

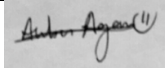




Attn: Dr. Sun Aixin



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
Anbu Agana N Annadurai	 /25-10-2021
Chiramal Esther	 /25-10-2021
Lee Si Si, Cecilia	 /25-10-2021
Low Shi Min	 /25-10-2021
Min Thant Ko	 /25-10-2021

Contributions

Part	Done by
1.2 Tokenization and Stemming	Chiramal Esther, Min Thant Koh
1.3 POS Tagging	Low Shi Min
1.4 Writing Style	Anbu Agana N Annadurai
1.5 Noun Adjective Pair	Lee Si Si Cecilia
2 Extraction of indicative Adjective phrases	Low Shi Min, Lee Si Si Cecilia
3 Application	Chiramal Esther, Anbu Agana N Annadurai
Others	Collective Effort

CZ4045 Natural Language Processing - Assignment 1

G33

Anbu Agana N Annadurai
U1822360E
Nanyang Technological University
anbuagan001@e.ntu.edu.sg

Chiramal Esther
U1821246K
Nanyang Technological University
esther003@e.ntu.edu.sg

Lee Si Si Cecilia
U1821291A
Nanyang Technological University
lees0185@e.ntu.edu.sg

Low Shi Min
U1822167C
Nanyang Technological University
lows0049@e.ntu.edu.sg

Min Thant Ko
U1821227E
Nanyang Technological University
mint0008@e.ntu.edu.sg

ACM Reference Format:

NTU CZ4045 Natural Language Processing: Assignment Report. In Proceedings of CZ4045 Natural Language Processing Course Assignment, Nanyang Technological University. Nanyang Technological University, Singapore, 7 pages.

1 Dataset Analysis

1.1 Background

The data that would be used is a subset of Yelp's businesses, reviews, and user data that has been made publicly available for use for personal, educational, and academic purposes. The dataset used in this assignment contains 15,300 reviews, and is presented in a JSON format. It contains the following components: review id, user id, business id, stars, date, text, useful, funny, cool.

1.2 Tokenization and Stemming

1.2.1 Tokenization

Tokenization is the process of breaking down strings of text into small units known as tokens [1]. Reviews from a randomly chosen business from the dataset were tokenized, using the NLTK (Natural Language Toolkit) tokenizer package [2]. The resulting word frequency distributions of the chosen reviews were presented in a table.

1.2.2 Stemming

Stemming is the process of reducing a word to its root form [3] and it was applied to the dataset using the NLTK Stemmers package. Most of the words obtained were valid English words. However, in some cases after stemming, there were some words that lost their meanings. The table below shows an example, where the word 'fries' was reduced to 'fri', an invalid English term given the context of the sentence. This is one of the disadvantages of Porter's Stemming as it does not use a dictionary to identify words. Hence, there is a chance for words to become invalid after stemming.

Table 1: Incorrect stemming Results

Original Word	Word Obtained through Stemming
fries	fri
restaurant	restaur

1.2.3 Improvement

Overstemming can be minimised using lemmatization of the dataset. Lemmatization takes into consideration the overall context of the text and computes an output based on a word's part of speech.

1.2.4 Results

The table below depicts the word frequency distributions of 2 randomly chosen businesses after tokenization and stemming. Words such as 'shooting' were stemmed to 'shoot', increasing the frequency of the stem word.

Table 2: Stemming Results

Before Stemming (Count)	After Stemming (Count)
Dishes (58)	Dish (72)
Shoot (42)	Shoot (152)

However, there were instances of failed tokenization, resulting in outputs such as `table style="display:inline"`.

1.2.5 Discussion on Findings

A comparison plot for the change in word lengths before and after stemming is shown below. According to the data, there was a clear increase in the frequency of shorter words after stemming. This shows that words in a text had been reduced to their stem form, thus proving that stemming was demonstrated.

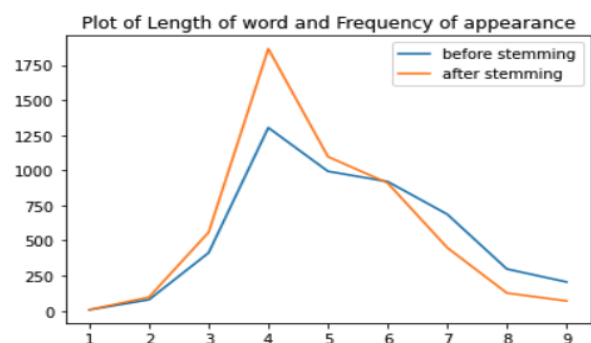


Fig 1. Word length frequency plot

1.3 POS Tagging

Part-of-Speech (POS) Tagging is the practice of assigning a token (word) in a text corpus to a corresponding component of a speech tag based on its context and definition [4].

1.3.1 NLTK and spaCy

POS Tagging was performed on the same set of five random sentences using two different toolkits, namely Natural Language Toolkit (NLTK) and spaCy [5]. (*Appendix Table 5.1.1*) NLTK is a library for string processing. It takes strings as inputs and outputs strings or string lists. On the other hand, spaCy takes an object-oriented approach. When a text is parsed, spaCy returns a document object [6].

1.3.2 General Observation

Both NLTK and spaCy yielded similar resulting POS tags. The tags obtained might not be entirely identical due to the different tag lists used by both toolkits. However, the tags obtained from both toolkits were consistent in their denotations of meanings. For tokens with completely different tags, spaCy yielded a more accurate result.

The following presents the 2 cases in which NLTK and spaCy yielded different results. The full results obtained can be found in *appendix 5.1.2*.

Table 3: case 1 of a token yielding different tags

Sentence	No judgement here, as I like to lick my plates <u>clean</u> too!	
Word	NLTK	spaCy
clean	VBP	ADJ

Based on contextual knowledge, “clean” is an adjective. Thus, proving spaCy which tagged it as an “adjective (ADJ)” to be more accurate than the “verb, non third-person singular present (VBP)” tagged by NLTK.

Table 4: case 2 of a token yielding different tags

Sentence	We dropped in during show time and since the box office was <u>closed</u> , we decided to sneak into the main show.	
Word	NLTK	spaCy
closed	VBN	ADJ

Based on contextual knowledge, “closed” is an adjective. Thus, proving spaCy which used an “adjective (ADJ)” tag to be more accurate than the “verb, past particle (VBN)” tag by NLTK.

1.3.3 Discussion on Findings

SpaCy supports word vectors unlike NLTK [6]. This denotes that the meaning of the text parsed in spaCy can be mathematically represented, modelling the representational content humans have assigned to the text [7]. This allows for more meaningful results to be obtained. Furthermore, since spaCy uses the latest and best algorithms, a better performance is expected (compared to NLTK). As evident from the results obtained, spaCy outperformed NLTK in POS tagging. This can be further supported by the diagram in *Appendix 5.1.3* whereby spaCy displayed better performance in terms of the execution timing recorded [6]. However, NLTK

has a wider range of languages supported. NLTK supports various languages while spaCy only has statistical models for 7 languages [6]. Nonetheless, as the data used in this assignment is solely in English, the results would not be affected significantly. Therefore, spaCy would generally be a more suitable toolkit for POS tagging.

