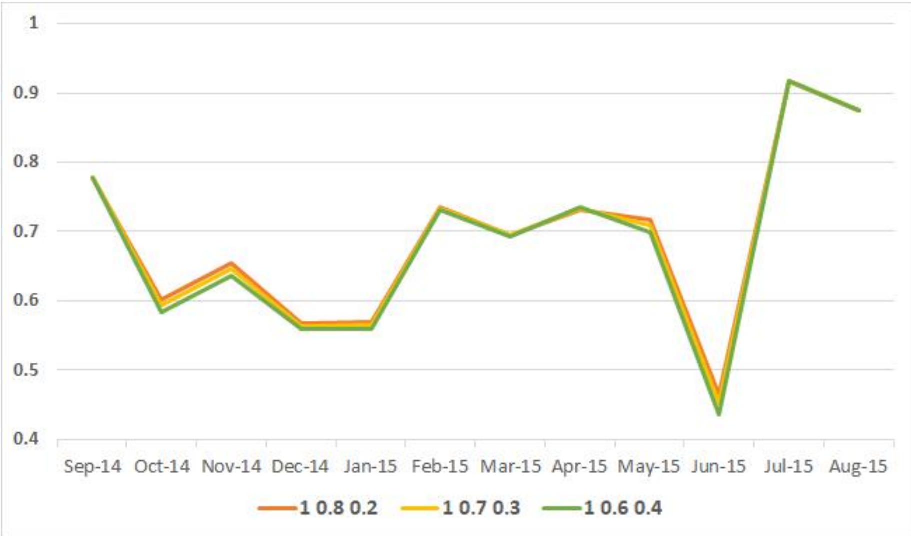


Figure 15: Product.127343313 Total Score

Each of these lines is the product score calculated in a set of given omegas. The green line is the weighted score line we originally designed. As is shown in the figure, when omega changes, the product score is only slightly affected.

6 Strengths and Weaknesses

6.1 Strengths

- We propose a new approach to combine star rating, review and helpfulness rating so that we can quantitatively calculate the overall score of a product and finally integrated them into an evaluation standard.
- We perform detailed and fully preprocessing of the review, and we use two different methods to analyze text-based review, including sentiment analysis using SentiWordNet and emotion analysis using NRC Emotion Lexicon.
- We have applied the Hovland persuasion model to our model, achieving the combination of the theory of social psychology and real-life context.
- The model is universal and can be used in other similar scenarios.
- We make detailed analysis when determining measures and giving suggestions.

6.2 Weaknesses

- In natural language processing, polysemous words such as "right" cannot be classified accurately.
- Misspelling of reviews affects the accuracy of our model.

Letter

To: Sunshine Company Marketing Director
From: Team #2012050
Date: Mar 9th, 2020
Subject: Product Review Analysis and Suggestions

Dear Sunshine Company Marketing Director,

After we analyzed the microwave, pacifier and head dryer products sold on Amazon in recent years, we have the following findings and recommendations.

Analysis: Star ratings show customers the quality of the product in an intuitive way. In the existing market on Amazon, the star ratings of microwave ovens indicate a polarized distribution. Reviews reveal customers' concerns. Helpful votes show the quality of a review. With a certain total number of votes, the more helpful votes a review receives from other customers, the more valuable the review is.

Measure: Taken all three main factors into consideration, we provide you with a measure to track. The total score shows the reputation and of a product in a short period.

Suggestions: For those products you have confidence in, invite Amazon Vine to test. Based on Hovland Persuasion Model, vine customers are the most trusted reviewers, so their reviews have more expertise and trustworthiness for other customers and can even affect their reviews. Not only pay attention to the product's ratings, but also the helpful votes. The most helpful reviews will be shown to more potential customers in the 'top review.' Regularly check the product review, discover the shortcomings of the products proposed by customers, and find the areas that need improvement to upgrade products.

Extra Info: The following are the word clouds for the three product reviews we generated. It is easy to extract the customers' most pressing concerns about the product. For example, for microwave products, customer service and time are the main concerns, which you can improve to increase product competitiveness.

(a) Review Wordcloud of Hair dryer (b) Review Worcloud of Microwave (c) Review Wordcloud of Pacifier

Figure 16: Review Wordcloud of Each Product

The above is the summary of our study. We sincerely hope that it will provide you with valuable information. Thanks!

