SMTP (Mathematics) Written Report

Decoding Hotel Reviews through Sentiment Analysis

Group 8–01

Fu Jinghang (3i104) Tan Jia He (3i325)

Yap Hao Ming Darren (3i227, L)

2023

Hwa Chong Institution (High School)

Contents

1 Introduction and Rationale

After the Singapore government relaxed travel restrictions due to COVID-19, there has been a recent increase in the number of tourists travelling in and out of Singapore. As such, hotels have seen a rise in the number of prospective tourists to be housed, and this may encourage an increase in the number of reviews hotels may receive.

Today, it is common to use social networks, messengers, and review websites to receive data from customer opinions. This is especially true for hotels, where previous occupants may evaluate the hotel on several factors through their reviews — be it cleanliness, facilities, location and convenience, etc. These come in two forms — a quantitative review (based on stars, diamonds, hearts, etc.) and a more qualitative review through text.

However, quantitative reviews do not always paint the full picture of customers' opinions towards a certain hotel. Though it is certainly helpful to have a more objective rating system using numerical scores, eg. the Department of Tourism grading system in the Philippines, or the European Hotelstars Union system, these are given by customers subjectively and do not reflect the reasons for customers giving the rating. There is also evidence of manipulation of ratings by hotel management itself, where hotels may be compelled to forge positive or negative ratings to bias the overall rating. This made up 2.1% of the 66 million reviews submitted to TripAdvisor (TripAdvisor, 2019). Therefore, we propose using sentiment analysis to extract customers' true feedback on hotels instead.

2 Objectives and Research Questions

2.1 Objectives

1. To run sentiment analysis on individual words and quantify them on a numerical scale

2. To run sentiment analysis on paragraphs and quantify them on a numerical scale

3. To use sentiment analysis on hotel reviews to determine consumers' overall opinions of hotels

2.2 Research Questions

1. How could we quantify the sentiments of individual words on a numerical scale?

2. How could we quantify the sentiments of paragraphs on a numerical scale?

3. How could we use tokens in hotel reviews to predict the overall sentiment of the review?

2.3 Fields of Math

- Data Science

- Machine Learning

- Probability and Statistics

2.4 Terminology

Below is a listing of the terminology, mostly pertaining to sentiment analysis, used in this report.

Table 1: Terminology used in this report.

Term	**Definition**
Sentiment analysis	Process of determining the emotional tone of text
Token	A unit of meaning (usually a word) that carries sentiment
Tokenise	To split a piece of text up into its constituent tokens, to be used for further analysis.
Lemmatise	To sort, so as to group together, inflected or variant forms of the same word, eg. ‘watching’, ‘watchful’, and ‘watched’
Stop words	Tokens that carry little meaning during sentiment identification, such as ‘is’, ‘I’, ‘that’, etc.; that should be filtered out before the processing of text
Lexicon	A dictionary that maps singular tokens to a category of sentiments or sentiment scores
Polarity	Whether a piece of text is positive, negative or neutral in sentiment
Bipartite sentiment	Sentiment which is grouped into two categories, usually positive and negative
Tripartite sentiment	Sentiment which is grouped into three categories, usually positive, negative and neutral
Sentiment score	A number that shows the overall sentiment of a token or a piece of text
Lexicon-based sentiment analysis	A method of sentiment analysis which sorts tokens into categories, aided by a lexicon, then calculates the overall sentiment score of a piece of text
Corpus-based sentiment analysis	A method of sentiment analysis which relies on the co-occurrences of tokens within a piece of text itself, rather than relying on an external lexicon

3 Literature Review

Sentiment analysis is a field of study which utilises computational methods to analyse text, and then categorises the text, usually into three main polarities — positive, neutral and negative. It has broad applications which range from determining consumers' opinion in sales and product analysis, to competitor research in marketing, and even detecting public opinion in social media monitoring. Sentiment analysis will be used in this project to analyse hotel occupants' reviews, and also determine the most significant upsides and downsides of each hotel, without interference from human bias.

Sohangir et al. (2018) predicted the stock market opinion of StockTwits communities of expert investors. Sentiment analysis was used in a Deep Learning model to extract sentiment from Big Data. A Pearson Correlation Coefficient combined the linear correlation between users' sentiment and future stock prices, which proved the accuracy of user sentiment to 53%. It was concluded that convolutional neural networks (CNN), a Deep Learning algorithm, was able to predict stock market movement based on sentiment.

