Problem Chosen
C

2020
MCM/ICM
Summary Sheet

Team Control Number
2013868

An MCM Paper Made by Team 2013868

Summary

User review is a crucial component of e-commerce company such as Amazon. How do we automatically summarize millions of user reviews and make sense out of them?

We build a model to identify data measures based on ratings and reviews that are most informative. Our work contains two main systems to calculate the score and weight based on our model and multiple subsystems to help visualize and utilize the result of the model. Our model aims at accurately estimate the reputation of the product by scoring and weighting reviews. What makes our work special is that we designed an evaluation system weighting every review entry to find out if the review entry, or the customer himself, is trustworthy, making sure that every review we take into consideration is reliable. With the subsystems enhancing the readability of the data, the model can easily be understood and deployed by users.

With the model we found the connection between the rating levels and the keywords used by the customers, enabling our system to judge the sentiment of the review and 'predict' the reputation trend of the product. We realized a long-term time-based analyzing system of reputation which can give clients an overall view of the product's reputation and a real-time inflection point alert at the same time. We identified the most informative data our clients need, and our subsystem is able to help clients conclude the popular or bad features from the reviews. We also found that specific star ratings will incite more common reviews, so the clients will be warned to take measures against a potentially dropping reputation.

Keyword: NLP, emotion analysis, customer review, snowball effect, weighted average

Contents

1 Introduction **4**
1.1 Problem Background 4
1.2 Literature Review 4
1.3 Our work 4

2 Preparation of the Models **5**
2.1 Assumptions 5
2.2 Notations 5

3 The Model **5**

4 Solutions **9**
4.1 The First Solution 9
4.2 The Second Solution 9
4.3 The Third Solution 10
4.4 The Forth Solution 10
4.5 The Fifth Solution 11

5 Strengths and Weaknesses **11**
5.1 Strengths 11
5.2 Weaknesses 12

6 Conclusion **12**

References **12**

Appendix **13**
Appendix Sentiment analysis with NLTK 13
Appendix Main System 15

A Letter

Dear the Marketing Director of Sunshine Company:

It's our honor to be hired as consultants to identify key patterns, relationships, measures, and parameters in past customer-supplied ratings and reviews associated with other competing products .We are writing this letter to report our latest findings.

First, we use natural language processing to analyze the data of three product reviews and derive the semantic and emotional keywords along with the sentiment of the reviews.

With the support of our NLP instance, our keywords subsystem is able to collect keywords that match certain requirements. For example, it could show all the keywords that are most frequently used by the customers who like/dislike the product, which can be an evidence indicating that the reputation of the product is going to rise/drop, your team may as a result update the marketing strategy to increase sales or fix the reputation of the product.

As an instance, the following figure are 'word clouds' generated by the keywords subsystem. The keywords are collected from the most negative reviews, serving as the indicators of a coming reputation drop. Your team can monitor the most frequently shown keywords and take measure in advance of the reputation loss.

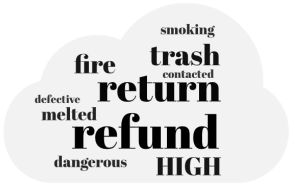

Figure 1: hair_ dryer

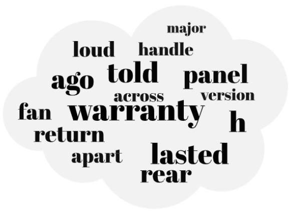

Figure 2: microwave

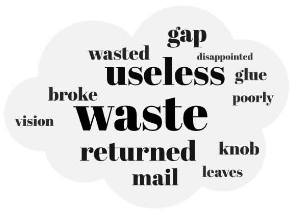

Figure 3: pacifier

In additional to that, our keywords subsystem can also be configured to output only nouns and verbs, showing the most popular/hated points of the product. This indicates potentially important design features that would enhance product desirability. Your team are suggested to advertise the advantages or improve the weakness based on the keywords provided. For example, the word 'warranty' in microwave's word cloud clearly indicates that warranty service is of vital importance and many customers and not satisfied with the current warranty policy. Your team may consider improving and advertising your warranty policy to attract customers to buying your product instead of your competitors'.