1.4 Writing Style

Analysis of Writing Style includes studying various criteria related to language structure, lexicon structure and sentence semantics. In this experiment, we will be focusing on writing style analysis using language and lexicon structures.

1.4.1 Approach

4 key criteria were identified to differentiate writing styles and they are as follows:

1.4.1.1 The Average Length of a Sentence (word count):

This was implemented by first tokenizing the text, followed by calculating the average word count. (*Appendix Fig 5.2.2.1*) The average sentence length indicates the complexity of the sentence. This attribute can be translated in understanding Writing Styles as sophisticated or simple, providing some basic level insight into audience awareness of the text.

1.4.1.2 The Average Length of Word (character count):

This was implemented by first tokenizing the text and removing stop-words and punctuations before calculating average word length. (*Appendix Fig 5.2.2.2*) The average word length indicates the sophistication of the vocabulary used. Longer words will imply a richer vocabulary within the text which can be translated to a more technical and advanced language style.

1.4.1.3 Accuracy of Grammar Used:

This was implemented using the `tool.check()` function from the LANGUAGE TOOL PYTHON library [8]. (*Appendix Fig 5.2.2.3*) A match corresponds to a potential grammar error. The grammar check score indicates the tone of language used. A higher score (many grammar errors detected) could imply an informal writing style.

1.4.1.4 Accuracy of Sentence Capitalisation:

This was implemented by first tokenising the text and performing a simple string check. (*Appendix Fig 5.2.2.4*) The score is a measure of how erroneous capitalisation is in the text. This attribute also indicates the formality of the text as poor capitalisation implies an informal and non-academic/official piece of text.

1.4.2 Hypothesis

Our contextual understanding of the style of writing across the different sites allows us to draft hypotheses of our findings. StackOverflow is a forum for developers to exchange programming knowledge. As such, diction used would be highly technical and lines of code would be prevalent. This should result in long average sentence and word lengths. Grammar and capitalisation scores are expected to be high as lines of code do not follow the syntax of proper English.

Hardware Zone is a site which provides a range of post types - forums, articles, etc... Forums have a more informal and simple writing style (shorter average sentence and word lengths) as well as poor grammar and capitalisation (high grammar and capitalisation scores). Informal social interaction is similar to texting with little focus on grammar.

Articles would have writing styles similar to CNA posts (as detailed below).

CNA is a site which publishes news articles. We expect a very formal writing style as well as a relatively complex diction (long word lengths) coupled with perfect grammar as well as perfect sentence capitalization (scores should be 0).

1.4.3 Extracting Text

2 posts from each of the required sites (StackOverflow [9], Hardware Zone [10] and CAN [11]) were manually selected at random. Web scraping was conducted using *BeautifulSoup* to extract the relevant text for all the posts [12]. Division classes were specified to only extract the text body. CNA had blocked web scraping on their sites which prevented the text to be directly scraped from post URL [13]. Hence, CNA posts were manually copied to a word document from which it was read using *docx2txt* [14].

1.4.4 Results and Discussions of Findings

```
Average word count per sentence for Stackoverflow: 28.071
Average word count per sentence for HardwareZone: 19.861
Average word count per sentence for CNA: 24.476
Average word length for Stackoverflow: 5.336
Average word length for HardwareZone: 5.846
Average word length for CNA: 24.476
Grammar Check Score for Stackoverflow: 65.500
Grammar Check Score for HardwareZone: 15.000
Grammar Check Score for CNA: 25.500
Capitalisation Check Score for Stackoverflow: 3.000
Capitalisation Check Score for HardwareZone: 2.000
Capitalisation Check Score for CNA: 5.500
```

Fig 2: Writing Style Criteria Findings

Our findings confirmed our hypotheses to a significant extent. StackOverflow posts indicated a high level of technicality (long average sentences) but reflected poorly on grammar due to lines of codes' (high grammar score) poor adherence to English syntax. CNA posts indicated a high level of formality (long average sentences) and sophistication in diction (long average word lengths) in its writing style. Hardware Zone posts indicated a more informal and simple writing style as expected (short average sentences and words).

However, some of the criteria had unexpected findings which did not support our contextual knowledge. These discrepancies can be attributed to the limitations prevalent in this analysis (Discussed below).

1.4.5 Limitation

Web scraping of the posts cannot guarantee the format of the texts scraped. While the text scraped was cleaned to minimise advertisements and 'recommended posts', this cannot be eradicated completely. Many of the posts still contained unwanted 'noise' text like 'share button' and 'comment' lines [15].

Some posts had unreliable formats which impeded the analysis. This was especially a limitation in StackOverflow posts which prioritise accuracy of code. This content does not follow regular English syntax and grammar which would make using the toolkit libraries of English unreliable. Even for posts of news articles (CNA), sentence structures of the various parts of the articles - headings and subheadings carry a different syntax compared to proper English which would explain the grammar accuracy and capitalisation scores.

These limitations impacted the tokenization of the text as sentence tokens and word tokens became highly inaccurate. As tokenization forms the basis of all the functions to compute the scores for the various criteria, inaccurate tokenization resulted in inaccurate scores. This was evident in how CNA posts were implied to be grammatically inaccurate from the findings when in fact, they were the most accurate.

Semantics of texts was not decomposed for simplicity. This limited the identification of the potential writing styles prevalent (e.g. Passive/active voice, Opinionated and Objective writing styles).

Furthermore, sampling only 2 posts from each site added to inaccuracy as sample size is far too small.

1.4.6 Conclusion

Texts extracted from the posts have to be cleaned based on contextual and structural understanding before analysis. In addition to the current analysis matrix, semantic analysis should be performed to increase accuracy.

Furthermore, the codes and keywords used in these posts do not follow the conventional English syntax. Therefore, POS Tagging and Tokenization cannot be performed directly on the StackOverflow posts. This can be addressed by removing the lines of code from the texts before tokenization and POS Tagging. Alternatively, a separate tokenization algorithm and POS Tagging algorithm that takes into account the corresponding syntax of the various programming languages can be implemented.

1.5 Most Frequent <Noun - Adjective > Pairs

1.5.1 Preprocessing

Minimal preprocessing was performed to maintain the integrity and sentence structure of the text. All the text was converted to lower case for standardisation.

1.5.2 Detection of Noun - Adjective pairs

SpaCy library was chosen to perform dependency parsing on the reviews. Dependency parsing and noun phrase extraction were done first to obtain noun-adjective pairs in each review.

Noun/Pronoun and Adjective pairs directly next to each other were extracted first.

Auxiliary words (e.g. is, are, was, were) were searched and extracted as phrases such as 'Food was great' were present. If such phrases were found, child dependencies were scanned to identify the corresponding adjective phrases. Referring to the previous example, 'Food was great', food would be extracted as a noun whereas great would be extracted as an adjective.

In addition, if the adjective had a preceding adjective modifier (e.g. adverb), it would be extracted as well. For example, 'very happy atmosphere', very happy would be extracted as the adjective.