References

[1] J.D. Kelleher, B.M. Namee, and A. D'Arcy. Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies. The MIT Press. MIT Press, 2015.

[2] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10), Valletta, Malta, May 2010. European Language Resources Association (ELRA).

[3] Nikolaos Korfiatis, Elena García-Bariocanal, and Salvador Sánchez-Alonso. Evaluating content quality and helpfulness of online product reviews: The interplay of review helpfulness vs. review content. Electronic Commerce Research & Applications, 11(3):205—217.

[4] amazon. amazon. Website, 2020. `https://www.amazon.com/gp/vine/help`.

[5] Timothy D. Wilson Aronson, Elliot and Robin M. Akert. Social Psychology. Upper Saddle River, N.J: Pearson Education, 2010.

[6] National Research Council Canada. The sentiment and emotion lexicons. Website, 2016. `http://sentiment.nrc.ca/lexicons-for-research/`.

Appendices

Here are programmes we used in our model.

Analyze the sentiment of the review

```python
import nltk
import re
import tqdm
import pandas as pd
from sklearn import preprocessing

class SentimentAnalysis(object):
    def __init__(self, filename='SentiWordNet.txt'):
        self.swn_pos = {'a': {}, 'v': {}, 'r': {}, 'n': {}}
        self.swn_all = {}
        self.build_swn(filename)
    def geometric_weighted(self, score_list):
        weighted_sum = 0
        num = 1
        for el in score_list:
            weighted_sum += (el * (1 / float(2 ** num)))
            num += 1
        return weighted_sum
    def build_swn(self, filename):
        records = [line.split('\t') for line in open(filename)]
        for rec in records:
            words = rec[4].split()
            pos = rec[0]
            for word_num in words:
                word = word_num.split('#')[0]
                try:
                    sense_num = int(word_num.split('#')[1])
                except:
                    continue
                if word not in self.swn_pos[pos]:
                    self.swn_pos[pos][word] = {}
                self.swn_pos[pos][word][sense_num] = float(
                    rec[2]) - float(rec[3])
                if word not in self.swn_all:
                    self.swn_all[word] = {}
                self.swn_all[word][sense_num] = float(rec[2]) - float(rec[3])
        for pos in self.swn_pos.keys():
            for word in self.swn_pos[pos].keys():
                newlist = [self.swn_pos[pos][word][k] for k in sorted(
                    self.swn_pos[pos][word].keys())]
                self.swn_pos[pos][word] = self.geometric_weighted(newlist)
        for word in self.swn_all.keys():
            newlist = [self.swn_all[word][k] for k in sorted(
                self.swn_all[word].keys())]
            self.swn_all[word] = self.geometric_weighted(newlist)
    def score_word(self, word, pos):
        try:
            return self.swn_pos[pos][word]
        except KeyError:
            try:
                return self.swn_all[word]
            except KeyError:
                return 0
```

```python
    def score(self, sentence):
        impt = {'JJ', 'JJR', 'JJS', 'CC'}
        non_base = {'VBD', 'VBG', 'VBN', 'VBP', 'VBZ', 'NNS', 'NNPS'}
        word_database = pd.read_csv('data/word_database.csv', encoding="utf-8")
        extend_word = word_database['extend_word']
        limited_word = word_database['limited_word']
        negations = {'not', 'n\'t', 'less', 'no', 'never', 'nothing', 'nowhere', 'hardly', 'barely', 'scarcely',
                     'nobody', 'none'}
        stopwords = nltk.corpus.stopwords.words('english')
        wnl = nltk.WordNetLemmatizer()
        scores = []
        tokens = [w.lower() for w in nltk.tokenize.word_tokenize(sentence)]
        tagged = nltk.pos_tag(tokens)
        new_token = []
        new_tag = []
        for word, pos in tagged:
            if (pos not in impt and word not in negations and word.lower() not in extend_word) or word in limited_word:
                continue
            else:
                new_tag.append((word, pos))
                new_token.append(word)
        if len(tagged) == 1:
            return self.score_word(tagged[0][0].lower(), self.pos_short(tagged[0][1])), 1
        index = 0
        for el in tagged:
            pos = el[1]
            try:
                word = re.match('(\w+)', el[0]).group(0).lower()
                start = index - 5
