






Attn: Dr. Sun Aixin



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
Lai Ming Hui U1922654F	 23/10/2021
Lim Shihao U1823110B	 23/10/2021
Hoy Di Sheng Max U1922110F	 23/10/2021
Renice Loh Yi Xuan U1822247D	 23/10/2021
Chong You Min U1821078C	 23/10/21

Natural Language Processing

Group 37

Lai MingHui
U1922654F
LAIM0012@e.ntu.edu.sg

Hoy Di Sheng Max
U1922110F
MHOY001@e.ntu.edu.sg

Chong You Min
U1821078C
CH0033in@e.ntu.edu.sg

Lim Shihao
U1823110B
SLIM129@e.ntu.edu.sg

Renice Loh Yi Xuan
U1822247D
RLOH004@e.ntu.edu.sg

1.1 Tokenizing and Stemming

In Section 1.1, the task is to find out the top 10 most frequent words of the reviews of two randomly selected businesses B_1 and B_2 , excluding stop words, before and after performing stemming.

1.1.1 Tokenizing The first step to do is tokenizing which is a fundamental step in NLP. It is the process of breaking down a given text into smaller units such as words or terms called a token. This is necessary so that the meaning of the text can be easily understood by analyzing the words present. There are multiple ways to perform Tokenization. In this project, we use regular expression and the library Natural Language Toolkit (NLTK). Using regular expression, we change the text reviews into lower case and remove white space and newline. From the NLTK library, we use `sent_tokenize()` method to split paragraphs into sentences and `word_tokenize()` method to split a sentence token into individual word tokens. Now the tokens are ready to be used to create a vocabulary for Stemming. Vocabulary refers to a set of unique tokens in the corpus.

1.1.2 Stemming Next, Stemming is performed. Stemming is the normalization of words which means reducing a word to its root form. This allows the text to be easily interpreted by the machine. The algorithm works simply by cutting the suffix or prefix from the word. PorterStemmer() is used to do stemming. However, we only stem words that are not in the stop word list. Stopwords are frequently used words in a language. In this project, we use the list of stopwords stored in NLTK. They are removed as they do not add value to the meaning and may skew the count of frequency as they are words that are seen often. By removing these words, results will be more accurate.

1.1.3 Experimental Results Below show the results of the experiments.

Word	Frequency
beer	87

great	71
place	66
games	51
selection	49
hour	47
happy	46
wine	44
bar	41
downtown	34

Table 1.1.1: Top 10 most frequently words used in reviews of Business B_1 before stemming

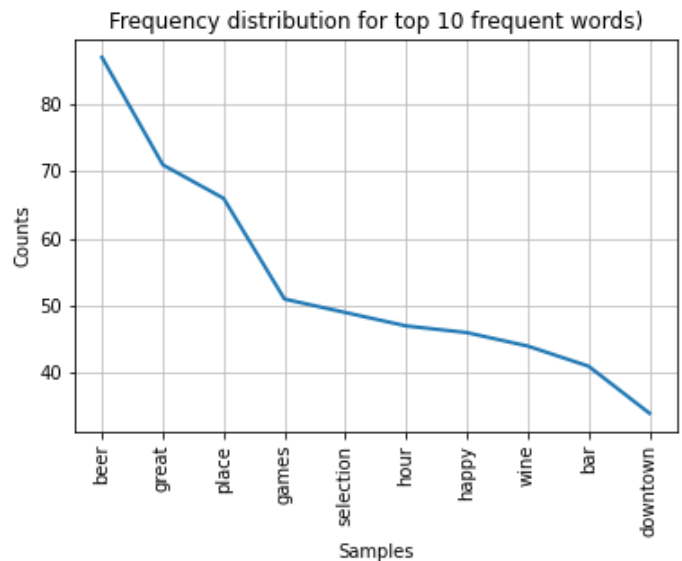


Figure 1.1: Frequency Distribution of top 10 most frequently words used reviews of B_1 before stemming

Word	Frequency
beer	113
place	71
great	71
game	65
select	51
wine	51
hour	50
happi	46
bar	46
downtown	34

Table 1.1.2: Top 10 most frequently words used in reviews of Business B₁ after stemming

