**Attn: Dr. Sun Aixin**

Logo

Description automatically generated

**CE/CZ4045 Natural Language Processing**

We hereby declare that the attached group assignment has been researched, undertaken, completed and submitted as a collective effort by the group members listed below. We have honored the principles of academic integrity and have upheld Student Code of Academic Conduct in the completion of this work. We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work.

Important note: Name must **EXACTLY MATCH** the one printed on your Matriculation Card. Any mismatch leads to **THREE (3)** marks deduction.

|  |  |
| --- | --- |
| Name | Signature / Date |
| Lin Yue | **24/10/2021** |
| Suthakar Shiny Gladdys | 24/10/2021 |
| Wong Yung Shen | Shape  Description automatically generated with low confidence 24/10/2021 |
| Marcus Tang Zi Yang | 24/10/2021 |

**Review Data Analysis and Processing**



Lin Yue †  
 School of Computer Science and Engineering  
 Nanyang Technological University, Singapore  
 liny0087@e.ntu.edu.sg

Suthakar Shiny Gladdys  
 School of Computer Science and Engineering  
 Nanyang Technological University, Singapore  
 su0001ys@e.ntu.edu.sg

Wong Yung Shen  
 School of Computer Science and Engineering  
 Nanyang Technological University, Singapore  
wong1100@e.ntu.edu.sg

Marcus Tang Zi Yang  
School of Computer Science and Engineering  
 Nanyang Technological University, Singapore  
mtang004@e.ntu.edu.sg

ABSTRACT

In this project, the use of natural language processing was demonstrated by analyzing a collection of reviews posted on Yelp as well as posts from popular sites such as StackOverflow, HardwareZone and ChannelNewsAsia. In the first section, we have done data cleaning on the reviews to make sure that all reviews are written in English language without stopwords that are included in the NLTK library to remove noises for the following experiments. In section 2.1, 2.2 and 2.4, analysis was done on the Yelp reviews and in section 2.3, the writing style of posts from websites of different domains were analyzed. In section 3, we extracted and examined the adjective phrases in the reviews of a selected business and identified the unique characteristics of the business that can be derived from the reviews. Lastly, in section 4, we developed a console-based application that could take in a sentence and classify whether the sentence is a negation expression.

1 Data Preprocessing

Before performing any NLP task, data cleaning had performed to remove irrelevant and non-English reviews. With the langdetect library, a direct port of Google’s language-detection library from Java to Python [1], we had filtered out 3 irrelevant reviews that each does not gives us any information.

Text

Description automatically generated

Figure 1: Data Cleaning on Irrelevant Review



Figure 2: Reviews that are classified not a language

From reviews in Fig. 2 above, these reviews are not written in any known human language hence they should be removed from our dataset. Then, we performed language classification on all reviews to remove all non-English review.

Text

Description automatically generated

Figure 3: Language classification on reviews and its results

Fig. 3 shows there are 50 non-English reviews in the dataset. After manual checking with the 50 classified non-English reviews, there is a review ‘grand opening’, written in English but it is not a proper written review. Hence, our final dataset will only have a total 15247 reviews.

**2.1 Tokenization and Stemming**

Before tokenization of our reviews, stopwords and any other irrelevant word/ phrases should be removed.

Text

Description automatically generated

Figure 4: Function to clean up and tokenize review

Fig. 4 shows how review is being cleaned and tokenized. With this function, hyperlinks, tickers, prices and other irrelevant words/ phrases will be removed before tokenization. TweetTokenizer from NLTK.tokenize library is being used.

Text

Description automatically generated

Figure 5: Function to clean up, tokenize, and stem review

Fig. 5 shows how review is being cleaned and stemmed. Cleaning process for review is same as Fig 4 above. Porter Stemmer from NLTK.stem library is being used after tokenization.

Graphical user interface, text

Description automatically generated

Figure 6: Functions to build the frequency dictionary

Fig 6 shows how a word frequency dictionary is built by the function. With these functions, dictionary for recording frequency of words can be built to support future experiments.