                if start < 0:
                    start = 0
                neighborhood = tokens[start:index]
                if ((pos in impt) and (word not in stopwords) or word in extend_word) and word not in limited_word:
                    if pos in non_base:
                        word = wnl.lemmatize(word, self.pos_short(pos))
                    score = self.score_word(word, self.pos_short(pos))
                    if word == 'if':
                        score -= 0.3
                    if word == 'supposed':
                        score -= 0.2
                    if len(negations.intersection(set(neighborhood))) > 0:
                        r = negations.intersection(set(neighborhood))
                        a = list(r)
                        i_n = tokens.index(a[0])
                        if tagged[i_n + 1][0] in impt or tagged[i_n + 1][0] in extend_word:
                            score = -score
                    scores.append(score)
            except AttributeError:
                pass
            index += 1
        if len(scores) > 0:
            return sum(scores) / float(len(scores)), len(scores)
        else:
            return 0, 0
```

```python


def decontracted(phrase):
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
if __name__ == '__main__':
    s = SentimentAnalysis(filename='SentiWordNet.txt')
    data = pd.read_csv('data/microwave.tsv', sep='\t', header=0, encoding="utf-8")
    review = data['review_body']
    review_title = data['review_headline']
    review_id = data['review_id']
    star = data['star_rating']
    star_dict = {}
    title_reviews = {}
    preprocessed_reviews = {}
    score = {}
    for index in tqdm.tqdm(range(0, data.shape[0])):
        star_dict[review_id[index]] = star[index]
        title_reviews[review_id[index]] = review_title[index]
        try:
            if '<br />' in review[index]:
                tmp = review[index].copy()
                review[index] = tmp.replace('<br />', '')
            without_short = decontracted(review[index])
            without_short1 = decontracted(review_title[index])
            without_number = re.sub("\S*\d\S*", "", without_short).strip()
            without_number1 = re.sub("\S*\d\S*", "", without_short1).strip()
            score1, len1 = s.score(without_number)
            score2, len2 = s.score(without_number1 + " machine")
            if len1 == 0 and len2 == 0:
                continue
            if len2 == 0:
                preprocessed_reviews[review_id[index]] = score1
            elif len1 == 0:
                preprocessed_reviews[review_id[index]] = score2
            else:
                preprocessed_reviews[review_id[index]] = 0.1 * score1 + 0.9 * score2
        except:
            continue
    del preprocessed_reviews[min(preprocessed_reviews, key=preprocessed_reviews.get)]
    del preprocessed_reviews[max(preprocessed_reviews, key=preprocessed_reviews.get)]
    min_value = preprocessed_reviews[min(preprocessed_reviews, key=preprocessed_reviews.get)]
    max_value = preprocessed_reviews[max(preprocessed_reviews, key=preprocessed_reviews.get)]
    ndict = {}
```

```
    nstar = {}
    s = pd.Series(preprocessed_reviews)
    ndf = pd.DataFrame(s, columns=['score'])
    std_scaler = preprocessing.StandardScaler()
    std_label_data = std_scaler.fit_transform(ndf)
    min_max_scaler = preprocessing.MinMaxScaler()
    min_max_label_data = min_max_scaler.fit_transform(std_label_data)
    index = 0
    for key in preprocessed_reviews.keys():
        ndict[key] = min_max_label_data[index][0]
        if 'Five Stars' in title_reviews[key]:
            ndict[key] = 1
        elif 'Four Stars' in title_reviews[key]:
            ndict[key] = 0.75
        elif 'Three Stars' in title_reviews[key]:
            ndict[key] = 0.5
        elif 'Two Stars' in title_reviews[key]:
            ndict[key] = 0.25
        elif 'One Star' in title_reviews[key]:
            ndict[key] = 0
        elif 'zero star' in title_reviews[key]:
            ndict[key] = 0
        index += 1
        nstar[key] = (star_dict[key] - 1) / 4
    key1 = list(ndict.keys())
    value2 = list(ndict.values())
    value3 = list(nstar.values())
    dataframe = pd.DataFrame({'review_id': key1, 'review_score': value2, '
        star_rate': value3, })
    dataframe.to_csv("nor_score2.csv", index=False, sep=',')
```