Using the social networking site Twitter, Villavicencio et al. (2021) determined Filipinos' sentiment in response to the Philippine government's efforts at tackling COVID–19, specifically the implementation of vaccination. Natural Language Processing (NLP) techniques such as sentiment analysis were used to extract sentiment from text in the English and Filipino languages, which was used to train a Naïve Bayes model. A confusion matrix was produced, representing the prediction accuracy of the Naïve Bayes model (81.77%) at classifying sentiment into positive, neutral and negative categories. It was concluded that sentiment analysis towards COVID-19 vaccines were very accurate, even helping the Philippine government better conduct budget planning and coordinate COVID-19 efforts.

Borrajo-Millán et al. (2021) Borrajo-Millán et al., 2021 analysed tourism quality in

Spain by extracting sentiment from reviews by Chinese people on the tourism social networking sites Baidu Travel, Ctrip, Mafengwo, and Qunar. Two sentiment analysis methods, lexicon-matching and corpus-based machine learning methods, were used. These methods allow the processing of unstructured text of comparatively longer lengths. Clustered data visualisation categorised aspects of Spanish tourism into positive and negative groups, with the majority residing with positive sentiment. It was concluded that sentiment analysis can be used to improve tourism quality and sustainability decision-making.

Guzman et al. (2014b) used SentiStrength, a tool for lexical sentiment analysis — sentiment analysis done on short, low quality texts — to study emotions expressed in GitHub commit comments of different open-source projects. Their method involved assigning scores to each word, then calculating the net score for each comment. SentiStrength splits each comment into snippets, assigns each a score by computing the maximum and minimum scores of the sentences it contains. Following which, the average of the positive and negative scores is taken as the sentiment score of the entire commit. This study showed that Java projects warranted more negative comments, and projects which had more distributed teams tended to have a higher positive sentiment.

In conclusion, the literature reviewed showed many possible applications of sentiment analysis in quantifying the underlying emotion of feedback on online platforms. Lexicon-based sentiment analysis, which assigns each word a sentiment, then calculates a sentence's total sentiment score, can be used, due to its simplicity in implementation, and the availability of many open-source sentiment lexicons. In addition, sentiment categorisation using lexicon-based sentiment analysis makes accurate predictions upwards of 70% of the time (Khoo et al., 2017). SentiStrength would also be useful for detecting sentiment from hotel reviews which are usually short in length quickly and efficiently, optimising the process of extracting sentiment from tourists' reviews of hotels. Using SentiStrength for sentiment generation is also rather accurate, generating both positive and negative sentiments with more than 60% accuracy (Thelwall et al., 2010). Therefore, the strategies listed above could be adopted or emulated on a smaller scale for this project.

4 Methodology

4.1 Research Question 1

4.2 Research Question 2

4.3 Research Question 3

5 Results

5.1 Research Question 1

5.2 Research Question 2

5.3 Research Question 3

6 Discussion and Future Work

6.1 Summary

6.2 Limitations

6.3 Future Work

7 References

Borrajo-Millán, F., Alonso-Almeida, M.-d.-M., Escat-Cortes, M., & Yi, L. (2021). Sentiment analysis to measure quality and build sustainability in tourism destinations. https://doi.org/10.3390/su13116015