Chart

Description automatically generated

Figure 7: Top 20 Most Frequent Words in Selected Business B1's Review

Shape

Description automatically generated

Figure 8: Top 20 Most Frequent Words in Selected Business B2's Review

Fig. 7 and Fig. 8 show the top 20 most frequent words/stemmed words for selected business B1 and B2. After stemming, word frequency tends to be more concentrate on those words that occur frequently.

Chart, funnel chart

Description automatically generated

Figure 9: Top 10 Most Frequent Words in Selected Business B1's Review

For business B1, list of top-10 most frequent words before stemming is ['pho', 'food', 'good', 'vietnamese', 'place', 'service', 'rolls', 'rice', 'restaurant', 'like', 'beef']. After stemming, the list is ['pho', 'roll', 'food', 'place', 'restaur', 'good', 'vietnames', 'servic', 'like', 'order', 'rice'].

The word ‘rolls’ is 7th most frequent word, but ‘roll’ is 2nd most frequent stemmed words. We believe both words represent the same meaning and after stemming, its frequency increased approximately 20.

Chart

Description automatically generated

Figure 10: Top 10 Most Frequent Words in Selected Business B2's Review

For business B2, list of top-10 most frequent words before stemming is ['pho', 'good', 'place', 'great', 'food', 'service', 'broth', 'delicious', 'friendly', 'try', 'go']. After stemming, the list is ['pho', 'place', 'good', 'great', 'food', 'servic', 'tri', 'price', 'order', 'time', 'broth'].

After stemming, frequency of word ‘place‘ and ‘tri’ had increased. Hence, we believe stemming has reduced inflectional forms of words, helps in accuracy of the following experiments.

**2.2 POS Tagging**

Parts-of-speech tagging is done using two libraries – spaCy and NLTK. The NLTK default POS tagger uses the Penn Treebank Tagset, while the spaCy tagger “en\_core\_web\_sm” uses the Universal Dependencies Scheme [1]. Using the native Python random library, we randomly sample 5 reviews from the dataset.

lol

Figure 11: Random sampling of 5 reviews from dataset

From the five selected reviews, we select the first sentence. POS tagging is applied to the entire review, and we extract the relevant POS tags for each sentence. The five sentences are:

1. This hotel is absolutely magnificent with a hipster vibe!
2. This bar is amazing!!
3. Pretty good place to nosh in funky cold medina.
4. Amazing food with a unique bar layout!
5. I went for dinner at Octagon this week with a group of 10 people.

The spaCy and NLTK taggers are applied to each review. The POS tags are extracted and stored.

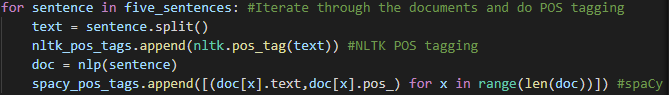


Figure 12: Extracting POS tags using NLTK and spaCy

The tags corresponding to the five sentences are manually extracted from the data. The tags returned by NLTK tagger are:

1. ('This', 'DT'), ('hotel', 'NN'), ('is', 'VBZ'), ('absolutely', 'RB'), ('magnificent', 'JJ'), ('with', 'IN'), ('a', 'DT'), ('hipster', 'NN'), ('vibe!', 'NN')
2. ('This', 'DT'), ('bar', 'NN'), ('is', 'VBZ'), ('amazing!!', 'JJ')
3. ('Pretty', 'RB'), ('good', 'JJ'), ('place', 'NN'), ('to', 'TO'), ('nosh', 'VB'), ('in', 'IN'), ('funky', 'JJ'), ('cold', 'JJ'), ('medina.', 'NN')
4. ('Amazing', 'VBG'), ('food', 'NN'), ('with', 'IN'), ('a', 'DT'), ('unique', 'JJ'), ('bar', 'NN'), ('layout!', 'VBD')
5. ('I', 'PRP'), ('went', 'VBD'), ('for', 'IN'), ('dinner', 'NN'), ('at', 'IN'), ('Octagon', 'NNP'), ('this', 'DT'), ('week', 'NN'), ('with', 'IN'), ('a', 'DT'), ('group', 'NN'), ('of', 'IN'), ('10', 'CD'), ('people.', 'NN')