Calculate the score of a single review

```
import numpy as np
import pandas as pd
import tqdm
import json

class NpEncoder(json.JSONEncoder):
    def default(self, obj):
        if isinstance(obj, np.integer):
            return int(obj)
        elif isinstance(obj, np.floating):
            return float(obj)
        elif isinstance(obj, np.ndarray):
            return obj.tolist()
        else:
            return super(NpEncoder, self).default(obj)

if __name__ == '__main__':
    # dict store origin data
    data_tuple = {}
    # read data
    origin_data = pd.read_csv('data/hair_dryer.tsv', sep='\t', header=0,
        encoding="utf-8")
    part_data = pd.read_csv('data/a1.csv', encoding="utf-8")
    # prepare data
    # get data from source data
    review_id_origin = origin_data['review_id']
    review_id_part = part_data['review_id']
```

```
    # data from origin data
    helpful = origin_data['helpful_votes']
    total = origin_data['total_votes']
    product_id = origin_data['product_parent']
    date = origin_data['review_date']
    vine = origin_data['vine']
    verified = origin_data['verified_purchase']
    # data from part data
    review_score = part_data['review']
    star_score = part_data['star']
    review_id = list(review_id_part.values)
    for index in tqdm.tqdm(range(0, len(origin_data.values))):
        if review_id_origin[index] in review_id:
            date_split = date[index].split('/')
            index_part = review_id.index(review_id_origin[index])
            if date_split[-1] not in data_tuple.keys():
                data_tuple[date_split[-1]] = {date_split[0]: {review_id_origin[index]: (
                    str(product_id[index]), star_score[index_part], review_score[index_part], helpful[index],
                    total[index], vine[index], verified[index], date[index])}}
            else:
                exist_data = data_tuple[date_split[-1]]
                if date_split[0] not in exist_data.keys():
                    exist_data[date_split[0]] = {review_id_origin[index]: (
                        product_id[index], star_score[index_part], review_score[index_part], helpful[index],
                        total[index], vine[index], verified[index], date[index])}
                    data_tuple[date_split[-1]] = exist_data
                else:
                    exist_data[date_split[0]][review_id_origin[index]] = (
                        product_id[index], star_score[index_part], review_score[index_part], helpful[index],
                        total[index],
                        vine[index], verified[index], date[index])
                    data_tuple[date_split[-1]] = exist_data
    data_fre = {}
    for key, value in data_tuple.items():
        month = {}
        for s_key, s_value in value.items():
            tmp_star_dic = {}
            tmp_review_dic = {}
            for t_key, t_value in s_value.items():
                if str(t_value[1]) not in tmp_star_dic.keys():
                    tmp_star_dic[str(t_value[1])] = 1
                else:
                    tmp_star_dic[str(t_value[1])] += 1

                if str(t_value[2]) not in tmp_review_dic.keys():
                    tmp_review_dic[str(t_value[2])] = 1
                else:
                    tmp_review_dic[str(t_value[2])] += 1
            month[s_key] = {'star': tmp_star_dic, 'review': tmp_review_dic}
        data_fre[key] = month
    data_pro = {}
    for key, value in data_fre.items():
        month = {}
        for s_key, s_value in value.items():
```

```
            tmp_star_dic = {}
            tmp_review_dic = {}
            sum_star = 0
            sum_review = 0
            for t_key, t_value in s_value['star'].items():
                sum_star += t_value
            for t_key, t_value in s_value['star'].items():
                tmp_star_dic[t_key] = t_value / sum_star

            for t_key, t_value in s_value['review'].items():
                sum_review += t_value
            for t_key, t_value in s_value['review'].items():
                tmp_review_dic[t_key] = t_value / sum_review
            month[s_key] = {'star': tmp_star_dic, 'review': tmp_review_dic}
        data_pro[key] = month
    e = {}
    for key, value in data_pro.items():
        month = {}
        for s_key, s_value in value.items():
            e_star = 0
            e_review = 0
            for t_key, t_value in s_value['star'].items():
                e_star += t_value * np.log(t_value)
            e_star = (-1.0 / np.log(5)) * e_star
            for t_key, t_value in s_value['review'].items():
                e_review += t_value * np.log(t_value)
            e_review = (-1.0 / np.log(5)) * e_star