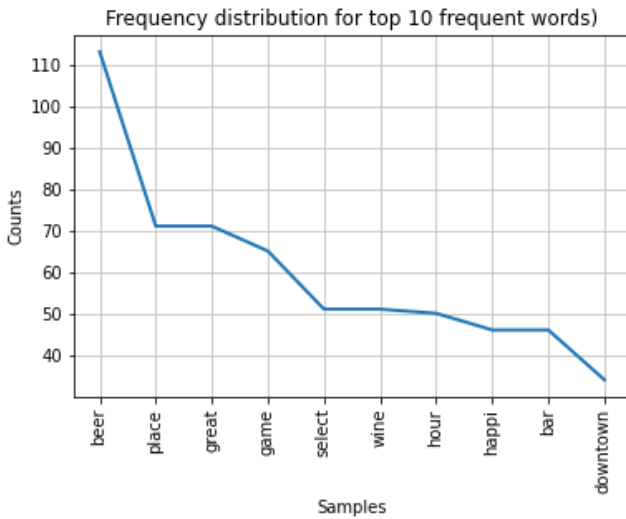


Figure 1.2: Frequency Distribution of top 10 most frequently words used reviews of B₁ before stemming

Word	Frequency
tint	112
great	71
service	65
car	59
recommend	45
express	37
place	36
auto	35
work	33
done	32

Table 1.1.3: Top 10 most frequently words used in reviews of Business B₂ before stemming

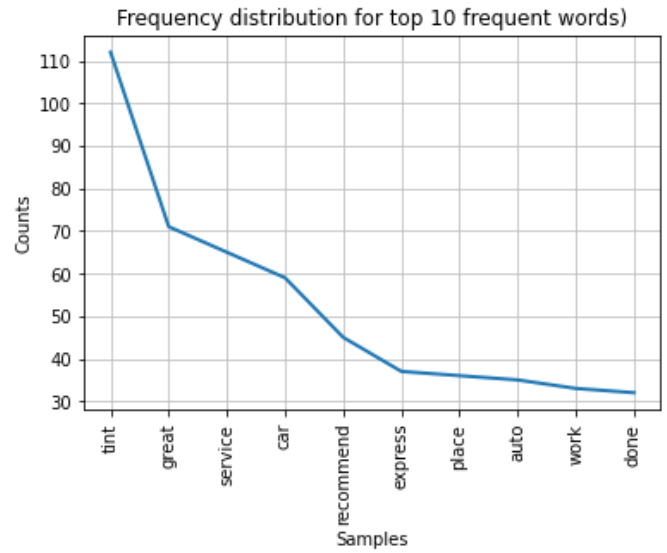


Figure 1.3: Frequency Distribution of top 10 most frequently words used reviews of B₂ before stemming

Word	Frequency
tint	155
great	71
servic	70
car	68
recommend	53
wait	44
price	40
window	39
express	38
place	37

Table 1.1.4: Top 10 most frequently words used in reviews of Business B₂ after stemming

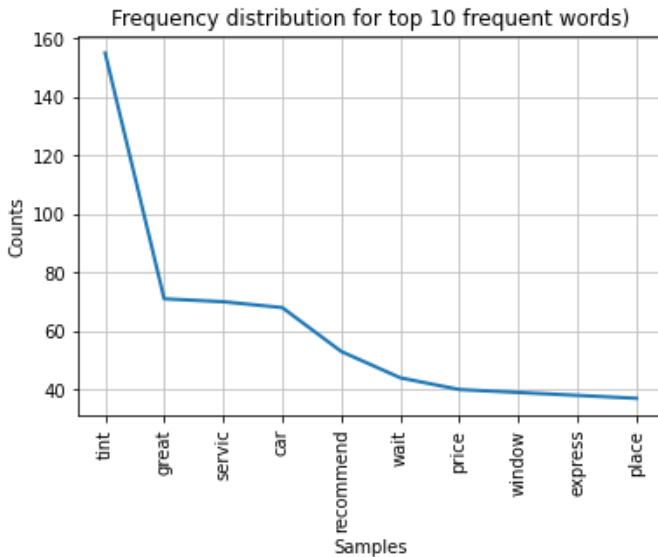


Figure 1.3: Frequency Distribution of top 10 most frequently words used reviews of B₂ after stemming

1.1.4 Analysis From the experimental results, we can see that tokenization and stemming allows us to analyze the reviews of the businesses more precisely by finding which kind of words appear the most. While stemming is very simple and easy to carry out, it can lead to overstemming or understemming as it is based on heuristics. Overstemming occurs when a word is excessively reduced. This can lead to meaningless stems, in which the word's original meaning is lost. Consequently, it can result to words being resolved to the same stems when they should not be. On the other hand, understemming occurs when we have a number of words that are forms of one another but do not resolve to the same stem. One solution is lemmatization. Unlike stemming, lemmatization follows a more complex algorithm and reduces the inflected words properly, ensuring that the root word belongs to the language.