We believe that while evaluation of the reviews is of vital importance, it is also crucial to tell if a customer's review is 'reliable', which means whether the customer and the review is trustworthy and objective or a intentionally written lie trying to deceive customers. Our weighting system evaluate every review entry to find out if the review entry, or the customer himself, is 'reliable'. We believe that our weighted reputation score can accurately evaluate the customer feedback of the product, thus indicate a potentially successful or failing product.

With the help of our weighting system, our visualizing subsystem can visualize the 'real-time' and 'accumulated' reputation of a selected model into time-based patterns. The 'real-time' pattern is the weighted average of a certain period, for example, the following figure on the top is the average reputation score of every 30 days. The 'accumulated' pattern is the weighted average of all the scores before the time stamp, as shown the following figure at the bottom.

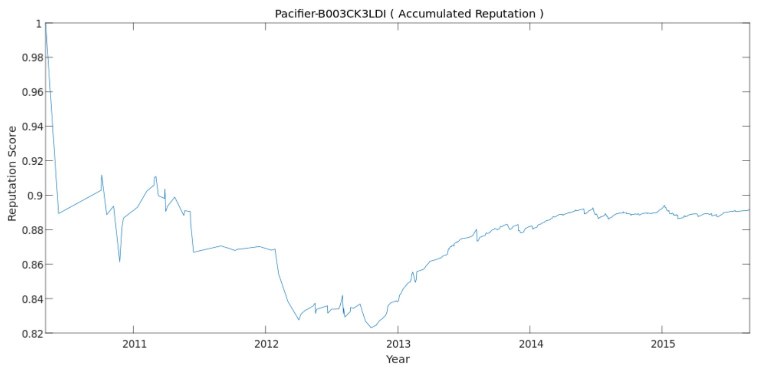

Figure 4: Pacifier-B003CK3LDI(Accumulated Reputation)

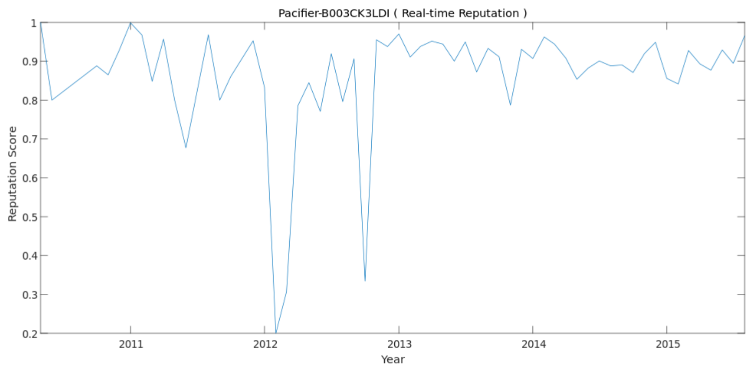

Figure 5: Pacifier-B003CK3LDI(Real-time Reputation)

It is easy to identify the inflection points from the 'accumulated' pattern and with the help of the 'real-time' pattern we can conclude that the sharp drop of the reputation caused by an extremely negative review DID make the customers tend to write their reviews less positively than before. And your team can work out a plan against any upcoming negative reviews to minimize the potential loss.

That's all for our work. We truely hope that our work will help your company have an insight into your competitors and the whole market, as well as help your team develop a more accurate sales strategy and make your product more popular than ever.

Yours sincerely
Team 2013868

1 Introduction

1.1 Problem Background

Product reviews have become an essential part of both electronic and traditional commerce.Users submit feedback on purchases in the form of star ratings and text reviews.The feedback contains a lot of useful information for the merchant, such as improvement suggestions, customer needs, and so on. How to use these three variables, star ratings, reviews, and helpfulness ratings to build a real-time measurement of the reputation of the product, has become an urgent problem.

Five major problems are discussed in this paper, which are:

- Identify data measures based on ratings and reviews that are most informative.
- Identify and discuss time-based measures and patterns within each data set.
- Determine combinations of text-based measure(s) and ratings-based measures that best indicate a potentially successful or failing product.
- Do specific star ratings incite more reviews?
- Are specific quality descriptors of text-based reviews such as 'enthusiastic', 'disappointed', and others, strongly associated with rating levels?