The tags returned by spaCy tagger are:

1. ('This', 'DET'), ('hotel', 'NOUN'), ('is', 'AUX'), ('absolutely', 'ADV'), ('magnificent', 'ADJ'), ('with', 'ADP'), ('a', 'DET'), ('hipster', 'ADJ'), ('vibe', 'NOUN'), ('!', 'PUNCT')
2. ('This', 'DET'), ('bar', 'NOUN'), ('is', 'AUX'), ('amazing', 'ADJ'), ('!', 'PUNCT')
3. ('Pretty', 'ADV'), ('good', 'ADJ'), ('place', 'NOUN'), ('to', 'ADP'), ('nosh', 'PROPN'), ('in', 'ADP'), ('funky', 'ADJ'), ('cold', 'PROPN'), ('medina', 'PROPN'), ('.', 'PUNCT')
4. ('Amazing', 'ADJ'), ('food', 'NOUN'), ('with', 'ADP'), ('a', 'DET'), ('unique', 'ADJ'), ('bar', 'NOUN'), ('layout', 'NOUN'), ('!', 'PUNCT')
5. ('I', 'PRON'), ('went', 'VERB'), ('for', 'ADP'), ('dinner', 'NOUN'), ('at', 'ADP'), ('Octagon', 'PROPN'), ('this', 'DET'), ('week', 'NOUN'), ('with', 'ADP'), ('a', 'DET'), ('group', 'NOUN'), ('of', 'ADP'), ('10', 'NUM'), ('people', 'NOUN'), ('.', 'PUNCT')

The NLTK tagger does not provide overt punctuation tags unlike the spaCy tagger. Referring to the Penn Treebank Tagset [2], most of the tags identified by the two taggers correspond to each other. However, some tags are different as spaCy’s tagset has an additional AUX “auxiliary” tag which is not defined in the NLTK tagset. Words in this class, however, are consistently tagged in the same way across both taggers – for example, the word ‘is’, functions as a verbal copula and is tagged as AUX for spaCy and VBZ “Verb, 3rd person singular present” for NLTK. There are POS tagging errors noted for both the spaCy and NLTK taggers – for example, the word ‘nosh’ in sentence 3 is slang for ‘eating’, which is a verb and is therefore incorrectly classified as a proper noun. This word may not have been seen in the training corpus and therefore the POS taggers are unable to accurately tag the word.

Generally, the POS taggers used here were not trained specifically on review-like text and hence cannot be expected to perform as well.

**2.3 Writing Style**

For the analysis of writing style, 2 posts/articles are randomly sampled from StackOverflow, HardwareZone and ChannelNewsAsia each (refer to Appendix A).

The news articles sampled from ChannelNewsAsia have a consistently good writing style in terms of the first letter of the first word in every sentence being capitalized, accurate grammar, first letter of the words in named entities being capitalized and complete punctuation. For example, the named entities’ first letters are capitalized such as ‘Pope Francis’ and ‘Shopify’ in the sampled articles from ChannelNewsAsia. The main reason could be since these are all formal articles. Thus, they are published by an organization or on behalf of an organization and they would have been vetted thoroughly before being published.

The posts sampled from StackOverflow almost follow the above trend too except for slight grammatical errors and proper nouns not being capitalized seldomly. For example, in the first sampled post from StackOverflow, the word ‘to’ is written as ‘o’. Furthermore, the phrase ‘it should give decaying trend’ should be changed to be grammatically correct, which would be ‘it should give a decaying trend’. This could be due to the fact that the posts are posted by an individual and since StackOverflow is a public forum, most likely they would not be vetted or checked thoroughly prior to posting. Moreover, since other people will most probably still understand the posts with informal writing and slight grammatical errors, post owners might not give much importance to grammar and writing style when they post.