1.2 Parts Of Speech (POS) Tagging

POS tagging is one of the most important and frequently used technique in natural language processing. Many other techniques in NLP relies on knowing the role of a word in a piece of text to perform further processing, such as grammar parsing and syntax tree parsing. However, POS tagging is not a trivial task, as a word can play different roles or have different meanings based on its context and position in a piece of text. The method of POS tagging can produce different results with varying degrees of accuracy as well. Fortunately, there are many libraries and APIs available that have already POS tagging methods and other NLP techniques. In this section, we will demonstrate some examples of POS tagging methods that are implemented in the Python libraries, **NLTK** and **spacy**, and examine the results produced from these libraries and different POS tagging methods.

1.2.1 POS tagging methods. Before we perform the experiments, we need to decide on which POS tagging method to use from each NLP library. For the purpose of this report, we have chosen to use three POS tagging methods: the NLTK's default tagger, NLTK's Trigram 'n' Tag tagger, and the Spacy library's tagger.

NLTK provides a default tagger that is used by calling the **pos_tag()** function and passing it a list of tokens produced from splitting a piece of text into its constituent words. Under the hood, the default function uses a pre-trained **Perceptron tagger**. Besides the default tagger, NLTK also provides classes that implements other POS taggers, such as Brill tagger and **Trigram'n'Tag (TnT) Tagger**. However, NLTK does not provide pre-trained version of those taggers out of the box and requires the user to train them before use. To demonstrate POS tagging for this section of the report, we will use the default tagger and the TnT tagger of NLTK. We chose these two taggers as the Perceptron and TnT taggers implement different techniques and will produce differing results that we can examine. Besides the NLTK methods, we have also opted to use the POS tagger used in the **spacy** library. The spacy POS tagger is popular due to its accuracy and ease of use and is written using the latest research and innovations in NLP. Although spacy is more suited for industrial or learning applications, we decided to use it as another point of comparison in POS tagging results.

1.2.2 Experiments. The experiment for this section is performed by extracting five random review texts from the dataset and applying POS tagging on each of those sentences using the taggers mentioned in section 1.2.1. For this section of the report, our main objective is to examine the differences in the results produced by the selected tagging methods. The main steps for the experiment are as follows:

1. Extract 5 random sentences from the dataset
2. Tag the sentence by hand for point of comparison
3. Perform text cleaning on the sentences
4. Train any taggers, if necessary, on a corpora
5. Apply POS tagging methods from NLTK and Spacy on each of the sentences
6. Compare and discuss the differences in POS tagging results

As the TnT tagger from NLTK is not trained out of the box, we used the treebank corpora from NLTK to train the tagger before applying it in the experiment. The default tagger in NLTK and spacy do not need training as they already uses pre-trained models.

The code for each method of POS tagging is shown below:

```
nlp = spacy.load("en_core_web_sm")
doc = nlp(text)
I
tagged_tokens = [(token.text, token.tag_) for token in doc if token.tag_ != '_SP']

tags_per_word_spacy = {token.text:set() for token in doc if token.tag_ != '_SP' }

for token in doc:
    if token.tag_ == '_SP':continue
    else: tags_per_word_spacy[token.text].add(token.tag_)

return tagged_tokens, tags_per_word_spacy
```

Figure 1.1 Code for POS tagging with spacy

```
tokens = text.split()
pos_method = pos_method or nltk.pos_tag
tagged_tokens = pos_method(tokens)

vocab = sorted(set(tokens))
tags_per_word = {word:set() for word in vocab}
I
for word_tag in tagged_tokens:
    tags_per_word[word_tag[0]].add(word_tag[1])

return tagged_tokens, tags_per_word
```

Figure 1.2 Code for POS tagging with NLTK (pos_method is either nltk.pos_tag() or the TnT tagger's tag() function)

1.2.3 Results. The following tables show some of the similarities and differences in POS tagging results between each method used. All taggers use the universal tag set to tag each word. Each sentence is preprocessed to have all punctuations removed and converted entirely to lower case in step 2 of the experiment steps.

Although the experiments was repeated on 5 separate sentences, we will only show the results of threes entences for brevity.