1.2 Literature Review

Prior to this, there was a lot of researches around user reviews in e-commerce. Some researchers use natural language processing to perform sentiment analysis of online product reviews [1][2],analyzing consumer's satisfaction with products. Some researchers used techniques that decompose the reviews into segments that evaluate the individual characteristics of a product such as quality, price and etc.[3]These measures help merchants improve products and services.But few researchers have integrated the three parameters star ratings, reviews, especially helpfulness ratings and converted them into quantitative indicators to make a quantitative measurement of the reputation of the product in the market.

1.3 Our work

First, we process the text reviews of the three product datasets to extract keywords and corresponding frequencies. As for the measurement of the success of the product, we have established a scoring model, which takes the three parameters star ratings, reviews, and helpfulness ratings as input to get the final score of the product.

1. We deployed an NLP instance to process the text reviews for keywords extraction and sentiment score evaluation.
2. We designed a scoring system and a weighing system to make a quantitative measurement of the star ratings, reviews, and helpfulness, thus calculate the product's reputation score.

3. We wrote several subsystems to help clients take advantage of our findings, the subsystems can emphasize the key factors which indicates the alteration of the product's reputation, give clients suggestions based on the keywords trend, and visualize the product's reputation with time-based pattern.

2 Preparation of the Models

2.1 Assumptions

We make the following basic assumptions in order to simplify the problem. Each of our assumptions is justified and is consistent with the basic fact.

- We trust Amazon's effort on being scrupulous in its consideration of vine qualifications, so we trust vine customers.
- We turst the Amazon community to be fair and objective when voting for the helpfulness of each review.
- We believe that if the length of view is longer, it will carry more information and being helpful most of the time.

2.2 Notations

The notation table contains all the notations we use in this paper.

Table 1: Notations

Symbol	Definition
$Multiplier$	Applied to the weight, controls the weight according to its input.
$Weightdictionary$	Customers' weights are stored here.
$Customer'sweight$	Applied to all reviews written by the same customer.
$Review'sweight$	Applied to the review only.
$Real-timereputation$	Weighted average of a certain period.
$Accumulatereputation$	Weighted average of all the scores before the time stamp.

3 The Model

We believe that it is important to tell if a customer's review is 'reliable', which means whether the customer is a trustworthy person who actually purchased the merchandise and gave an objective review, or a person who gave false review with nasty intention, or maybe even a bot made to deceive customers. We designed an evaluation system weighting every review entry to find out if the review entry, or the customer himself, is trustworthy.

Our model contains two main systems and some subsystems. The two main systems respectively give the score customer evaluate the merchandise and the weight of the review entry. The subsystems process the data given by main systems to enhance the readability of the data.

The scoring system is the main system that aims at making the mechanism more accurate in reflecting customer's rating. The star-rating system is discrete and having only 5 levels for customers to choose which struggles in judging 'how much' do the customers love/hate the merchandise. We apply an NLP instance to analyze the sentiment of the review content as a complement to the star rated which is converted to a float number between 0 and 1. Due to the accuracy of the NLP instance, if the difference of the two scores are larger than 0.3, the 'sentiment score' will not be used, otherwise the final score will be the average of the sentiment score and the star score.

The weighting system is the main system that aims at judging if the review or the customer is reliable. The output of the system is the weight of each review entry while the overall score of a product is the weighted average of each entry's 'final score'.

The weighting system will judge the weight in 2 aspects: the review itself and the customer. The review's weight will only be applied to itself, while the customer's weight will be applied to every review he wrote. The system holds a 'dictionary' to record the customer's weight. The system uses several 'weight multipliers' to adjust the weight as following:
First, the system gives the customer's weight calculated by the current review entry:

1. If the customer is marked as 'vine', then he should be awarded for the trust he earned. A 'vine' customer's weight will be directly set to 3.0, which is the largest weight of the system. The 'vine' customer's review will use no other multipliers.

2. If the review has not less than 5 votes, we will take the votes into consideration and enables 'vote multiplier'. We define 'helpful rate' as the percentage of helpful votes among all votes. If 'helpful rate' is more than 0.9 or less than 0.3, we will consider the customer is 'remarkably' supported or opposed by others, which indicates the customer is trustworthy or suspicious. The multiplier will be set at 0.2-0.8 (helpful rate under 0.3) or 1.2-1.5 (helpful rate over 0.9).