The posts sampled from HardwareZone forums have mediocre writing style due to improper sentence structure, missing punctuation, heavy grammatical errors, and Singaporean slang even though the first letter of the first word in a sentence and proper nouns are capitalized in seldom cases. Since HardwareZone forums are based in Singapore and they are more of lifestyle discussion forums, they are bound to contain Singlish slang phrases and words and have a more informal tone, compared to StackOverflow. Since StackOverflow is more of an educational forum and is based worldwide, it has a more formal tone and does not really contain slang specific to any cultural group, for the posts to be understood better by the wider target audience.

Graphical user interface, application, Word

Description automatically generated

Figure 13: First post sampled from HardwareZone forums

For example, as seen in Fig. 13, for the first post sampled from HardwareZone forums, the post title contains a Singlish abbreviation ‘GVGT’, which expands to ‘Got Video Got Talk’. This abbreviation is probably mentioned as the post contains a video in relation to the post content. Furthermore, we can see that Singlish slang words such as ‘AngMoh’ and ‘tong’ are being used in both the posts sampled from HardwareZone forums.

The above tools used for tokenization and POS tagging can only be used for the posts in StackOverflow after they have undergone some intermediate level of pre-processing. This is because many a times, StackOverflow posts contain pieces of code and since code is not proper linguistic text, they need to be removed before the post can undergo tokenization and POS tagging. Furthermore, the pre-processing should include removing improper words and applying punctuation wherever appropriate and needed. This is due to the fact that some StackOverflow posts have the tendency to contain incomplete words or punctuation due to human error and the lack of vetting of the informal posts as discussed above.

**2.4** **Most frequent ⟨ Noun - Adjective ⟩ pairs for each rating**

Here, we analyze the recurrence of noun-adjective pairs in reviews belonging to each rating. Firstly, we prepare the data by removing reviews that do not contain actual text and reviews that are of non-English language. Furthermore, we ensure the reviews do not contain prices, stock market tickers, numbers, hyperlinks, ellipses, and hashtags (as discussed above).

Graphical user interface, application, Word

Description automatically generated

Figure 14: Random sampling of 50 reviews from unique businesses

As seen in Fig. 14, we then proceed with finding the top 10 most frequent noun-adjective pairs for reviews of rating 1. After extracting only reviews that have a rating of 1 from the dataset, we first randomly sample one review from every business, and then from the resulting data we randomly sample 50 reviews. This leaves us with 50 reviews with unique business\_id values.

Graphical user interface, application, Word

Description automatically generated

Figure 15: Finding the noun-adjective pairs of each review

Inferring from Fig. 15, we now perform the identification of noun-adjective pairs among the 50 reviews. The ‘en’ (English pipeline) model is first loaded from spaCy. For every review, each word’s POS tag is checked to find if it is a noun or proper noun. If so, the first adjective that appears after that word is searched. If it exists, the adjective and the noun or proper noun are stored as a tuple. Since the above piece of code returns a list of noun-adjective pairs for every review, we would get a list of lists of noun-adjective pairs for all the reviews.

Graphical user interface, application, Word

Description automatically generated

Figure 16: Obtaining frequency of noun-adjective pairs

As we can see from Fig. 16, we convert the previously obtained list of lists of noun-adjective pairs to a flat list of noun-adjective pairs for easier data handling. All the noun-adjective pairs are converted to lowercase. A Counter instance is used to store the elements of the flat list as keys and their frequency as values. Through this, we can get the counts of the unique noun-adjective pairs. as shown below.

Graphical user interface, application, Word

Description automatically generated

Figure 17: Finding the top 10 most frequent noun-adjective pairs

In Fig. 17, we convert the Counter into a DataFrame and get the top 10 noun-adjective pairs in terms of frequency. Fig.17 shows the result for the sampled reviews of rating 1.

Graphical user interface, application, Word

Description automatically generated

Figure 18: Top 10 most frequent noun-adjective pairs for reviews of rating 1

We repeat the above process for reviews of ratings 2, 3, 4, 5 (with 20 samples each) to obtain the following results respectively.