1.5.3 Results

50 reviews, from different businesses, with a rating of 1, were randomly chosen to extract the top 10 most-frequent noun-adjective pairs.

Table 5: Top 10 most frequent noun-adjective pairs from 50 reviews that have rating '1'

Noun	Adjective	count
money	hard	3
service	horrible	3
tool	hot	3
place	other	3
time	long	2
service	bad	2
cook	new	2
wait	long	2
deal	big	2
information	personal	2

20 reviews, from different businesses, with a rating of 2-5 were randomly chosen to extract the top 10 most-frequent noun-adjective pairs for each of the different ratings respectively. The results can be found in the *Appendix 5.3*.

As visibly evident from the results, higher ratings saw an increase in the frequency of positive noun-adjective pairs.

1.5.4 Discussion on Findings and limitations

It was observed that for sentences that have multiple adjectives, the extractor would only be able to identify and extract the first adjective. For example, 'Quick and attentive service, despite being continuously busy', the pair that was extracted is 'service attentive', whereas 'service quick' was overlooked. Thus, consecutive noun phrases that correspond to the same adjective is a limitation of this extractor.

However, when an auxiliary word is involved in the sentence, it would be able to extract both. For example, 'Environment, and atmosphere was thoroughly enjoyable.', 2 pairs were extracted, 'environment thoroughly enjoyable' and 'atmosphere thoroughly enjoyable'.

It was also observed that when the words are supposed to be hyphenated, and it is an adjective, the extractor would not be able to extract it. For example, 'Potatoes were extremely well seasoned (anymore and it might be too salty), and the crispness of the potatoes was amazing', the adjective 'well-seasoned' was not detected and thus, not extracted.

In conclusion, there are limitations to this extractor. The pairs extracted were mostly accurate. However, due to the small number of reviews that were used during extraction, it is hard to evaluate. Thus, based on the results that were generated, this extractor performed relatively well.

2 Extraction of Indicative Adjective Phrase

According to Cambridge Dictionary, an adjective phrase (ADJP) always has an adjective acting as the head. The adjective phrase may also contain words or phrases before or after the head (modifiers and complements) [16]. Thus, a phrase should have at least 2 words.

An ADJP would be considered as an indicative ADJP for a particular business when it appears often in the reviews of

the particular business, but relatively less often in other reviews for other businesses.

Extraction of indicative adjective phrases would be described in the subsequent sections.

2.1 Preprocessing

Minimal preprocessing was performed to maintain the integrity and sentence structure of the text. Firstly, all HTML tags were removed from the text. Secondly, all emotion icons (emojis) were removed from the text as emoticons were frequently used in the reviews. Thirdly, accented characters were converted. Words with accent marks like "latté" and "café" were converted and standardized to just "latte" and "cafe" using 'unidecode library' [17]. Fourthly, contractions were removed from the text. Contractions are words or combinations of words that are shortened by dropping letters and replacing them with an apostrophe (E.g. 'I'd' was converted to 'I would') using 'contraction library' [18]. This was done such that the parser would catch the syntactic structure of the text more accurately. Next, extra spaces were removed from the text, as spaces could be generated after preprocessing above. Lastly, all text was converted to lower case for standardisation.

2.2 Extraction of ADJP

2.2.1 Approach

To extract ADJP, constituency parsing was used. Constituency Parsing is the process of analyzing the sentences by breaking them down into sub-phrases also known as constituents. These sub-phrases belong to a specific category of grammar like Noun Phrase (NP) and Verb Phrase (VP). Thus, ADJP could be identified and extracted.

2.2.2 Extracting of ADJP

StanfordParser was used for the extraction of ADJP for all the reviews [19]. After the ADJP were identified by constituency parsing, extraction would be done. One of the results of constituency parsing is presented below.

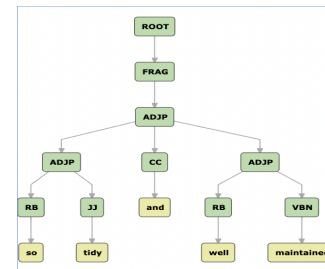


Fig 3: Constituency parsing by Stanford parser

In detail, a Stanford server would be set up in the background. Next, 'nltk.parse.corenlp library' is used to connect to Stanford server and sentences would be parsed. Parsing result will be returned in the data type 'NLTK tree' and it is shown below.

```

(ROOT
  (FRAG
    (ADJP
      (ADJP (RB so) (JJ tidy))
      (CC and)
      (ADJP (RB well) (HYPH -) (VBN maintained))))))

```

Fig 4: NLTK Tree for results of Stanford parsing

As seen, there could be ADJP inside an adjective phrase. Thus, there were 2 different extraction methods implemented, deep extraction and shallow extraction.

2.2.2.1 Deep extraction:

During deep extraction, the deepest adjective phrase would be extracted. For example, the sentence in Figure 3, 'so tidy' and 'well maintained' would be extracted separately even though 'so tidy and well maintained' was considered as an adjective phrase.

Since the sentence was broken down, it was predicted that there would be more identical ADJP. Thus, it would be more useful to identify the frequently used ADJP in the next step. This will be evaluated in section 2.4.2.

Code implementation of Deep extraction:

We would traverse the tree and find ADJP. Once found, we would continue to find ADJP in the ADJP. If there is ADJP inside an ADJP, it will replace the original ADJP. This is done recursively, and the implementation is shown in the code snippet below.

```
def traverse_tree_sf(t):
    try:
        t.label()
    except AttributeError:
        return
    adjp_list = []
    subtree_have_adjp = False

    for subtree in t:
        have_adjp = False
        if subtree.height() != 2:
            have_adjp, adjp_list_subtree = traverse_tree_sf(subtree)

        if (have_adjp):
            subtree_have_adjp = True
            adjp_list.extend(adjp_list_subtree)

    if t.label().find("ADJP") != -1:
        if (subtree_have_adjp):
            return True, adjp_list
        else:
            return True, [" ".join(t.leaves())]
    else:
        return subtree_have_adjp, adjp_list
```

Fig 5: Code snippet for deep extraction

2.2.2.2 Shallow extraction:

The first adjective phrase that was identified would be extracted even if there were ADJP inside it. For example, the sentence in Figure 3, "so tidy and well maintained" would be extracted even though "so tidy" and "well maintained" are ADJP as well.

This was done as in deep extraction, the breaking down of sentences entirely might alter the original intention of the adjective phrase. However, as mentioned above, it might be less useful to identify the frequently used ADJP. This will be evaluated in section 2.4.2.

Code implementation of Shallow extraction:

We would traverse the tree and find ADJP. Once found, it will not continue to find ADJP in the ADJP, but rather continue to find ADJP in the next subtree. This is done recursively, and the implementation is shown in the code snippet below.

```
def traverse_tree_sf_shallow(t):
    try:
        t.label()
    except AttributeError:
        return
    adjp_list = []
    subtree_have_adjp = False
    if t.label().find("ADJP") != -1:
        return True, [" ".join(t.leaves())]
    for subtree in t:
        have_adjp = False
        if subtree.height() != 2:
            have_adjp, adjp_list_subtree = traverse_tree_sf_shallow(subtree)

    if (have_adjp):
        subtree_have_adjp = True
        adjp_list.extend(adjp_list_subtree)

    return subtree_have_adjp, adjp_list
```

Fig 6: Code snippet for shallow extraction

2.2.3 Post-processing of extracted ADJP

Due to the way the ADJP were extracted, there may be instances where punctuations were included. Therefore, the removal of punctuation was done to all the ADJP extracted.