3. Meanwhile, if there are more votes, the 'helpful rate' is more convincing. The system uses an amplifier which the value will increase 0.02 for every vote more than the base 5 votes with its maximum value limited at 2.0. The final output of the vote multiplier will be the 'amplifier value' power of 'multiplier value'.

4. We noticed that some reviews are long, but the whole review are talking nonsense, some of them are even not English (or any other language). If the review's length reaches the average length while number of its keywords fails to reach half of the average, the weight of the customer will be reduced by half.

5. If the customer's weight is modified, the system will write the customer and his weight into the dictionary. If the customer already has a weight, the two weights will get geometric averaged.

Second, the system gives the review's own weight:

1. The system will use the same 'vote multiplier' and amplifier for the reviews whose 'helpful rate' is more than 0.3 and less than 0.9. The multiplier will be set at 0.8-1.0 (helpful rate under 0.7) or 1.0-1.2 (helpful rate over 0.7), since the

helpful rate cannot be considered as 'remarkable', the multiplier will be applied to the review itself only.

2. The 'length multiplier' will measure the length of the review and modify the weight. If the length of review is less the 4/5 of the average, the multiplier will be reduced to the minimum of 0.9, otherwise the multiplier will be enlarged to the maximum pf 1.5.

3. Some customers may abuse the refund policy and intentionally add negative reviews for the product, and some customers may receive free products from the reseller to help 'boost' the score of the product, so if the purchase is not verified, the review's weight will be reduced to half.

The final weight that will be applied to calculating the average is the multiply of the two weights.

The visualizing subsystem calculates the 'real-time' and 'accumulated' reputation score of the select period and product and outputs time-based patterns. The 'real-time' pattern is the weighted average of a certain period, for example, 30 days, and the 'accumulated' pattern is the weighted average of all the scores before the time stamp. This subsystem will help analysts to have an overall view of the product's reputation and identify the inflection points.

The keyword subsystem collects keywords that required by clients. Based on the NLP instance, the subsystem can run a statistic on all keywords and found keywords in strong relationship with certain emotion, and provide the 'feature keywords' of a certain type of review for user to predict the trend of product reputation or frequently mentioned keywords along with sentiments used to improve marketing strategy.

Here is a simplified flowchart of our model.

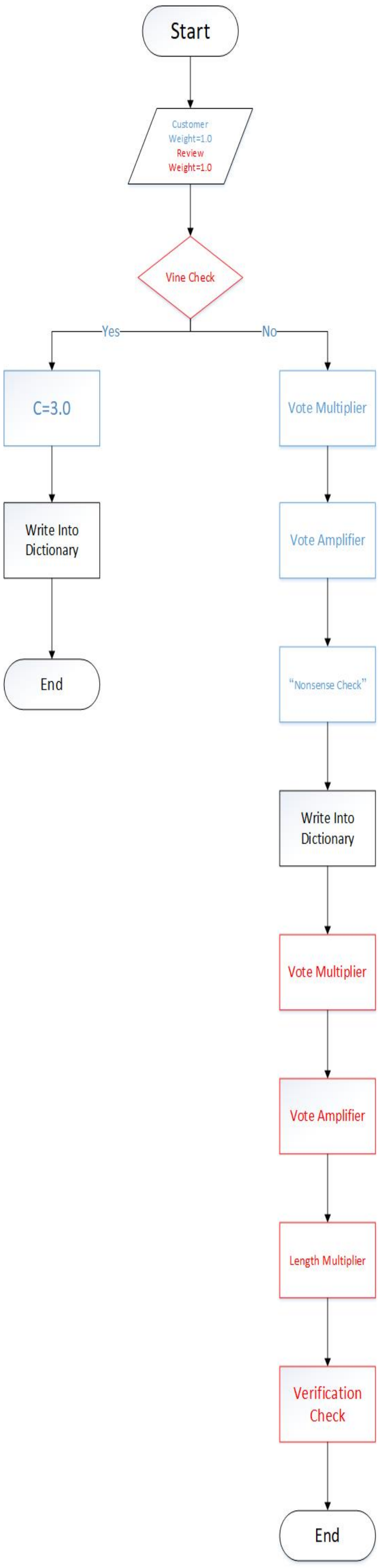

Figure 6: The Weighting System

4 Solutions

4.1 The First Solution

After the NLP analysis of experimental data, we have developed a scoring system with strong discrimination ability. The model to score and weight each review entry has been introduced in the last section and will not be repeated here. Using this model, we can evaluate a product's review and the customer made the review to get specific scores and corresponding weights for both of them. The final score is calculated using a weighted average of the scores for a certain time period (for example, one month). Our subsystem makes the reputation score can be tracked continuously, as well as visualized by tables or other visual forms, to give clients a better grasp of the overall product sales. The support of the accumulated and real-time mode makes it convenient for clients to track product reputation and analyze the alteration.