Table 1.2.1: Tagging results for sentence 1

Sentence Words	Human Tagged	Spacy tags	NLTK default tagger	NLTK TnT tagger
store	NN	NN	NN	NN
is	VBZ	VBZ	VBZ	VBZ
clean	JJ	JJ	JJ	VB
and	CC	CC	CC	CC
toppings	NNS	NNS	NNS	Unk
look	VBP	VBP	VBP	NN
yummy	JJ	JJ	RB	Unk
Num. matches	7	7	6	3

Table 1.2.2: Tagging results for sentence 2

Sentence Words	Human Tagged	Spacy tags	NLTK default tagger	NLTK TnT tagger
this	DT	DT	DT	DT
one	NN	NN	CD	NN

Sentence Words	Human Tagged	Spacy tags	NLTK default tagger	NLTK TnT tagger
is	VBZ	VBZ	VBZ	VBZ
not	RB	RB	RB	RB
up	IN	IN	RB	RB
to	TO	IN	TO	TO
par	NN	VB	VB	NN
Num. matches	7	5	3	4

Table 1.2.3: Tagging results for sentence 3

Sentence Words	Human Tagged	Spacy tags	NLTK default tagger	NLTK TnT tagger
she	PRP	PRP	PRP	PRP
made	VBD	VBD	VBD	VBD
us	PRP	PRP	PRP	PRP
feel	VB	VB	VB	VBP
very	RB	RB	RB	RB
comfortable	JJ	JJ	JJ	JJ
and	CC	CC	CC	CC
was	VBD	VBD	VBD	VBD
not	RB	RB	RB	RB
pushy	JJ	JJ	JJ	Unk
Num Matches	10	10	10	8

1.2.4 Analysis. From the above tables, we can see that the pretrained taggers from NLTK and spacy are the most accurate, while the TnT taggers lags in many of the experiments. The spacy tagger's accuracy can be attributed to it being specifically engineered for production applications and thus is made to be as accurate as possible. NLTK's default tagger comes a close second to spacy, thanks to its pre-trained perceptron tagger under the hood. Unlike spacy, NLTK is not built specifically for production use, and thus may not be trained with as vast a dataset as spacy's model is. The most interesting result comes from the TnT tagger. This tagger does not perform as well as the other two, and its weakness is apparent in its results. In NLTK, the 'Unk' tag is assigned to words that the tagger had not been trained to recognize. Since this tag shows up sometimes in the TnT tagger's results, it can mean that the corpora that it is trained with is limited in its vocabulary. The TnT tagger also misclassify words more often that the other taggers. This shows that the training dataset is very important in producing an accurate tagger, as the tagger can only learn from patterns and words in the dataset. A dataset that is limited will result in a tagger that is performs poorly with inputs that it has not encountered before.

1.3 Writing Style

An author's writing style is crucial in helping the reader understand the meanings and ideas behind a piece of text. Depending on the word choice, tone, and use of formal or

informal language, different writing styles may convey a specific mood to its readers.

In this section, we sampled two posts from StackOverflow, two posts from Hardwarezone, and two news articles from ChannelNewsAsia to compare the different writing style of each text. [5][6][7][8][9][10]

1.3.1 Extraction Method. Firstly, extraction of the contents from each sample was carried out using BeautifulSoup, a Python library used for scraping data from HTML and XML files, and *urllib.request* for opening URLs. With the URL for each sample, we retrieved the respective contents and identified the attribute class where the posts or articles is under. Each website contains a different structure and attribute name, hence we fine-tuned the extraction process by specifying the respective attribute names of the posts in each website. Subsequently, we used *soup.find()*, *find_all()*, and *get_text()* to retrieve the data from each sample and store it into a dataframe.

```
try:
    page = urlopen(Request(url, headers=hdr))
except HTTPError as e:
    print(e.fp.read())
content = page.read()
soup = BeautifulSoup(content, 'lxml')
```

Figure 1.3.1: Implementation of a BeautifulSoup object

1.3.2 Pre-processing. Some posts were found to contain tabs, extra whitespaces, and newlines, therefore we removed them using regular expression and substituting with a single whitespace.

1.3.3 Upper casing of Start Word. In a typical sentence, the first word is always capitalized to signal the start of a new sentence. Hence we made use of regular expression and *split()*, to separate a piece of text by sentence and subsequently extract the first words of each line.

```
def first_word(text):
    pattern = r"(?<!\w\.\w.)(?<![A-Z][a-z]\.)(?<=\.|\?)\s"
    text = re.split(pattern, text)
    sents = [sent for sent in text]
    first_words = [sent.split()[0] for sent in sents]
    return first_words
```

Figure 1.3.3: Implementation of Extracting first words

1.3.4 Results. With reference to Figure 1.3.4, we observed that sentences from StackOverflow and ChannelNewsAsia have capitalized first words for all its sentences. On the other hand, Hardwarezone had a mix of capitalized first words and sentences containing only lower-cased words.

site	first_words
StackOverflow	[Having, This, People, How, Simple]
StackOverflow	[I, Can]
Hardwarezone	[I, Other, My]
Hardwarezone	[cet87, Click, yes, and, the, instead]
ChannelNewsAsia	[TARTU,, Sodium-ion, Scientists, "Peat, The, T...
ChannelNewsAsia	[HONG, Tropical, As, Hong, The, Schools, On, A...