In addition, as defined at the start, a phrase consists of at least 2 words. Thus, adjectives extracted out by StanfordParser would be removed. An example is shown in the diagram.

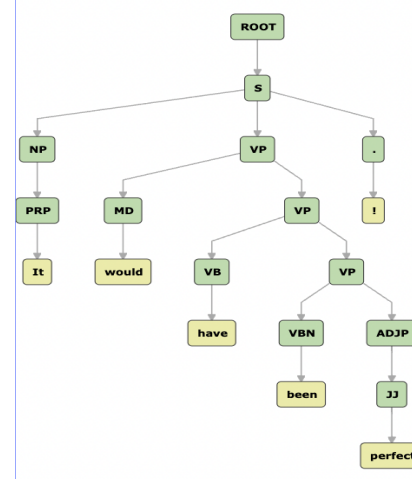


Fig 7: ADJP consisting of only 1 word from Stanford parser

2.2.4 Generate statistics of all ADJP extracted

A table showing the top 5 frequently used ADJP was generated.

Table 6: Top 5 frequently used ADJP for deep and shallow extraction

Deep Extraction		Shallow Extraction	
'much better'	166	'so much'	148
'so much'	156	'so many'	133
'so many'	148	'much better'	107
'not sure'	93	'not sure'	81
'how much'	90	'how much'	78

The full table can be found in appendix 5.4.

2.3 Find the indicative adjectives in a business

Indicative ADJP are those that frequently appear in a particular business review, but not as often in other reviews. To identify indicative ADJP for the specific business, frequently used ADJP would need to be excluded. Thus, frequently used ADJP for all reviews need to be identified.

Frequent is defined as the top 5% of the ADJP in table 6 in terms of frequency.

Formula:

Indicative ADJP of the business = All ADJP found in the business - frequently used ADJP in all reviews

Full list of indicative adjectives can be found in Appendix 5.5.

2.4 Analysis of indicative ADJP extracted

2.4.1 Approach

A random business was chosen and indicative ADJP were extracted. In this case, the indicative ADJP were extracted using Shallow extraction in Section 2.2.2.2. This was to keep the original intention of the adjective phrase as much as possible.

To analyse indicative ADJP:

1. Identify those phrases that are not ADJP - these would be deemed as parsing errors.
2. Identify those phrases that are not a unique characteristic of the business but of other subjects - these would be deemed as identification errors.
3. Make sure that the phrases do not appear more than 2 times in 200 randomly extracted reviews - if the adjective phrase appears more than 2 times in the n (200, 1000, 2000) reviews, it would be deemed as not indicative.

A business with id "VWb8gk_DKUCDKw3Xsdq8Jg" was chosen randomly and analysed.

To perform step 1 above, ADJP were identified manually from the original sentence and compared with the ADJP from the algorithm.

Some of the errors identified are shown in the table below:

Table 7: Parsing errors identified

Sentence	Generated ADJP
For the latte drinkers, their signature latte is amazing, hot, or cold	Amazing, hot or cold
Maybe you shouldn't seat people behind the serving pick up line where they are in danger.	pick up

ADJP that are generated were not supposed to be ADJP. This is usually due to the way the sentence was phrased or punctuations were used.

Similarly, to perform step 2 above, the noun or pronoun of ADJP were identified manually from the original sentence and compared with the ADJP from the algorithm.

Some of the errors identified are shown in the table below:

Table 8: Identification errors identified

Sentence	Generated ADJP
Everyone loved their food especially my 4 year old who ate every bite of his eggs and bacon and deemed this his new favorite restaurant so clearly, we will be back :)	4 year old

As seen, the generated ADJP did not provide any unique characteristic regarding the business.

The extracted indicative ADJP and manually identified indicative ADJP, together with the different types of errors can be found in appendix 5.6.

For step 3, the errors were generated by code and the code snippet is shown below.

```
def get_adjp_appear_more_than_n_times(random_reviews, adjp_list, n):
    adjp_appear_list = []
    for i in adjp_list:
        if len([m.start() for m in re.finditer(i, random_reviews)]) > n:
            adjp_appear_list.append(i)
    print('Number of adjp that appear more than n times: {}'.format(len(adjp_appear_list)))
    return adjp_appear_list
```

Fig 8: Code snippet to get ADJP that appear more than n times in random reviews

The results be further discussed in the next section.

2.4.2 Comparing results of deep and shallow extraction

Full list of indicative ADJP can be found in Appendix 5.7.

Below shows a summary table of the results from the above method.

Table 9: Summary table of extraction results

Count of	Shallow Extraction	Deep Extraction
all Indicative ADJP	152	174
correctly parsed	118	137
partially parsed	7	7
incorrectly parsed	16	18

Partially parsed meant that the ADJP phrase that was extracted was not the complete ADJP phrase.

All indicative ADJP generated by shallow extraction was found in indicative ADJP generated by deep extraction. However, there are ADJP that is generated by deep extraction but not shallow extraction. Here are some the ADJP that were generated **only** by deep extraction and not generated by shallow extraction:

Table 10: ADJP extracted only by deep extraction and not shallow extraction

'just decent',
'really poor',
'thick and tasty',
'mediocre at best',
'friendly and informative',
'so good',
'patient with us',

As seen from the table, it was suspected that 'really poor' and 'so good' was not extracted out as an ADJP from as it was because it is frequently used. The full list can be found in appendix 5.8.

To further compare the results, on how indicative the ADJP, phrases were compared to the n reviews that were randomly selected. Results are shown in the table below.

Table 11: Comparing results between shallow extraction and deep extraction

	Number of ADJP appeared more than 2 times for n randomly extracted reviews		
	n = 200	n = 1000	n = 2000
Shallow extraction	1	7	8
Deep extraction	2	11	13

If ADJP appears more, it is less indicative.

In this case, deep extraction performance is slightly better as there are more correctly identified indicative ADJP in table 9. However, it was also observed that deep extraction tends to have a higher tendency of appearing in other reviews in table 11, which might be generic to all other businesses as well.

Thus, shallow extraction could be used when strictly indicative ADJP should be extracted. Deep extraction could be used when more indicative ADJP should be extracted.

Below shows a summary of the entire workflow for the extraction of indicative ADJP of a particular business.