For example, our keywords subsystem can provide keywords which show frequently in positive/negative review. The client may use the information provided by the keywords to find the popular/hated feature of the product so that they could adjust their market strategy or improve their design.

4.2 The Second Solution

According to our analysis, each commodity can be weighted to obtain the result graphs (a) (real-time reputation value) and (b) (accumulated reputation value).

Figure (a) directly reflects the change of market reputation of the product in a certain period (In this case, one month). Be advised that the data here will not be representative unless combined with figure (b), as shown in the first sharp drop of the reputation caused by a review which is a low weight one, but at the same time, the only one in that month. We will discuss how to deal with this situation later. Figure (b) shows the general reputation of the product. Taking these two pictures as examples, we can clearly figure out that the overall 'accumulated' reputation of the product is dropping during the whole investigation cycle. At the same time, we can find that at two key inflection nodes (the first half of 2012 and the second half of 2013), both figures show a significant drop. This indicates that the product may have some problems during these two periods, leading to a worse reputation than before. Actually, we can find that the two figures are related while figure (a) serving as the derivative of figure (b). By combining the two figures, we can tell whether the product's reputation is rising or falling. Meanwhile, figure (b) makes it possible for us to ignore the low weight review that caused the sharp drop to obtain an overall view of the reputation, thus improve the reliability of our conclusion.

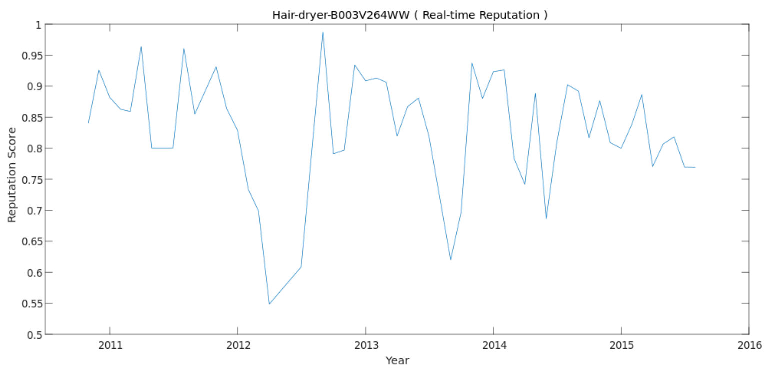

Figure 7: a

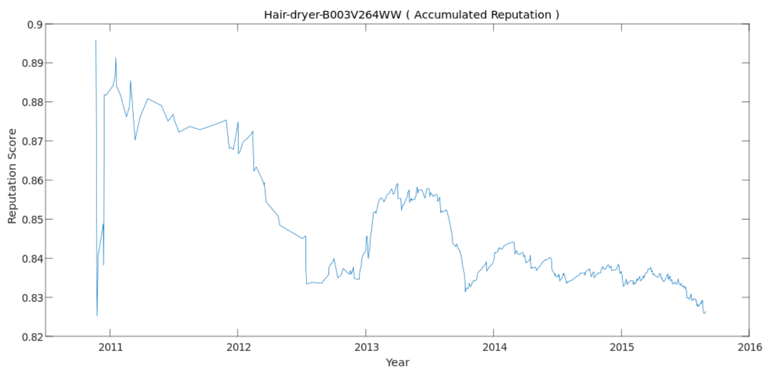

Figure 8: b

4.3 The Third Solution

Our model provides two ways to 'forecast' if the product is going to success.

First, the overall reputation score calculated by the model indicates if the product receive positive feedback and is welcomed by customers, our subsystem can provide visualized figures to show the current trend and any possible inflection points.

Second, our keyword subsystem also provides most frequently seen keywords in positive/negative reviews. These keywords can serve as an alert for a (potential) coming rise/drop in product reputation.

4.4 The Forth Solution

"Do specific star ratings incite more reviews?"