8 Appendices

8.1 Source Code (Research Question 1)

```
# imports
from afinn import Afinn
from os import path
import matplotlib.pyplot as plt
import nltk as nt
import pandas as pd
import wordcloud as wc

nt.download('punkt')
nt.download('stopwords')

# matplotlib
plt.figure(figsize=(3, 6), dpi=60)
plt.style.use('seaborn-v0_8')
plt.rcParams['font.family'] = ['Times New Roman', 'serif']

# define some stopwords
stop = nt.corpus.stopwords.words('english')
for i in '$-@_.&+#!*\\(),\'"?:%':
    stop.append(i)
stop.append('n\'t')

# read the data
data = pd.read_csv('./data/datafiniti_reviews.csv',
                   header=0,
                   sep=',',
                   on_bad_lines='skip')

# extract the title and body text of each review into a large list
bodies = data['reviews.text'].astype(str)
titles = data['reviews.title'].astype(str)

'''
remove extraneous words that should not be analysed:
```

```
remove '... More' from reviews (if it exists):
        '... More' (captured while web-scraping)
        'Bad', 'Good'
'''
bodies = bodies.str.replace('((Bad|Good):)|(\\.\\.\\. More)', '', regex=True)

# tokenise, remove stop words and puncutation
bodies_tokens = (bodies.apply(nt.word_tokenize)).apply(
    lambda x: [token for token in x if token.lower() not in stop])

# get a large array of all tokens to be analysed
bodies_tokens_raw = []
for bodies_sentence in bodies_tokens:
    for bodies_token in bodies_sentence:
        bodies_tokens_raw.append(bodies_token)

# create a list of tuples (token, sentiment)
tokens_sentiments = []

# sentiment analysis starts here.
afn = Afinn()

# rq1: token-based sentiment analysis.

'''loop through the tokens one by one,
assign each word a score, then add it to the list.'''
for token in bodies_tokens_raw:
    tokens_sentiments.append(tuple((token, afn.score(token))))

'''filter the sentiment data into three categories:
positive, neutral and negative.'''
sentiments_pos, sentiments_neg, sentiments_neu = [], [], []
for token_sentiment in tokens_sentiments:
    if token_sentiment[1] > 0:
        sentiments_pos.append(token_sentiment)
    elif token_sentiment[1] < 0:
        sentiments_neg.append(token_sentiment)
    else:
        sentiments_neu.append(token_sentiment)
```

```python
# generate a string of positive and negative tokens --
# these will be used for generating the wordclouds.
tokens_pos = "".join(token_pos[0] + " " for token_pos in sentiments_pos)
tokens_neg = "".join(token_neg[0] + " " for token_neg in sentiments_neg)

totals_bi = [len(sentiments_pos), len(sentiments_neg)]
totals_tri = [len(sentiments_pos), len(sentiments_neg), len(sentiments_neu)]
labels_bi = ['Positive', 'Negative']
labels_tri = ['Positive', 'Negative', 'Neutral']

# plot a bar graph for bipartite sentiments (+ve, -ve)
figure, axes = plt.subplots()
bars_container = axes.bar(labels_bi, totals_bi)
axes.set_title('Sentiments (Token-Based, Bipartite)')
axes.set_xlabel("Sentiment (Bipartite)")
axes.set_ylabel('Number of Tokens')
axes.bar_label(bars_container, fmt="{:,.0f}")
plt.savefig("./results/rq1/bar_bipartite.png", dpi=600)

# plot a bar graph for tripartite sentiments (+ve, -ve)
figure, axes = plt.subplots()
bars_container = axes.bar(labels_tri, totals_tri)
axes.set_title('Sentiments (Token-Based, Tripartite)')
axes.set_xlabel("Sentiment (Tripartite)")
axes.set_ylabel('Number of Tokens')
axes.bar_label(bars_container, fmt="{:,.0f}")
plt.savefig("./results/rq1/bar_tripartite.png", dpi=600)

# pie chart for bipartite sentiments
fig_pie_bi, ax_pie_bi = plt.subplots()
ax_pie_bi.set_title('Proportion of Tokens (Bipartite)')
ax_pie_bi.pie(totals_bi, labels=labels_bi, autopct="%1.1f%%", shadow=False)
plt.savefig("./results/rq1/pie_bipartite.png", dpi=600)

fig_pie_tri, ax_pie_tri = plt.subplots()
ax_pie_tri.set_title('Proportion of Tokens (Tripartite)')
ax_pie_tri.pie(totals_tri, labels=labels_tri, autopct="%1.1f%%", shadow=False)
plt.savefig("./results/rq1/pie_tripartite.png", dpi=600)
```

```
# wordcloud (positive tokens)
wordcloud = wc.WordCloud(background_color="white",
                         mode="RGB",
                         width=1280,
                         height=720)
wordcloud.generate(tokens_pos)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title("Word Cloud: Positive Tokens")
# plt.show()
wordcloud.to_file("./results/rq1/wordcloud_pos.png")

# wordcloud (negative tokens)
wordcloud.generate(tokens_neg)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title("Word Cloud: Negative Tokens")
# plt.show()
wordcloud.to_file("./results/rq1/wordcloud_neg.png")
```