            month[s_key] = {'star': e_star, 'review': e_review}
        e[key] = month
    w = {}
    for key, value in e.items():
        month = {}
        for s_key, s_value in value.items():
            e_star = s_value['star']
            e_review = s_value['review']
            w1 = (1 - e_star) / (2 - (e_star + e_review))
            w2 = (1 - e_review) / (2 - (e_star + e_review))
            month[s_key] = {'star': w1, 'review': w2}
        w[key] = month
    # model part 2
    helpful_dic = {}
    for key, value in w.items():
        month = {}
        for s_key, s_value in value.items():
            review_item = {}
            product_total_set = {}
            reviews = data_tuple[key][s_key]
            for t_key, t_value in reviews.items():
                if len(product_total_set) == 0:
                    tmp_total = [t_value[4]]
                    product_total_set[t_value[0]] = tmp_total
                else:
                    if t_value[0] not in product_total_set.keys():
                        tmp_total = [t_value[4]]
                        product_total_set[t_value[0]] = tmp_total
                    else:
                        product_total_set[t_value[0]].append(t_value[4])
            for t_key, t_value in reviews.items():
                if t_value[4] == 0:
                    hvr = 0.5
```

```
                else:
                    hvr = t_value[3] / t_value[4]
                total_list = product_total_set[t_value[0]]
                max_value = max(total_list)
                min_value = min(total_list)
                if (max_value - min_value) == 0:
                    if max_value == 0:
                        scale = 0
                    else:
                        scale = 1
                else:
                    scale = (t_value[4] - min_value) / (max_value - min_value)
                review_item[t_key] = (str(t_value[0]), t_value[1], t_value[2], hvr, scale, t_value[5], t_value[6])
            month[s_key] = review_item
        helpful_dic[key] = month
    # model part 3
    right = [[0.4, 0.6], [1.0, 0]]
    scores = {}
    for key, value in helpful_dic.items():
        month = {}
        for s_key, s_value in value.items():
            score_item = {}
            product_score_set = {}
            product_k_set = {}
            alpha = w[key][s_key]
            for t_key, t_value in s_value.items():
                star_review_score = alpha['star'] * t_value[1] + alpha['review'] * t_value[2]
                hvr_k = t_value[3] * t_value[4]
                if t_value[5] == 'Y':
                    i = 1
                else:
                    i = 0
                if t_value[6] == 'Y':
                    j = 1
                else:
                    j = 0
                authority_k = right[i][j]
                score = authority_k * star_review_score + hvr_k * star_review_score
                k = authority_k + hvr_k
                if len(product_score_set) == 0:
                    product_score_set[t_value[0]] = score
                    product_k_set[t_value[0]] = k
                else:
                    if t_value[0] not in product_score_set.keys():
                        product_score_set[t_value[0]] = score
                        product_k_set[t_value[0]] = k
                    else:
                        product_score_set[t_value[0]] += score
                        product_k_set[t_value[0]] += k
            for t_key, t_value in product_score_set.items():
                score_item[t_key] = t_value / (product_k_set[t_key])
            month[s_key] = score_item
        scores[key] = month
    measure = {}
    for key, value in scores.items():
        for s_key, s_value in value.items():
```

```
            for t_key, t_value in s_value.items():
                if len(measure) == 0:
                    month = {s_key: t_value}
                    year = {key: month}
                    measure[t_key] = year
                else:
                    if t_key not in measure.keys():
                        month = {s_key: t_value}
                        year = {key: month}
                        measure[t_key] = year
                    else:
                        year = measure[t_key]
                        if key not in year.keys():
                            year[key] = {s_key: t_value}
                            measure[t_key] = year
                        else:
                            year[key][s_key] = t_value
                            measure[t_key] = year
    with open('result/product_score_per_month_1.json', 'w') as f:
        json.dump(measure, f, cls=NpEncoder)
```