Our view is that customers will more likely give negative reviews after seeing a lot of negative ones. Meanwhile, if he saw that most of the previous reviews are positive, he would be more inclined to make positive comments. From a psychological perspective, this is a herd mentality. For merchants, there is a snowball effect. As shown in the figure below, the product was rated as 1 star (0.2 score) by a single customer at about 2012, though the review was marked by the system as a low weight one and not reliable at all, it did make the customers tend to write their reviews less positively than before. Our subsystem is able to be aware of the abnormal reviews and warn the client to take measures immediately to avoid a potential loss.

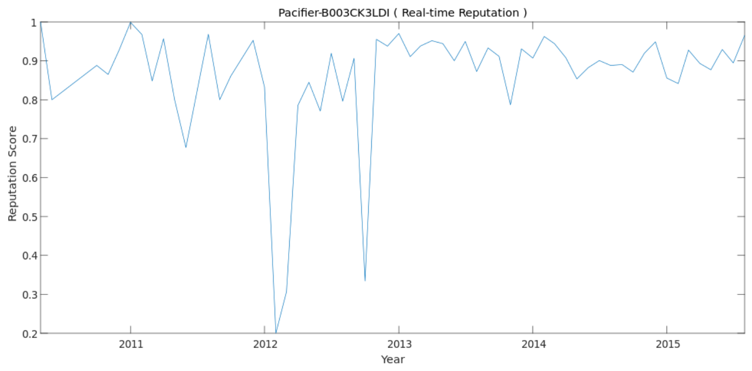

Figure 9: Pacifier-B003CK3LDI(Real-time Reputation)

4.5 The Fifth Solution

“Are specific quality descriptors of text-based reviews such as ‘enthusiastic’, ‘disappointed’, and others, strongly associated with rating levels? ”

We analyzed and filtered the keywords with our keywords subsystem and found that they, especially those from negative reviews, are highly correlated with star ratings. For example, in file ‘pacifier.tsv’, the frequency of the word ’waste’ in one-star reviews is 78.6 times of that in five-star reviews. The word ‘useless’ has a ratio of 75.5. Some more ‘negative words’ can be found in the table below.

Other than that, some scholars have discovered the phenomenon of "Text-Rating-Inconsistency".[4]For example,The comment expresses strong negative sentiment, but is associated with a 5-star rating.Removing these TRI reviews also helps us reduce the noise in the dataset, yielding better performance in later analysis. (note: the comment ratio is the ratio of the keyword appearing in low-star text reviews (like one-star and two-star) to five-star reviews)

Table 2: emotional keywords of pacifier

Keywords	Frequency ratio
disappointed	37.9
terrible	33.7
sadly	28.0

5 Strengths and Weaknesses

5.1 Strengths

Our model take time, star ratings, reviews, and helpfulness ratings into consideration. Based on the analysis above, we list strengths of our model as follows:

- **Time-based** Our subsystem provides time-based reputation figures and real-time keywords trend analysis, which enables the clients to analyze the inflection point and predicts the product’s reputation.

- **Accurate** Our model is configured with pre-built weight factors that would identify most of the fake reviews and make the reputation score reliable.
- **Expansible** Our and weighting system is made up of weight multipliers written in individual blocks, the system could be easily upgraded by adding new multipliers to take other factors into consideration.
- **Comprehensive** Our subsystem can visualize the data analyzed in multiple ways with virable parameters.

5.2 Weaknesses

- The parameters of weighting system is manually set and not verified, if we could get labelled data to train the weighting system the system will perform better.
- The NLP instance is not well and fully trained, thus its accuracy is lower than expected. We had to give up some of the sentiments scores whose difference with star rating is larger than 0.3.

6 Conclusion

In this paper, we use NLP to analyze the reviews made by customers to extract its sentiment and keywords. By analyzing the reviews and customers, we managed to realize a quantitative and review analyzing model which fulfilled our task. With our system, clients can get the reputation, quality issues and coming trends of the product by combining the visualized output of the systems. Furthermore, the research direction we are working on has greater practical significance in today's society. With the rapid development of e-commerce, how to analyze the online feedbacks? How to select the most helpful reviews from a large number of them? These issues still need further optimizations. The NLP instance, the weighting system, the key phrase analysis, machine learning, bigger datasets, better algorithm, our model has a long way to go. And we will not stop here.