Graphical user interface, application, Word

Description automatically generated

Figure 19: Top 10 most frequent noun-adjective pairs for reviews of rating 2

A screenshot of a computer

Description automatically generated with medium confidence

Figure 20: Top 10 most frequent noun-adjective pairs for reviews of rating 3

Graphical user interface, application, Word

Description automatically generated

Figure 21: Top 10 most frequent noun-adjective pairs for reviews of rating 4

Graphical user interface, application, Word

Description automatically generated

Figure 22: Top 10 most frequent noun-adjective pairs for reviews of rating 5

Since the above method finds the adjective closest to the noun or proper noun, it preserves the attribute of the noun or proper noun to a large extent. One limitation of this method is that if a food name appearing in a review consists of more than one word, it is broken up among the noun-adjective pairs. Since almost all reviews are restaurant-based, most of the named entities are referring to food. Named Entity Recognition (NER) tagging models offered by toolkits such as NLTK and spaCy only work on named entities such as names of persons, places, organizations, and monetary values. We will have to train a customized model specifically if we want to apply NER on food names.

**3** **Extraction of Indicative Adjective Phrases**

In this section, we will randomly select a business b1 from the dataset and find the adjective phrases that appear more frequently in b1 than in other businesses. The adjective phrases obtained might be able to reflect the unique characteristic of b1.

Text

Description automatically generated

Figure 23: A random business\_id is selected with the Random library

To extract the adjective phrases from the reviews instead of extracting the adjective word alone, we made use of the Stanford Corenlp.

A screenshot of a computer

Description automatically generated with low confidence

Figure 24: Pipeline of Stanford Corenlp

The Stanford Corenlp parser takes in a sentence and automatically perform tokenization, splitting, POS tagging, lemmatization on the input sentence.

Graphical user interface, text, application

Description automatically generated

Figure 25: An example of the output of the parser

In the above figure, we passed a sample sentence “The restaurant is very tight an uncomfortable” and obtained a tree as shown above.

Text

Description automatically generated

Figure 26: Traverse the tree

Text

Description automatically generated

Figure 27: Traverse each tree leaf

We then traversed through the NLTK tree and extract those phrases with tag ADJP, which indicates an adjective phrase.

Taking the tree in Fig. 25 as an example, the phrase “very tight and uncomfortable” will be extracted as it is labelled as ADJP generally.

**3.1** **Pre-Requisite**



Figure 28: install StanfordCoreNLP



Figure 29: import the StanfordCoreNLP library

To run this section, we need to install the StanfordCoreNLP and download the library from the link provided in Appendix B. Then we need to specific the file path and create an instance of StanfordCoreNLP.

StanfordCoreNLP is written in **Java**; recent releases require **Java 1.8+.**

**3.2** **Data Cleaning**

Text

Description automatically generated

Figure 30: Clean and split review

Some additional data cleaning is done on the data. Each review is split into sentences using Tokenize.sent\_tokenize() to speed up the parsing process.

**3.3** **Applying on B1**

Text

Description automatically generated

Figure 31: Frequency of each ADJP

After applying the parser, we count the number of each ADJP and divide it with the number of reviews to obtain the frequency of each ADJP.

In the above figure, we observed that “good” and “great” has the highest frequency. However, they do not reflect any unique characteristic of the b1.

**3.4** **Applying on other Businesses**

Graphical user interface, text, application

Description automatically generated

Figure 32: Randomly select 10 businesses

We then randomly selected 10 other businesses to compare what are the adjective phrases that appeared more in b1.

Text

Description automatically generated

Figure 33: Most common adjective phrases in the 10 selected businesses

Without any surprise, the common words in other business are “good” and “great” as well. There is a need to compare with the ADJPs of b1 to determine the characteristics of b1.

Text

Description automatically generated

Figure 34:Comparing b1 and 10 other businesses

By comparing the frequency of each ADJP in b1 and the ADJPs obtained from the randomly selected 10 businesses, we could observe that the ADJP which is the most unique to b1 is the phrase “vietnamese”. This indicates b1 might be a Vietnamese restaurant.