Figure 1.3.4: List of first words from each post/article

1.3.5 Grammar Used. LanguageTool is an open-source library that supports grammar checking and detecting any possible errors. With the use of LanguageTool, we observed the grammar usage in each text with the following method:

```
matches = tool.check(text)
```

LanguageTool can match possible errors such as misspellings, grammar, casing, etc., and returns a match object with the violated rule, suggestions for correction, the portion of the text where error was detected, etc.

From the tables below, StackOverflow mostly contain misspellings that are commonly used in programming but is classified as an error in the English language. The word “MultiIndex” for example was misidentified as a misspelling. Other text such as code segments from the post may also be incorrectly matched.

```
Simple dataframes can be put together, e.g.: import pandas as pd
df = pd.DataFrame({'user': ['Bob', 'Jane', 'Alice'],
                  'income': [40000, 50000, 42000]})
```

Figure 1.3.5: Code segment from a StackOverflow post

Hardwarezone was found to contain various slangs, abbreviations, and use of informal writing. For example, the abbreviation “oyk” stands for “Ong Ye Kung” but was identified as a misspelling. Grammar errors were also detected as well as lower-cased start words.

As for ChannelNewsAsia, punctuation errors in metrics were detected as a space should be included after the number and the abbreviated unit of measurement. Since LanguageTool does not seem to support British English, American English was used in this experiment instead. Therefore, words spelled in British English were matched.

Table 1.3.5.1: Errors matched for StackOverflow		
Words/Phrase Matched	Rule Category	Suggestions
dataframes	TYPOS	data frames
datetime	TYPOS	date time
datastructure	TYPOS	data structure
MultiIndex	TYPOS	Multitude, Multiplex, etc.

Table 1.3.5.2: Errors matched for Hardwarezone

Words/Phrase Matched	Rule Category	Suggestions
dun	CONFUSED_WORDS	don't
My goals is	GRAMMAR	My goals are
and	CASING	And
oyk	TYPOS	OK, oak, CYK, etc.
the	CASING	The
sg	TYPOS	SG, so, St, kg, etc.
covid	GRAMMAR	Covid-19, COVID-19
instead	CASING	Instead

Table 1.3.5.3: Errors matched for ChannelNewsAsia

Words/Phrase Matched	Rule Category	Suggestions
TARTU	TYPOS	TART, TARTS, etc.
northern Europe	CASING (Proper noun)	Northern Europe
flavour	TYPOS (British English)	flavor
whisky	TYPOS (British English)	whiskey
480km	PUNCTUATION (unit space)	480 km
epicentre	TYPOS (British English)	epicenter
kilometres	TYPOS (British English)	kilometers
460mm	PUNCTUATION (unit space)	460 mm

1.3.6 Discussion. Overall, we observed that text from StackOverflow contains codes, and the sentences generally follow good grammar. POS tagging and tokenization may be applied to parts of text that are non-code to ensure better accuracy. Posts from Hardwarezone does not always follow good grammar and may contain slangs hence pos tagging may not always apply to every word. Lastly, ChannelNewsAsia has relatively formal writing style does not contain grammatical errors.

1.4 Most frequent < Noun - Adjective > pairs for each rating

In this section, we are tasked to randomly select 50 reviews (one from each business) of rating 1, 20 reviews of rating 2, 3, 4 and 5 and extract the top 10 most frequent noun-adjectives pair. A noun is a term that refers to a person, place, thing, event, substance, or quality, according to the Cambridge dictionary. A word that describes a noun or pronoun is defined as a 'adjective'. In this situation, a noun-adjective pair is defined as a pair that can provide customers with the most relevant information about a restaurant and assist them in determining whether the restaurant is suitable for them to dine in.

```
noun_adj_pair_list = []
for i in randomReview["text"]:
    doc = nlp_sm(i)
    for i,token in enumerate(doc):
        if token.pos_ not in ('NOUN','PROPN'):
            continue
        for j in range(i+1,len(doc)):
            if doc[j].pos_ == 'ADJ':
                noun_adj_pair_list.append((str(token),str(doc[j])))
            break
noun_adj_pair_list
```

Figure 1.4.1: Implementation of Extracting Noun-Adjective Pairs

An example of identifying a most common noun-adjective pair: Servers – Careful

Which can be found in the full text reviews:

I will give this place 0 star if this review has an option for 0. Worst **servers** and customer service ever! I will never come back to this place. I went here with my coworkers and they sat us by the pathway and each of the servers who passed by had no consideration whatsoever and kept bumping on my chair. When I told one of the servers to tell the others to be **careful**, one of the servers girl just said "well this is the pathway for the servers, we can't do anything".