References

[1] Fang, X., Zhan, J. Sentiment analysis using product review data. *Journal of Big Data* 2, 5 (2015). from `https://doi.org/10.1186/s40537-015-0015-2`

[2] T. U. Haque, N. N. Saber & F. M. Shah, Sentiment analysis on large scale Amazon product reviews, 2018 IEEE International Conference on Innovative Research and Development (ICIRD), Bangkok, 2018, pp. 1-6.

[3] N. Archak, A. Ghose, and P. G. Ipeirotis. Deriving the pricing power of product features by mining consumer reviews. *Management Science*, 57(8):1485-1509, 2011.

[4] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, & N. Sadeh. Why people hate your app: Making sense of user feedback in a mobile app store. *In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1276–1284, 2013.

Appendix

Sentiment analysis with NLTK

```
import pandas as pd
import nltk
import random
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
from sklearn.linear_model import LogisticRegression,SGDClassifier
from sklearn.svm import SVC, LinearSVC, NuSVC
from sklearn.naive_bayes import MultinomialNB,BernoulliNB
from nltk.classify.scikitlearn import SklearnClassifier

filename='c:/data/hair_dryer.tsv'

train1=pd.read_csv(filename, sep='\t', usecols=['review_body'])
train1=train1.values.tolist()

train2=pd.read_csv(filename, sep='\t', usecols=['star_rating'])
train2=train2.values.tolist()

for i in range(len(train1)):
    train1[i]="".join("%s"%id for id in train1[i])
    train1[i]=word_tokenize(train1[i])

    train2[i]="".join("%s"%id for id in train2[i])
    train1[i] = [word for word in train1[i] if word.lower() not in

stopwords.words('english') and word.isalpha()]

all_words = []

for i in range(len(train1)):
    for j in train1[i]:
        all_words.append(j)

all_words = nltk.FreqDist(all_words)

word_features = all_words.most_common()[:3000]

def find_features(document):
    words = set(document)
    features = {}
    for w in word_features:
        features[w[0]] = (w[0] in words)
    return features



featuresets = [(find_features(train1[i]),train2[i]) for i in

range(len(train1))]

training_set=featuresets[0:7000]


test_set=featuresets[7000:]


classifier = nltk.NaiveBayesClassifier.train(training_set)


print("Classifier␣accuracy␣percent:",(nltk.classify.accuracy(classifier,

testing_set))*100)


classifier.show_most_informative_features(50)
```

```
MNB_classifier = SklearnClassifier(MultinomialNB())
MNB_classifier.train(training_set)
print("MNB_classifier␣accuracy␣percent:", (nltk.classify.accuracy(MNB_classifier, testing_set))*100)


BernoulliNB_classifier = SklearnClassifier(BernoulliNB())
BernoulliNB_classifier.train(training_set)
print("BernoulliNB_classifier␣accuracy␣percent:",

(nltk.classify.accuracy(BernoulliNB_classifier, testing_set))*100)

LogisticRegression_classifier = SklearnClassifier(LogisticRegression())
LogisticRegression_classifier.train(training_set)
print("LogisticRegression_classifier␣accuracy␣percent:",

(nltk.classify.accuracy(LogisticRegression_classifier, testing_set))*100)

SGDClassifier_classifier = SklearnClassifier(SGDClassifier())
SGDClassifier_classifier.train(training_set)
print("SGDClassifier_classifier␣accuracy␣percent:",

(nltk.classify.accuracy(SGDClassifier_classifier, testing_set))*100)

SVC_classifier = SklearnClassifier(SVC())
SVC_classifier.train(training_set)
print("SVC_classifier␣accuracy␣percent:",

(nltk.classify.accuracy(SVC_classifier, testing_set))*100)

LinearSVC_classifier = SklearnClassifier(LinearSVC())
LinearSVC_classifier.train(training_set)
print("LinearSVC_classifier␣accuracy␣percent:",

(nltk.classify.accuracy(LinearSVC_classifier, testing_set))*100)
```