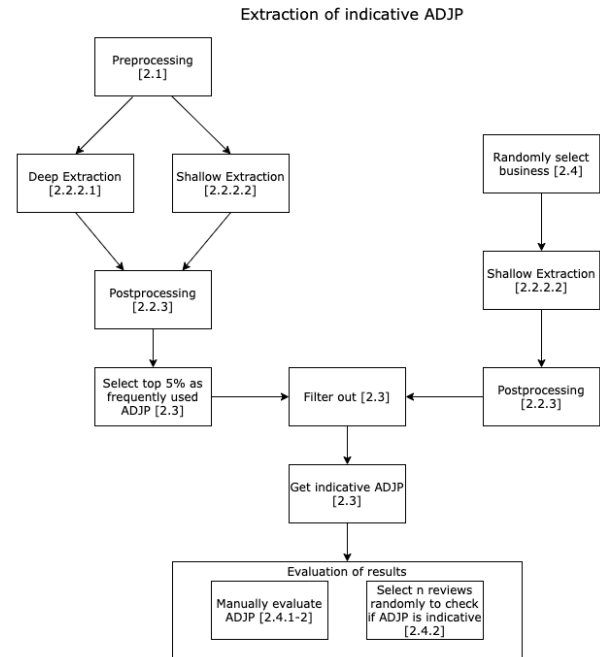


Fig 9: Entire workflow for extraction

2.4.3 Reasons for incorrectly identified ADJP

StanfordParser was not able to parse some of the sentences accurately, thus ADJP could not be extracted accurately.

StanfordParser has a tendency of overlooking negations during the parsing of the adjective parse. For example, in the sentence 'It took way too long to get our drinks, foods, and checks and it wasn't even crowded.', it will parse 'even crowded' instead of 'wasn't even crowded. Thus, almost all the extracted ADJP were affirmative, despite the original data being a negative sentence.

ADJP that were extracted out might not be describing anything related to the business (refer to table 8 for example).

ADJP that were extracted out might have a similar meaning and similar degree to frequently used ADJP in all reviews. For example, "less than acceptable" and "not acceptable" have a similar meaning and a similar degree. However, "less than acceptable" was still considered as an indicative adjective phrase.

2.5 Limitations and room for improvement

Manually extracted ADJP might not be as accurate due to limited linguistic abilities.

As mentioned in 2.4.2, phrases with similar meanings and similar degrees would not be considered when extracting indicative ADJP. Hence, to counter this, semantic clustering could be done such that "less than acceptable" will not be deemed as an indicative adjective phrase

Due to the nature of the Yelp dataset, the reviews might not be in proper English and abbreviations have been used. Thus, during preprocessing, it might be good to convert all these abbreviations into English words. For example 'AWSM' can be converted to 'awesome'.

Optimisation of computation. Currently, due to the traversing of the parsed tree, it took around 2.5 hours to complete 1 run of extraction. This could be improved with optimisation techniques such as multi-threading.

2.6 Conclusion

ADJP were extracted with an accuracy of more than 75% as shown in table 9. They were also considered quite indicative as at most 8% of the extracted indicative ADJP were not indicative as shown in table 10.

However, as mentioned above, indicative ADJP extracted might not be describing the business itself and thus, further filtering of these indicative ADJP needs to be done to get an accurate summary of the particular business.

In addition, some phrases might have similar meaning and degree. Thus, some indicative ADJP were extracted as it is phrased uncommonly and unconventional.

In conclusion, the 2 methods that were implemented here perform relatively well in terms of extraction of indicative adjective phrases.

3 Application

3.1 Description

Sentiment Analysis is the process of using NLP to classify text into three categories, positive, negative, and neutral. It is widely being used in business industries to study customer responses towards specific aspects. With the advancement of deep learning and the abilities to analyze text, sentiment analysis has become a very important tool for understanding human reactions given a product [20].

The analysis carried out will attempt to predict the sentiment of a review randomly chosen from the Yelp dataset. Sentiments are classified into two classes, 'positive' and 'negative'.

3.2 Methodology

The analysis is implemented on the entire dataset containing 15,300 reviews. Given the size of the dataset, it can be predicted that the neural network model generated will be overfitting due to the lack of reviews. Parameters are added to the model in order to mitigate the issue of overfitting.

3.2.1 Resources Used

Table 12: Resources used

NLTK Packages	Stemmer Lemmatizer Tokenizer Train_test_split
Data Extraction	Json Pandas [21] Numpy [22]
Tensorflow/Keras [23]	Padding Layers Model Utils
Result visualisation	Matlab [24]

3.2.2 Data Cleaning

Firstly, the 'text' field from the JSON file is extracted and stored in a Pandas Dataframe. The dataframe now contains a list of reviews of all businesses in the dataset. The following pre-processing methods were implemented on the raw set of text.

1. Remove punctuation
2. Remove HTML syntax
3. Remove URLs
This is to ensure that links posted in reviews are not taken into consideration for the analysis.
4. Remove Emojis
This is to remove any non-text form of expressions to mitigate causes of errors.
5. Lemmatize the text
Similar to stemming, the text is lemmatized maintaining the original context of the word before converting them to their meaningful base form.

In order to label all the reviews with a sentiment value, the NLTK Sentiment Vader package [25] was used to find a compound rating for the text based on the probability of the text being 'positive', 'neutral', or 'negative'.

The total sum of the scores for each word is adjusted according to the rules and then normalised to values between -1 and 1. If the overall compound rating is a negative value, the review is labelled as 'neg' and 'pos' if the value is positive

```
nltk_sent = SentimentIntensityAnalyzer()

rev_df_clean['ratings'] =
rev_df_clean["Reviews"].apply(lambda x:
nltk_sent.polarity_scores(x))
```

Fig 10. Code to apply polarity on text

3.2.3 Model Creation and Training

The target values which are the sentiment labels are factorized to categorical values using the factorize() method. This returns an array of numerical values along with the index of the respective labels.

```
sentiment_label = upd_df["comp_rating"].factorize()
sentiment_label

(array([0, 0, 0, ..., 0, 0, 0], dtype=int64),
Index(['pos', 'neg'], dtype='object'))
```

Fig 11. Categorical Labelling of sentiment tags

The reviews are tokenized and converted into an array of vector embeddings. The reviews in the dataset range from approximately 10 words to 800 words and hence padding is required to ensure that all sentences are of the same length.

A sequential model is implemented containing the following layers:

1. Embedding layer
2. LSTM (Long short-term memory) layer to handle sequential information
3. SpatialDropout1D layer to reduce overfitting
4. Dropout layer to reduce overfitting
5. Output layer with sigmoid activation.

The model was compiled using the 'adam' optimiser and based on the 'binary cross entropy' loss function (Appendix Fig 5.9.1). Early Stopping was implemented based on the minimum validation loss. Early stopping ensures that the model stops learning at a point to reduce generalisation error. The model performance was analysed by plotting a graph of training data loss against test data loss as shown in Fig. 12.

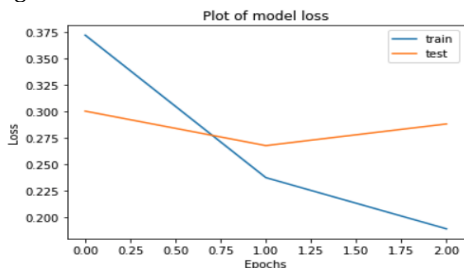


Fig 12. Plot of training loss and validation loss

As predicted, from the graph shown above the trained model shows instances of overfitting as the test loss begins to increase after convergence.