Some other adjective phrases that appear more in b1 are “very rude”, “very tasty”, “cheap”, “very busy”, “pretty decent”, “unfresh” etc. From these adjective phrases, we can infer that b1 is an affordable and busy Vietnamese restaurant with some food being tasty and some food being not as fresh.

**3.5** **Going through the Reviews Manually**

Text

Description automatically generated

Figure 35: Snapshot of the original reviews

The figure above is a screen shot of some sample reviews of b1. We could conclude from the reviews that b1 is a small Vietnamese restaurant that sell in an affordable price, but the customer service is horrible. However, the information about it having bad customer service is not picked up. Those reviews usually come as “worst customer service”, “poor customer service”, “horrible service” etc. The words “worst”, “poor” and “horrible” are labelled as adjective, JJS, at word level, but they are not extracted at phrase level.

In general, the selected adjective phrases reflect most of the unique characteristic of business b1.

**4** **Negation Expression Detection**

Our Negation Expression Detection application is developed using Naïve Bayes Theorem. All words will be stemmed before building our own dictionary.

A screenshot of a computer

Description automatically generated

Figure 36: Function to count frequency for (token, label) pair

Using function in Fig 36 above, a frequency dictionary with mapping each (word, label) pair to its frequency will be built.

Text

Description automatically generated

Figure 37: Function to build log likelihoods dictionary for each stemmed token

Using function in Fig 37 above and frequency dictionary, we can build a dictionary to store the log likelihoods for each word.

Text

Description automatically generated

Figure 38: Function to predict the review with logprior and log likelihood dictionary

With the log likelihood dictionary, we can calculate the probability of the review by adding the log likelihood of each word in the review. Review with probability>0, the review will be classified as positive expressions else a negation expression. Logprior and log likelihood will be saved in pkl files and to be used by the application.

Text

Description automatically generated

Figure 39: Snapshot of the Negation Expression Application

After running the application, enter review as instructed. The application will then predict whether the review input is a negation expression or not. In this example, ‘Manager’s attitude is bad’ is used and was classified as a negation expression. The model’s accuracy may be improved as dataset used for this model tends to have bias to food and beverages businesses.

APPENDIX A

Sampled Posts for Section 2.3

|  |  |  |
| --- | --- | --- |
| Source | Post No. | URL of Post |
| StackOverflow | 1 | <https://stackoverflow.com/questions/69645364/i-have-fitting-function-organizer-problem> |
| StackOverflow | 2 | <https://stackoverflow.com/questions/69645352/how-to-prevent-spyder-5-from-lagging-which-makes-the-editor-unusable-on-large-f> |
| HardwareZone | 1 | <https://forums.hardwarezone.com.sg/threads/gvgt-isis-declaring-war-on-china.6627382/> |
| HardwareZone | 2 | <https://forums.hardwarezone.com.sg/threads/how-long-are-u-willing-to-work-for-yr-company-without-pay.6627402/> |
| ChannelNewsAsia | 1 | <https://www.channelnewsasia.com/business/shopify-enlists-microsoft-oracle-business-tools-app-2243621?cid=internal_mcdrecs_17102021_cna#mdcrecs_s> |
| ChannelNewsAsia | 2 | <https://www.channelnewsasia.com/cna-lifestyle/persistent-boy-steals-show-papal-audience-2257016> |

APPENDIX B

Link to download the Stanford Corenlp:

[Release History - CoreNLP (stanfordnlp.github.io)](https://stanfordnlp.github.io/CoreNLP/history.html)

REFERENCES

[1] "langdetect", PyPI, 2021. [Online]. Available: https://pypi.org/project/langdetect/. [Accessed: 24- Oct- 2021].

[2] "Universal POS tags", Universaldependencies.org, 2021. [Online]. Available: https://universaldependencies.org/u/pos/. [Accessed: 24- Oct- 2021]Conference Name:ACM Woodstock conference

[2] "Penn Treebank P.O.S. Tags", Ling.upenn.edu, 2021. [Online]. Available: https://www.ling.upenn.edu/courses/Fall\_2003/ling001/penn\_treebank\_pos.html. [Accessed: 24- Oct- 2021].