Main System

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

microwave = pd.read_csv('microwave_alter.csv')
hair_dryer = pd.read_csv('hair_dryer_alter.csv')
pacifier = pd.read_csv('pacifier_alter.csv')

microwave = microwave.fillna('␣')
hair_dryer = hair_dryer.fillna('␣')
pacifier = pacifier.fillna('␣')

def calculate_vote_weight(helpful_votes, total_votes):

    support = helpful_votes / total_votes

    if support < 0.3:
        weight = 2 * support + 0.2
        type = 0 #customer
    elif support < 0.7:
        weight = 0.5 * support + 0.65
        type = 1 #review
    elif support < 0.9:
        weight = support + 0.3
        type = 1 #review
    else:
        weight = 3 * support - 1.5
        type = 0 #customer

    amp = ((float(total_votes) - 5) * 0.02 + 1)
    if amp > 2:
        amp = 2

    weight = weight ** amp

    return type, weight

def calculate_length_weight(content_length, average_length):
    ratio = float(content_length) / average_length
    if ratio < 0.8:
        weight = 0.125 * ratio + 0.9
    else:
        weight = 0.5 * ratio + 0.6
    if weight > 1.5:
        weight = 1.5
    return weight

def calculate_final_score(star_rating, content_score):
    star_score = float(star_rating) / 5
    if content_score == -1:
        return star_score
    if abs(star_score - content_score) > 0.3:
        return star_score
    else:
        return (star_score + content_score) / 2

def customer_dict_write(dict, customer_id, customer_weight):
    if customer_id in dict:
        dict[customer_id] = (dict[customer_id] * customer_weight) ** 0.5
    else:
        dict[customer_id] = customer_weight

def customer_dict_read(dict, customer_id):
    if customer_id in dict:
        return dict[customer_id]
    else:
        return 1.0

def judge_weight(row, dict, average_length, average_keywords):
    # The vine customers have one and only advantage of boosting their weight to 3.0
    # The vine customers shall share no other weight multipliers
    if row['vine'] == 'y' or row['vine'] == 'Y':
        customer_dict_write(dict, row['customer_id'], 3.0)
        return 1.0

    # Initialize the weight
    customer_weight = 1.0
    review_weight = 1.0

    # If there are more than 5 votes, apply the vote weight multiplier
    if row['total_votes'] >= 5:
        [vote_type, vote_weight] = calculate_vote_weight(row['helpful_votes'], row['total_votes'])
        if vote_type == 0:
            customer_weight = customer_weight * vote_weight
        else:
            review_weight = review_weight * vote_weight

    # Customer weight will be punished if 'customer' is suspected talking nonsense
    if len(row['review_body']) > average_length and len(ast.literal_eval(row['keyword'])) < average_keywords /
        customer_weight = customer_weight / 2

    # All customer weight multipliers have been applied, if customer weight is no longer 1.0, write the weight
    if customer_weight != 1.0:
        customer_dict_write(dict, row['customer_id'], customer_weight)

    # Then we go on to judge the review-only weight
    # If the purchase is not verified, cut the weight to half
    if row['verified_purchase'] == 'n' or row['verified_purchase'] == 'N':
        review_weight = review_weight / 2

    # Use the length weight multiplier
    review_weight = review_weight * calculate_length_weight(len(row['review_body']), average_length)

    return review_weight

def preprocess(df):
    df['final_score'] = 0.0
    df['weight_review_only'] = 1.0
    df['final_weight'] = 1.0

def cal_score_weight(df):
    preprocess(df)
    customer_weight = {}
    entry_count = len(df['review_body'])
    length_sum = 0
    keyword_sum = 0
    for index, row in df.iterrows():
        df['final_score'][index] = calculate_final_score(row['star_rating'], row['content_score'])
        length_sum = length_sum + len(row['review_body'])
        keyword_sum = keyword_sum + len(ast.literal_eval(row['keyword']))
    average_length = length_sum / entry_count
    average_keywords = keyword_sum / entry_count
    for index, row in df.iterrows():
        df['weight_review_only'][index] = judge_weight(row, customer_weight, average_length, average_keywords)
    for index, row in df.iterrows():
        df['final_weight'][index] = row['weight_review_only'] * customer_dict_read(customer_weight, row['custo

cal_score_weight(microwave)
microwave.to_csv('microwave_weight.csv')
cal_score_weight(hair_dryer)
hair_dryer.to_csv('hair_dryer_weight.csv')
cal_score_weight(pacifier)
pacifier.to_csv('pacifier_weight.csv')
```