1.4.1 Identification of Top 10 Most Frequent Noun-Adjective Pairs for Rating 1

Top 10 most frequent noun-adjective pairs:

1. Service, Bad
2. Tables, Dirty
3. Color, Dark
4. Service, Rude
5. Service, Upset
6. Restaurant, Empty
7. Night, Quiet
8. Food, Cold
9. Appointment, Disappointing
10. Wings, Small

1.4.2 Identification of Top 10 Most Frequent Noun-Adjective Pairs for Rating 2

Top 10 most frequent noun-adjective pairs:

1. Room, Good
2. Minutes, Busy
3. Food, Same
4. Service, Same
5. Drink, Same
6. Things, Better
7. Lady, Good
8. Bread, Good

9. Sandwich, Good
10. People, Young

1.4.3 Identification of Top 10 Most Frequent Noun-Adjective Pairs for Rating 3

Top 10 most frequent noun-adjective pairs:

1. Reviews, Good
2. Place, Good
3. Rebate, Best
4. Service, Good
5. Breakfast, Good
6. Room, Clean
7. Cheese, Good
8. Sauce, Bland
9. Pizza, Good
10. Drinks, Multiple

1.4.4 Identification of Top 10 Most Frequent Noun-Adjective Pairs for Rating 4

Top 10 most frequent noun-adjective pairs:

1. Food, Great
2. Tuna, Generous
3. Services, Top
4. Place, Cool
5. Beer, Great
6. Location, Excellent
7. Sushi, Regular
8. Tempura, Crisp
9. Menu, Good
10. Food, Delicious

1.4.5 Identification of Top 10 Most Frequent Noun-Adjective Pairs for Rating 5

Top 10 most frequent noun-adjective pairs:

1. Place, Pretty
2. Salad, Impressive
3. Items, Special
4. Tonight, Outstanding
5. Food, Many
6. Staff, Great
7. Restaurant, Huge
8. Service, Excellent
9. Treatment, Amazing
10. Parking, Amazing

However, extracting noun-adjective pairings does not provide useful information to customers about which restaurant is best for them or whether the feedback is positive or negative. For example, the noun-adjective pair "Restaurant" and "Huge" is ambiguous since it can convey both positive and negative feedback because some people prefer tiny restaurants while others prefer large restaurants.

2 Extraction of Indicative Adjective Phrases

A useful application of natural language processing is the extraction of sentiment and key words from a piece of text. This technique can be used by review aggregation websites to create a summary of a business based on the reviews left by its customers on the website. Such summaries can aid future customers to avoid terrible businesses quickly without scouring through pages of reviews, and it turn helps the website grow by being a useful aid that visitors will return to.

In this section of the assignment, we will attempt to perform such a task using indicative adjective phrases. In this report, an indicative adjective phrase is an adjective phrase that shows up in reviews of a business more frequently than other businesses. Using these phrases, we will be able to create a summary of features that distinguishes that business from the rest.

Adjective phrases are useful as they are a way customers describe a business' qualities. If a business gets many reviews containing many negative adjective phrases compared to other businesses, then we can conclude that the business is one of the worst among the reviews. Adjective phrases can also be extracted along with their described nouns to give an idea of the business' range of products and their qualities. For example, some restaurants may receive criticisms on some items on their menu, but compliments on other items. Extracting these comments allow us to have an idea on what the restaurant offers and how good they are. Thus, extracting adjective phrases along with their described nouns will be useful in finding distinguishing features of a business.

2.1 Grammar of Adjective Phrases. An adjective phrase is a phrase that has an adjective, which describes a noun or pronoun, as the head. For example, take the sentence "The spaghetti tastes **absolutely terrible**." In this sentence, the phrase "**absolutely terrible**" is an adjective phrase, and it describes the spaghetti. Adjective phrases come in two forms, **predicative** and **attributive**.

In **predicative phrases**, the adjective appears outside of the noun term it is describing. For example, "The car was **very dirty**". In such phrases, the adjective is linked with the noun with a verb, such as "was" or "is". These phrases are also called head-final or head-initial, as the adjective appears after or before the noun term.

In **attributive phrases**, the adjective appears in the noun phrase that it is describing, such as "I ate some very excellent fries here." The adjective phrase "very excellent" appears behind the noun "fries", and together they form a noun phrase. Since the adjective appears in the middle of the noun term, it is also referred to as a head-medial adjective phrase.