3.3 Discussion on findings

Table 13: Model predictions vs Actual Label values

Text	Model Prediction	Actual Label
...The condition was really good. The support is comfortable for the back...	pos	pos
This place is beyond horrible If I could give them a zero ...	pos	neg
If I could give 0 stars I would do not eat here ...	pos	pos
Great overall atmosphere ...	pos	neg

From the table above, it can be seen that the negative review in row 2 was predicted to be "pos" resulting in an invalid prediction. Similarly, the text in rows 3 and 4 shows evidence of the situation where VADER has identified the review incorrectly resulting in a decrease in the accuracy of predictions (even when our model predicted row 4 correctly).

The accuracy of the Sentiment Analysis Model created cannot be guaranteed due to 2 main limitations. The training dataset has a high level of bias towards positive reviews and the limited accuracy of the SentimentIntensityAnalyser tool from NLTK.

Positive reviews are classified with higher confidence as compared to negative reviews. As mentioned, the main reason for this issue is the imbalance in the 2 classes.

The SentimentIntensityAnalyser tool used to predefine the training data has inaccuracies. Reviews which should be defined as negative (based on contextual knowledge) were defined as positive (Appendix 5.9.2). This would affect the accuracy of the model in a few ways. Reviews predicted to be

correct by contextual knowledge would still be analysed as erroneous when compared with the pre-defined label. Alternatively, there would be reviews which are predicted to be consistent with the erroneous pre-defined labels.

One solution to mitigate the issue of bias will be to over sample the minority class during the preprocessing stage. It is also possible to implement reweighing or optimising algorithms produced by IBM's open source package known as 'AIF360' [26]. Furthermore, implemented rules to obtain sentiment labelling can be changed to using 'Star' ratings to differentiate between "Positive" and "Negative" reviews. This can reduce the probability of errors in sentiment labelling due to the limitation mentioned above.

4 References

- [1] T. Perry, 1 February 2021. [Online]. Available: <https://www.machinelearningplus.com/nlp/what-is-tokenization-in-natural-language-processing/>. [Accessed 15 October 2021].
- [2] S. E. L. a. E. K. Bird, Natural Language Processing with Python., O'Reilly Media Inc., 2009.
- [3] T. Contributor, "stemming," January 2018. [Online]. Available: <https://searchenterpriseai.techtarget.com/definition/stemming>. [Accessed 15 October 2021].
- [4] K. Pykes, 26 November 2020. [Online]. Available: <https://towardsdatascience.com/part-of-speech-tagging-for-beginners-3a0754b2ebba>. [Accessed 16 October 2021].
- [5] M. & M. I. Honnibal, spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing., 2017.
- [6] A. Malhotra, "Introduction to Libraries of NLP in Python — NLTK vs. spaCy," 24 June 2018. [Online]. Available: <https://medium.com/@akankshamalhotra24/introduction-to-libraries-of-nlp-in-python-nltk-vs-spacy-42d7b2f128f2>. [Accessed 13 October 2021].
- [7] K. Alizadeh, "Word Vectors and Word Meanings," [Online]. Available: <https://towardsdatascience.com/word-vectors-and-word-meaning-90493d13af76>. [Accessed 10 October 2021].
- [8] S. Cengiz, "Read from a word file in python," April 2020. [Online]. Available: <https://stackoverflow.com/questions/61147974/read-from-a-word-file-in-python>. [Accessed 10 October 2021].
- [9] Pradhan29, "How can I extract the text part of a blog page while web scraping using BeautifulSoup4?," 2017. [Online]. Available: <https://stackoverflow.com/questions/49205608/how-can-i-extract-the-text-part-of-a-blog-page-while-web-scraping-using-beautiful>. [Accessed 12 October 2021].

- [10] K. Yeo, "Apple iPad Mini (2021) review: A powerful & fantastic little tablet," 2 October 2021. [Online]. Available: <https://www.hardwarezone.com.sg/review-apple-ipad-mini-6th-generation-2021-review-singapore-price-specs>. [Accessed 12 October 2021].
- [11] "Torino a tougher task than Chelsea, Allegri warns Juventus," 1 October 2021. [Online]. Available: <https://www.channelnewsasia.com/sport/torino-tougher-task-chelsea-allegri-warns-juventus-2215766>. [Accessed 12 October 2021].
- [12] L. Richardson, *Beautiful soup documentation*, April 2007.
- [13] G. Pipis, "LanguageTool: Grammar And Spell Checker In Python," 14 September 2020. [Online]. Available: <https://predictivehacks.com/languagetool-grammar-and-spell-checker-in-python/>. [Accessed 12 October 2021].
- [14] "python-docx2txt," [Online]. Available: <https://github.com/ankushshah89/python-docx2txt>. [Accessed 10 10 2021].
- [15] DeepSmoseli, "Python nltk.tokenize.sent_tokenize() Examples," [Online]. Available: https://www.programcreek.com/python/example/73734/nltk.tokenize.sent_tokenize. [Accessed 12 October 2021].
- [16] "Cambridge Dictionary," Cambridge University Press 2021, [Online]. Available: <https://dictionary.cambridge.org/grammar/british-grammar/adjective-phrases>. [Accessed 24 October 2021].
- [17] [Online]. Available: <https://github.com/avian2/unidecode>. [Accessed 15 10 2021].
- [18] contractions. [Online]. Available: <https://github.com/kootenpv/contractions>. [Accessed 15 10 2021].
- [19] M. S. J. B. J. R. F. S. B. a. D. M. C. D. Manning, "The Stanford CoreNLP Natural Language Processing Toolkit," in *ACL (System Demonstrations)*, 2014, p. 55–60.
- [20] "Sentiment Analysis: A Definitive Guide," [Online]. Available: <https://monkeylearn.com/sentiment-analysis/>. [Accessed 20 October 2021].
- [21] W. a. o. McKinney, "Data structures for statistical computing in python," vol. 455, pp. 51-56, 2010.
- [22] C. M. K. v. d. W. S. e. a. Harris, "Array programming with NumPy," *Nature*, vol. 585, pp. 357--362, 2020.
- [23] A. A. P. B. E. B. Z. C. C. G. S. C. A. D. J. D. M. D. S. G. I. G. A. H. G. I. M. I. R. J. Y. J. L. artín Abadi, "TensorFlow: Large-scale machine learning on heterogeneous systems," [Online]. Available: <https://www.tensorflow.org/>.
- [24] MATLAB, MATLAB:2010 version 7.10.0 (R2010a), The MathWorks Inc., 2010.
- [25] C. & G. E. Hutto, "VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14).," 2014. [Online]. Available: <https://www.nltk.org/api/nltk.sentiment.vader.html>. [Accessed 21 October 2021].
- [26] IBM, "AIF360," [Online]. Available: <https://github.com/Trusted-AI/AIF360>. [Accessed 21 October 2021].
- [27] E. K. a. E. L. Steven Bird, "Categorizing and Tagging Words," 2019. [Online]. Available: <https://www.nltk.org/book/ch05.html>. [Accessed 21 October 2021].