NRC Emotion Lexicon: categorize the emotion of a word

```
import os
import sys
import json
import nltk
import numpy as np
import pandas as pd
from tqdm import tqdm
from textblob import TextBlob
from nltk.tokenize import sent_tokenize
from nltk.tokenize import TreebankWordTokenizer

data = pd.read_csv('original.csv')
data = pd.DataFrame(data)
data = data[["review_headline", "review_body"]]
nrc_lex = pd.read_csv("NRC-Emotion-Lexicon-Wordlevel-v0.92.txt", sep='\t')
def get_emotion(data, name):
    emotions = []
    for review in tqdm(data[name]):
        total = [0, 0, 0, 0, 0, 0, 0, 0]
        anger = 0
        fear = 0
        anticipation = 0
        trust = 0
        surprise = 0
        sadness = 0
        joy = 0
        disgust = 0
        if not review is np.nan:
            lyrics_text = review
            token_lyrics = sent_tokenize(lyrics_text)
            for sentence in token_lyrics:
                lyric_words = TreebankWordTokenizer().tokenize(sentence)
                for word in lyric_words:
                    anger_list = nrc_lex[nrc_lex['word'] == word][nrc_lex['
                        emotion'] == 'anger'].index.tolist()
                    if len(anger_list) == 1:
                        anger += int(nrc_lex.iloc[int(anger_list[0])]['
```

```
                                association'])
                        fear_list = nrc_lex[nrc_lex['word'] == word][nrc_lex['
                            emotion'] == 'fear'].index.tolist()
                        if len(fear_list) == 1:
                            fear += int(nrc_lex.iloc[int(fear_list[0])]['
                                association'])
                        anticipation_list = nrc_lex[nrc_lex['word'] == word][
                            nrc_lex['emotion'] == 'anticipation'].index.tolist
                                ()
                        if len(anticipation_list) == 1:
                            anticipation += int(nrc_lex.iloc[int(
                                anticipation_list[0])]['association'])
                        trust_list = nrc_lex[nrc_lex['word'] == word][nrc_lex['
                            emotion'] == 'trust'].index.tolist()
                        if len(trust_list) == 1:
                            trust += int(nrc_lex.iloc[int(trust_list[0])]['
                                association'])
                        surprise_list = nrc_lex[nrc_lex['word'] == word][
                            nrc_lex['emotion'] == 'surprise'].index.tolist()
                        if len(surprise_list) == 1:
                            surprise += int(nrc_lex.iloc[int(surprise_list[0])
                                ]['association'])
                        sadness_list = nrc_lex[nrc_lex['word'] == word][nrc_lex
                            ['emotion'] == 'sadness'].index.tolist()
                        if len(sadness_list) == 1:
                            sadness += int(nrc_lex.iloc[int(sadness_list[0])]['
                                association'])
                        joy_list = nrc_lex[nrc_lex['word'] == word][nrc_lex['
                            emotion'] == 'joy'].index.tolist()
                        if len(joy_list) == 1:
                            joy += int(nrc_lex.iloc[int(joy_list[0])]['
                                association'])
                        disgust_list = nrc_lex[nrc_lex['word'] == word][nrc_lex
                            ['emotion'] == 'disgust'].index.tolist()
                        if len(disgust_list) == 1:
                            disgust += int(nrc_lex.iloc[int(disgust_list[0])]['
                                association'])
                total = [anger, fear, anticipation, trust, surprise, sadness,
                    joy, disgust]
                emotions.append(total)
        else:
                emotions.append(total)
    return emotions
def gather(data, emotions):
    result = {"id": [], "anger": [], "fear": [], "anticipation": [], "trust
        ": [], "surprise": [], "sadness": [],
              "joy": [], "disgust": []}
    i = 0
    data = pd.DataFrame(data[["review_id"]])
    print(data)
    for row in data.values:
        result["id"].append(row[0])
        result["anger"].append(emotions[i][0])
        result["fear"].append(emotions[i][1])
        result["anticipation"].append(emotions[i][2])
        result["trust"].append(emotions[i][3])
        result["surprise"].append(emotions[i][4])
        result["sadness"].append(emotions[i][5])
        result["joy"].append(emotions[i][6])
        result["disgust"].append(emotions[i][7])
```

```
            i += 1
        result = pd.DataFrame(result)
        return result
if __name__ == '__main__':
    data_tmp = pd.read_csv('original.csv')
    data_tmp = pd.DataFrame(data_tmp)
    result_body = gather(data_tmp, emotion_body)
    result_body.to_csv('result_body.csv', index=None, sep=',')
    result_title = gather(data_tmp, emotion_title)
    result_title.to_csv('result_title.csv', index=None, sep=',')
    emotion_body = get_emotion(data, 'review_body')
    emotion_title = get_emotion(data, 'review_headline')
    data_tmp = pd.read_csv('original.csv')
    data_tmp = pd.DataFrame(data_tmp)
    result_body = gather(data_tmp, emotion_body)
    result_body.to_csv('result_body.csv', index=None, sep=',')
    result_title = gather(data_tmp, emotion_title)
    result_title.to_csv('result_title.csv', index=None, sep=',')
```