8.2 Source Code (Research Question 2)

```
# imports
import matplotlib.pyplot as plt
import nltk as nt
import pandas as pd
import wordcloud as wc
import numpy as np

# matplotlib things
plt.figure(figsize=(3, 6), dpi=60)
plt.style.use('seaborn-v0_8')
plt.rcParams['font.family'] = ['Times New Roman', 'serif']
```

```python
# nt.download('punkt')
# nt.download('stopwords')

# open the file.
df = pd.read_csv('./data/sentistrength_data.csv')
positives, negatives = df['sent.pos'], df['sent.neg']
nets, polarities = [], []
for row in df.index:
    net_sentiment = positives[row] + negatives[row]
    nets.append(net_sentiment)
    polarity = 2
    if net_sentiment > 0:
        polarity = 1
    elif net_sentiment < 0:
        polarity = -1
    elif net_sentiment == 0:
        polarity = 0
    polarities.append(polarity)

# write the sentiments to a new csv
reviews = pd.read_csv('./data/datafiniti_reviews.csv',
                      header=0,
                      sep=',',
                      on_bad_lines='skip')
combined_data = reviews[['reviews.rating', 'reviews.title',
                         'reviews.text']].copy()
combined_data.insert(1, value=df['sent.pos'], column='sent.pos')
combined_data.insert(2, value=df['sent.neg'], column='sent.neg')
combined_data.insert(3, value=nets, column='sent.net')
combined_data.insert(4, value=polarities, column='sent.polarity')
combined_data.to_csv('./data/combined_sentiments.csv')

positive_no = sum(pol == 1 for pol in polarities)
neutral_no = sum(pol == 0 for pol in polarities)
negative_no = sum(pol == -1 for pol in polarities)

# print charts and stuff
# tripartite
```

```
fig_tri, ax_tri = plt.subplots()
labels_tri = 'Positive', 'Negative', 'Neutral'
fracs_tri = [positive_no, negative_no, neutral_no]

ax_tri.pie(fracs_tri, labels=labels_tri, autopct='%1.1f%%', shadow=False)
ax_tri.set_title('Proportion of Positive, Negative and Neutral Reviews')
plt.savefig('./results/rq2/pie_chart_3part.png', dpi=600)

# bipartite
fig_bi, ax_bi = plt.subplots()
labels_bi = 'Positive', 'Negative'
fracs_bi = [positive_no, negative_no]
ax_bi.pie(fracs_bi, labels=labels_bi, autopct='%1.1f%%', shadow=False)
ax_bi.set_title('Proportion of Positive and Negative Reviews')
plt.savefig('./results/rq2/pie_chart_2part.png', dpi=600)
```