5 Appendix

5.1 POS Tagging

Table 5.1.1: 5 randomly selected sentences

	sentence
0	No judgement here, as I like to lick my plates clean too
1	It was a blast.
2	The night manager started shouting at me and even told me that my room was not his issue.
3	calls were extra charge.
4	We dropped in during show time and since the box office was closed, we decided to sneak into the main show.

Table 5.1.2.1: POS tagging results for the first randomly selected sentence

----- Sentence 1 -----

	word_x	POS_Tag_NLTK	POS_Tag_spaCy
0	No	DT	DET
1	judgement	NN	NOUN
2	here	RB	ADV
3	,	,	PUNCT
4	as	IN	ADP
5	I	PRP	PRON
6	like	VBP	VERB
7	to	TO	PART
8	lick	VB	VERB
9	my	PRP\$	PRON
10	plates	NNS	NOUN
11	clean	VBP	ADJ
12	too	RB	ADV
13	!	.	PUNCT

Table 5.1.2.2: POS tagging results for the second randomly selected sentence

----- Sentence 2 -----

	word_x	POS_Tag_NLTK	POS_Tag_spaCy
0	It	PRP	PRON
1	was	VBD	AUX
2	a	DT	DET
3	blast	NN	NOUN
4	.	.	PUNCT

Table 5.1.2.3: POS tagging results for the third randomly selected sentence

----- sentence
----- Sentence 3 -----

	word_x	POS_Tag_NLTK	POS_Tag_spaCy
0	The	DT	DET
1	night	NN	NOUN
2	manager	NN	NOUN
3	started	VBD	VERB
4	shouting	VBG	VERB
5	at	IN	ADP
6	me	PRP	PRON
7	and	CC	CCONJ
8	even	RB	ADV
9	told	VBD	VERB
10	me	PRP	PRON
11	that	IN	SCONJ
12	my	PRP\$	PRON
13	room	NN	NOUN
14	was	VBD	AUX
15	not	RB	PART
16	his	PRP\$	PRON
17	issue	NN	NOUN
18	.	.	PUNCT

Table 5.1.2.4: POS tagging results for the fourth randomly selected sentence

----- Sentence 4 -----

	word_x	POS_Tag_NLTK	POS_Tag_spaCy
0	calls	NNS	NOUN
1	were	VBD	AUX
2	extra	JJ	ADJ
3	charge	NN	NOUN
4	.	.	PUNCT

Table 5.1.2.5: POS tagging results for the fifth randomly selected sentence

sentence
----- Sentence 5 -----

	word_x	POS_Tag_NLTK	POS_Tag_spacy
0	We	PRP	PRON
1	dropped	VBD	VERB
2	in	IN	ADP
3	during	IN	ADP
4	show	NN	NOUN
5	time	NN	NOUN
6	and	CC	CCONJ
7	since	IN	SCONJ
8	the	DT	DET
9	box	NN	PROPN
10	office	NN	NOUN
11	was	VBD	VERB
12	closed	VCN	ADJ
13	,	,	PUNCT
14	we	PRP	PRON
15	decided	VBD	VERB
16	to	TO	PART
17	sneak	VB	VERB
18	into	IN	ADP
19	the	DT	DET
20	main	JJ	ADJ
21	show	NN	NOUN
22	.	.	PUNCT

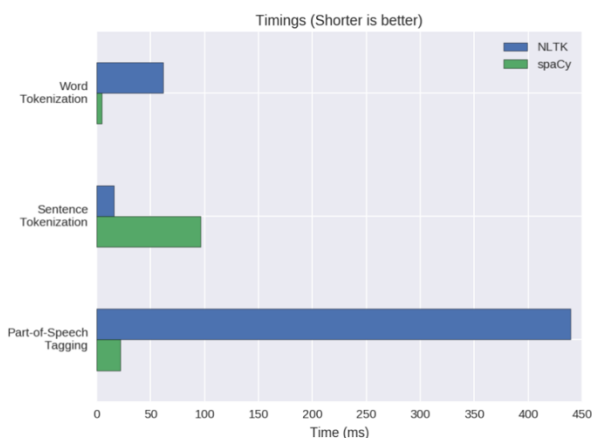


Figure 5.1.3: Comparison between NLTK and spaCy

5.2 Writing Style Analysis

5.2.1 Criteria Implementation

```
def Avg_SentLengthByWord(text):
    tokens = sent_tokenize(text)
    avg_sent = np.average([len(token.split()) for token
in tokens])
    return avg_sent
```

Fig 5.2.1.1: Source code of Average Sentence Length

```
def Avg_wordLength(str):
    str.translate(string.punctuation)
    tokens = word_tokenize(str, language='english')
    st = [",", ".", "!", ":", ";", "#", "$", "%", "&", "(", ")", "*",
"+", "-", ":", "/", ":", ":", "<", "=", ">", "?",
"@", "[", "\\", "]", "^", "_", ":", "{", "|", "}", "~", '\t',
'\n']
    stop = stopwords.words('english') + st
    words = [word for word in tokens if word not in
stop]
    avg_word = np.average([len(word) for word in
words])
    return avg_word
```

Fig 5.2.1.2: Source code of Average Word Length

```
def grammar_check(text):
    matches = tool.check(text)
    error_count = len(matches)
    #print(matches)
    return error_count
```

Fig 5.2.1.3: Source code of Grammar Check

```
def cap_check(text):
    error_cap = 0
    for sentence in sent_tokenize(text):
        if not sentence[0].isupper(): # sentence does not
start with a capital letter
            error_cap+=1
    return error_cap
```

Fig 5.2.1.4: Source code of Capitalisation Check

5.3 Top 10 most-frequent noun-adjective pairs

5.3.1 20 random reviews with rating 2

Noun	Adjective	count
service	slow	2
meat sandwich	loose	2
part	best	2
restaurant	clean	2
dish	deep	2
food	vietnamese	1
curry soup basis	very flavourful	1
beef	impossible	1
time	absolute worse	1
soup curry	good	1

5.3.2 20 random reviews with rating 3

Noun	Adjective	count
service	good	2
service	fast	2
restaurant	other	2
trio	awesome	2
food	average	2
pancake	good	1
breakfast	fast	1
breakfast	speedy	1
guy	great	1

5.3.3 20 random reviews with rating 4

Noun	Adjective	count
food	good	3
place	great	2
fudge	hot	2
place	good	2

strip mall	small	1
parking lot	small	1
monkey waffle	funky	1
waffle	terrific	1
edge	crisp	1
vehicle	excellent	1

5.3.4 20 random reviews with rating 5

Noun	Adjective	count
food	great	2
staff	friendly	2
stuff	other	2
good	own	2
occasion	multiple	1
experience	good	1
vibe	great	1
place	favorite	1
service	always clean fast	1
food	delicious	1

5.4 Frequency of ADJP

5.4.1 Deep extraction

The excel sheet can be found in 'results/extraction result' folder with the name `frequency_result_deep_extraction.csv`