Adjective phrases can be represented with a syntax tree, in head-initial, head-medial and head-final form, as shown in the pictures below. This is very useful later when we formulate a context-free grammar to parse review texts and extract different kinds of adjective phrases using the NLTK toolkit.

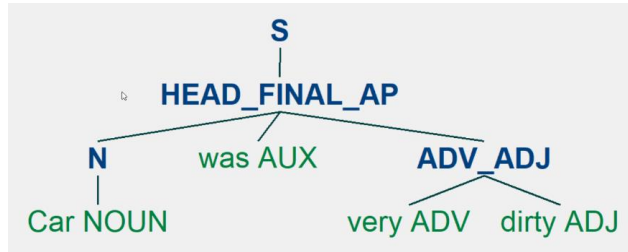


Figure 2.1: Head final AP in “Car was very dirty.”

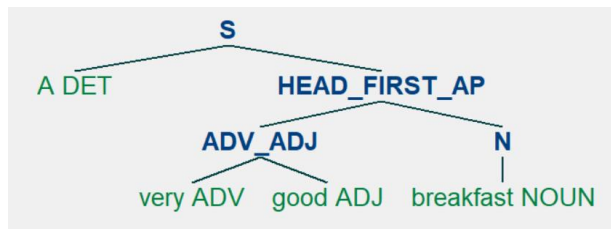


Figure 2.2: Head-First AP in “A very good breakfast.”

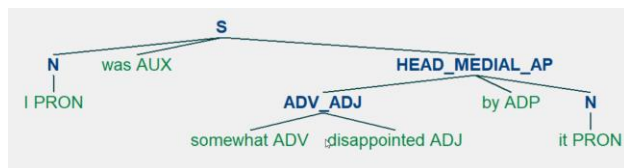


Figure 2.3: Head Medial AP in “I was somewhat disappointed by it.”

2.2 *Extracting adjective phrases.* The steps involved in extracting indicative adjective phrases are as follows:

1. Formulate a context free grammar to parse review texts for adjective phrases
2. For a random business id, collect all adjective phrases that appears in the business’ reviews.
3. For each adjective phrase, extract the noun that is modifying. Each adjective phrase and noun are counted as a pair.
4. Count the total number of times a pair has appeared in a business’ reviews
5. Use the extracted adjective phrases and nouns to come up with the business’ distinguishing features.
6. Apply steps 2-5 to two other random businesses and compare the results.

In step 1, the context free grammar is based on the head-final, head-medial and head-initial adjective phrase syntax trees. Nouns and pronouns are grouped together and labelled as their own entity to make extracting them easier. This technique is used with adverb and adjective pairs as well. The grammar is written to be parsed by NLTK’s **RegexParser class**, and thus is written in a similar style to regular expressions. The grammar used is as follows:

```

N:          {<NOUN>+|<PRON><NOUN>+}

V:          {<PART>?<VERB>}

ADP_DET:    {<ADP>?<DET>?}

ADV_ADJ:    {<PART>? (<ADV>+<ADJ>) | (<ADJ>+<ADV>) | <ADJ>}

ADJ_N:      {<DET>?<ADV_ADJ><N>+}

HEAD_FIRST_AP:  {<ADJ><ADP_DET>?<SCONJ>?<N>}

HEAD_MEDIAL_AP: {<ADV_ADJ><ADP_DET>?<SCONJ>?<N>}

HEAD_FINAL_AP:  {<N>?<AUX>+<V>*<ADP>*<ADV_ADJ><V>?}

```

For this section, three random businesses are picked from the dataset to be analyzed. The 10 most frequent adjective phrases are extracted from the entirety of their reviews and used to find each of the business’ distinguishing features.

The first business to be analyzed has the id ‘oICXzFAaUMrYGzjRWmkw4Q’ and will be referred to in this section as business 1. The 10 most frequent adjective phrases and nouns in the business’ reviews are shown below:

Table 2.1: Most frequent Adjective Phrase for business id oICXzFAaUMrYGzjRWmkw4Q

Adjective Phrase	Frequency
vietnamese restaurant	12
vietnamese food	10
imperial roll	5
fresh roll	5
extra noodle	4
vietnamese coffee	4
good food	4

From this table, we can discern that this business is a Vietnamese restaurant, and it has received many comments about its food. However, most of the adjective phrase does not actually describe the quality of the restaurant’s food and service. This can be due to the overwhelming number of references to food compared to descriptions of quality in business 1’s reviews. To gain a better understanding of the business, we used a word cloud formed from the extracted adjective phrases to show the most prominent words among the reviews.