8.3 Source Code (Research Question 3)

```
#!/usr/bin/env python
# coding: utf-8
'''
[RQ3]
can we predict the relationship between the:
- independent: frequency of tokens of a review and its polarity?
- dependent: polarity (positive / negative)
'''
# In[1]:

# data processing
from sklearn.metrics import (auc, average_precision_score,
                             precision_recall_curve, roc_curve)
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from funcsigs import signature
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
```

```python
import string
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords, wordnet
from nltk import pos_tag
import pandas as pd
# import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# natural language processing
import nltk

nltk.data.path.append('/usr/share/nltk_data/')

# machine learning imports

# matplotlib things
plt.style.use('seaborn-v0_8')

# In[2]:

# import the data
df = pd.read_csv('./data/combined_sentiments.csv',
                 header=0,
                 sep=',',
                 on_bad_lines='skip')

# lemmatise


def get_wordnet_pos(pos_tag):
    if pos_tag.startswith('J'):
        return wordnet.ADJ
    elif pos_tag.startswith('V'):
        return wordnet.VERB
    elif pos_tag.startswith('N'):
        return wordnet.NOUN
    elif pos_tag.startswith('R'):
        return wordnet.ADV
```

```
    else:
        return wordnet.NOUN


# check whether there is a digit or not


def check_digits(text):
    return any(i.isdigit() for i in text)


# tokenise


def clean_review(review):
    review = str(review)
    review = review.lower() # turn into lowercase
    review = [word.strip(string.punctuation)
              for word in review.split(' ')] # remove punctuation
    # remove digits
    review = [word for word in review if not check_digits(word)]

    # remove stop words
    stop = stopwords.words('english')
    review = [token for token in review if token not in stop]
    # remove empty tokens
    review = [token for token in review if len(token) > 0]

    # tag each token with its part of speech (pos)
    pos_tags = pos_tag(review)
    review = [
        WordNetLemmatizer().lemmatize(tag[0], get_wordnet_pos(tag[1]))
        for tag in pos_tags
    ]

    # remove words with only one letter
    review = [token for token in review if len(token) > 1]
    review = ' '.join(review)
    return review
```

```python
# generate a cleaned, tokenised and lemmatised version of the reviews
df['reviews.clean'] = df['reviews.text'].apply(lambda x: clean_review(x))

# In[3]:

# extract vector representations for each review.
documents = [
    TaggedDocument(doc, [i])
    for i, doc in enumerate(df["reviews.clean"].apply(lambda x: x.split(' ')))
]

# train a doc2vec model
model = Doc2Vec(documents, vector_size=5, window=2, min_count=1, workers=4)

# transform each document into vec data
df_vec = df['reviews.clean'].apply(
    lambda x: model.infer_vector(x.split(' '))).apply(pd.Series)
df_vec.columns = ['vec_' + str(x) for x in df_vec.columns]
df = pd.concat([df, df_vec], axis=1)

# In[4]:

# add the term frequency - inverse document frequency values for every word
tfidf = TfidfVectorizer(min_df=10)
tfidf_result = tfidf.fit_transform(df['reviews.clean']).toarray()
tfidf_df = pd.DataFrame(tfidf_result, columns=tfidf.get_feature_names_out())
tfidf_df.columns = ['word_' + str(x) for x in tfidf_df.columns]
tfidf_df.index = df.index
df = pd.concat([df, tfidf_df], axis=1)

# In[5]:

# distribution of sentiments
for polar in [-1, 1]: # positive or negative (don't consider neutrals)
    subset = df[df['sent.polarity'] == polar]
    if polar == -1:
        label = 'negative'
```

```
    else:
        label = 'positive'
    sns.distplot(subset['sent.net'], hist=False, label=label)

# In[6]:

# is_bad: True if polarity == -1 else False
df['review.is_bad'] = df['sent.polarity'].apply(lambda x: x == -1)

# feature selection
label = 'review.is_bad'
ignore_cols = [
    label, "sent.polarity", "sent.pos", "sent.neg", "sent.net", "index",
    "reviews.rating", "reviews.clean", "reviews.title", "reviews.text"
]
features = [col for col in df.columns if col not in ignore_cols]

# split the data into train and test
x_train, x_test, y_train, y_test = train_test_split(df[features],
                                                    df[label],
                                                    test_size=0.2,
                                                    random_state=42)

# In[7]:

# train a random forest classifier
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(x_train, y_train)

# show feature importance
feature_importances_df = pd.DataFrame({
    'feature': features,
    'importance': rf.feature_importances_
}).sort_values('importance', ascending=False)
# feature_importances_df.head(20)

# In[8]:

y_pred = [pred[1] for pred in rf.predict_proba(x_test)]
```

```
# false +ve rate, true +ve rate
fpr, tpr, thresholds = roc_curve(y_test, y_pred, pos_label=1)
'''
receiver operating characteristic:
the higher the curve above the diagonal baseline, the better the preds
'''
roc_auc = auc(fpr, tpr)

# plot the roc curve
# TODO export this curve.
plt.figure(1, figsize=(15, 10))
lw = 2
plt.plot(fpr,
         tpr,
         color='darkorange',
         lw=lw,
         label='ROC Curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], lw=lw, linestyle='-')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic of Sentiment Prediction')
plt.legend(loc="lower right")
plt.show()

# In[9]:

# area-under-curve precision-recall
average_precision = average_precision_score(y_test, y_pred)

precision, recall, _ = precision_recall_curve(y_test, y_pred)

step_kwargs = ({
    'step': 'post'
} if 'step' in signature(plt.fill_between).parameters else {})

plt.figure(1, figsize=(15, 10))
plt.step(recall, precision, color='b', alpha=0.2, where='post')
```

```
plt.fill_between(recall, precision, alpha=0.2, color='b', **step_kwargs)

plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
# TODO export this curve.
plt.title('2-Class Precision-Recall Curve. Average Precision: {0:0.2f}'.format(
    average_precision))
```