3.1 Necessary resources

There are important resources that are needed to make this application work. These resources include libraries such as pandas, nltk, SentimentIntensityAnalyser.

3.2 Sentiment Analysis Tool

VADER (Valence Aware Dictionary for Sentiment Reasoning) is a text sentiment analysis model that considers both the polarity (positive/negative) and the intensity (strong) of emotion. It's included in the NLTK package and may be used on text data directly.

3.3 Data Extraction

The data of the businesses can be found in the reviewSelected100.json file. The 'review_id' and 'text' columns contain important data that is required. In the pre-processing stage these columns will be filtered out to obtain the necessary data. Firstly, the data will need to extract the indicative adjective phrase which is an adjective phrase that shows up in the review section of a business. Using the library, these phrases will be processed through a lexicon and able to create a rating for the reviews that can be viewed at a quick glance.

3.4 Scoring

Once the data is imported and cleaned, we will be able to implement the sentiment expression model. The sentiment intensity analyzer will be used to check the polarity scores. When the compounded polarity score is greater than 0.05 it will return a positive score for the expression. When the compounded polarity score is lesser than -0.05 it will return a negative score for the expression. When the compounded polarity score does not fall between these ranges it will return a neutral score for the expression.

3.5 Summary

At the end, when all the scores are calculated the sentiment values will be saved and output back into the csv file with a sentiment analysis rating (positive, neutral, negative).

```
# implement the sentiment expression model
def fetch_sentiment(text):
    # applying the model on each review text and get it's expression score
    polarity_scores = analyser.polarity_scores(text.lower())
    # if the expression score > 0.05 then assign postive expression
    if polarity_scores['compound'] >= 0.05:
        # send back results
        return 'Positive'
    # if the expression score < -0.05 then assign negative expression
    elif polarity_scores['compound'] <= -0.05:
        # send back results
        return 'Negative'
    # if the expression score = 0 then assign neutral expression\
    else:
        # send back results
        return 'Neutral'
```

Figure 3.4 Scoring system

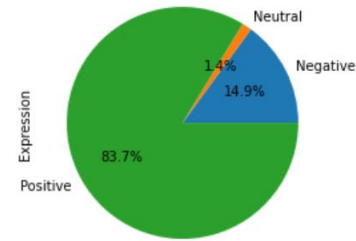


Figure 3.5 Expression distribution

REFERENCES

- [1] Bird, Steven, Edward Loper and Ewan Klein (2009). Natural Language Processing with Python. O'Reilly Media Inc.
- [2] Edward Loper and Steven Bird. Source code for nltk.tag. Retrieved October 23, 2021 from http://www.nltk.org/_modules/nltk/tag.html
- [3] Craig Shives. Adjective phrase. Retrieved October 23, 2021 from https://www.grammar-monster.com/glossary/adjective_phrases.htm
- [4] Anon. 2015. A good POS tagger in about 200 lines of python. (March 2015). Retrieved October 23, 2021 from <https://honnibal.wordpress.com/2013/09/11/a-good-part-of-speechpos-tagger-in-about-200-lines-of-python/>
- [5] Marius. How to make good reproducible pandas examples. (November 2013) Retrieved from: <https://stackoverflow.com/questions/20109391/how-to-make-good-reproducible-pandas-examples>
- [6] Charles Anderson. List of lists changes reflected across sublists unexpectedly. (October 2008) Retrieved from: <https://stackoverflow.com/questions/240178/list-of-lists-changes-reflected-across-sublists-unexpectedly>
- [7] Lareina. Hardwarezone Forum (October 2021) Retrieved from: <https://forums.hardwarezone.com.sg/threads/discuss-naive-sinkies-think-they-are-successful-in-life-by-having-these-3-items-in-sg.6624726/post-137385099>
- [8] Radish. Hardwarezone Forum (October 2021) Retrieved from: <https://forums.hardwarezone.com.sg/threads/well-done-ong-ye-kung-usa-classify-singapore-as-cat-iv-very-high-risk.6624719/page-2#post-137385434>
- [9] Andrius Sytas. Energy from bogs: Estonian scientists use peat to make batteries. (October 2021) Retrieved from: <https://www.channelnewsasia.com/business/energy-bogs-estonian-scientists-use-peat-make-batteries-2236431>
- [10] ChannelNewsAsia. Storms force Hong Kong to shutter twice in a week. (October 2021) Retrieved from: <https://www.channelnewsasia.com/asia/hong-kong-storm-cyclone-kompasu-weather-t8-warning-2238766>