5.4.2 Shallow extraction

The excel sheet can be found in 'results/extraction result' folder with the name

`frequency_result_shallow_extraction.csv`

5.5 Frequently-used ADJP

5.5.1 Deep extraction

The excel sheet can be found in 'results/extraction result' folder with the name

`frequently_used_adjp_deep_extraction.csv`

5.5.2 Shallow extraction

The excel sheet can be found in 'results/extraction result' folder with the name

`frequently_used_adjp_shallow_extraction.csv`

5.6 Extracted indicative ADJP and manually identified indicative ADJP

The excel sheet can be found in 'results/extraction result' folder with the name `business_indicative_adjp_analysis.xls`

5.7 Indicative ADJP extracted

5.7.1 Deep extraction

The excel sheet can be found in 'results/extraction result' folder with the name 'indicative_adjp_deep_extraction.csv'

5.7.2 Shallow extraction

The excel sheet can be found in 'results/extraction result' folder with the name 'indicative_adjp_shallow_extraction.csv'

5.8 Indicative ADJP extracted by deep extraction and not extracted by shallow extraction (difference)

Indicative Adjective phrases
'pick up',
'just decent',
'able to order',
'really poor',
'so watery',
'way too long',
'quite satisfying',
'a little runny in the middle',
'how expensive',
'thick and tasty',
'too popular',
'any more',
'mediocre at best',
'hard to decide what to get',
'hard to decide',
'thoroughly enjoyable',
'friendly and informative',
'so good',
'patient with us',
'slightly high',
'very helpful and accommodating',
'very well seasoned'

5.9 Application

5.9.1 Model implementation snippet

```
model = Sequential()
model.add(Embedding(len(tokenizer.word_index)+1,
                    EMBEDDING_SIZE,
                    input_length=SEQUENCE_LENGTH))
model.add(SpatialDropout1D(0.25))
model.add(LSTM(128, dropout=0.5,
               recurrent_dropout=0.2))
model.add(Dropout(0.3))
model.add(Dense(1, activation="sigmoid"))
model.compile(optimizer="adam",
              loss='binary_crossentropy',
              metrics=["accuracy"])
model.summary()
```

5.9.2 Results

The excel sheet of sampled data prediction accuracy can be found in 'results/application result' folder with the name 'model-accuracy.xlsx'

6 Extension

6.1 Pos Tagging

Alternatively, POS tagging could also be conducted using different Ngram tagging methods, namely Unigram Tagger and Bigram Tagger. Upon choosing 5 random sentences, the remaining sentences were used as training data for the respective taggers. The results along with the observations obtained after implementing POS tagging on the random sentences can be found in *table 6.1*.

Unigram Tagger: The POS tag for a given token is determined using a single word as the context [27].

Bigram Tagger: The POS tag is determined using two items as the context. The previously tagged word is the first item, while the currently marked word is the second [27].

General Observation: The Bigram model yielded more unknown (POS Tag: None) POS tags than the Unigram model. The implemented classification pipeline does not construct a vector for unseen ngrams, hence lowering the amount of available information for the classifier during training and inference. This would mean that the word was not seen during training with a 'None' tag on the previous word. Consequently, the Bigram tagger failed to tag the remaining texts in the sentence. This thus accounted for the higher number of undetermined tags.

Accuracy obtained:

accuracy of unigram: 0.9696969696969697

accuracy of bigram : 0.5

Unigram tagger had an approximate 0.4 higher accuracy than Bigram tagger. The Bigram tagger was unable to learn the context based on the singular previously tagged word. Whereas, the Unigram ignores the preceding context and estimates the most common tag for each word, resulting in the higher accuracy observed.

To increase the accuracy of the bigram tagger would be to combine taggers instead. Combining taggers would fall back

to the next chained tagger when one tagger does not cover a token. For example, if the bigram tagger is unable to find a tag for the token, a unigram would be executed next. Followed by a default tagger if the unigram tagger is also unable to find a tag.

A combining tagger was executed by combining bigram, unigram and default tagger together. Bigram tagger would first be executed. If no tags were found, it would fall back on the next callback which would be the unigram. Subsequently, if the unigram could not find any tags, it would fallback on the default tagger which would be 'NN', the most common tagger found from the training data. This yielded the following result with the accuracy being higher than that of a unigram. The results of the POS tagging can be found in *table 6.2*.

Accuracy obtained:

accuracy of unigram : 0.9696969696969697

accuracy of bigram : 0.5

accuracy of combining: 1.0

Table 6.1.1: POS tagging results for the first randomly selected sentence

----- Sentence 1 -----			
	word_x	POS_Tag_Unigram	POS_Tag_Bigram
0	No	DT	DT
1	judgement	NN	None
2	here	RB	None
3	,	,	None
4	as	IN	None
5	I	PRP	None
6	like	IN	None
7	to	TO	None
8	lick	VB	None
9	my	PRP\$	None
10	plates	NNS	None
11	clean	JJ	None
12	too	RB	None
13	!	.	None

Table 6.1.2: POS tagging results for the second randomly selected

----- Sentence 2 -----			
	word_x	POS_Tag_Unigram	POS_Tag_Bigram
0	It	PRP	PRP
1	was	VBD	VBD
2	a	DT	DT
3	blast	NN	NN
4	.	.	.

Table 6.1.3: POS tagging results for the third randomly selected sentence

----- Sentence 3 -----			
	word_x	POS_Tag_Unigram	POS_Tag_Bigram
0	The	DT	DT
1	night	NN	NN
2	manager	NN	NN
3	started	VBD	VBD
4	shouting	VBG	None
5	at	IN	None
6	me	PRP	None
7	and	CC	None
8	even	RB	None
9	told	VBD	None
10	me	PRP	None
11	that	IN	None
12	my	PRP\$	None
13	room	NN	None
14	was	VBD	None
15	not	RB	None
16	his	PRP\$	None
17	issue	NN	None
18	.	.	None

Table 6.1.4: POS tagging results for the fourth randomly selected

sentence			
----- Sentence 4 -----			
	word_x	POS_Tag_Unigram	POS_Tag_Bigram
0	calls	NNS	None
1	were	VBD	None
2	extra	JJ	None
3	charge	NN	None
4	.	.	None

Table 6.1.5: POS tagging results for the fifth randomly selected sentence

----- Sentence 5 -----			
	word_x	POS_Tag_Unigram	POS_Tag_Bigram
0	We	PRP	PRP
1	dropped	VBD	VBD
2	in	IN	IN
3	during	IN	IN
4	show	NN	NN
5	time	NN	NN
6	and	CC	CC
7	since	IN	IN
8	the	DT	DT
9	box	NN	NN
10	office	NN	NN
11	was	VBD	VBD
12	closed	VCN	VCN
13	,	,	,
14	we	PRP	PRP
15	decided	VBD	VBD
16	to	TO	TO
17	sneak	VB	VB
18	into	IN	IN
19	the	DT	DT
20	main	JJ	JJ
21	show	NN	NN
22	.	.	.

Table 6.2.1: POS tagging results for the first randomly selected

sentence			
----- Sentence 1 -----			
	word_x	POS_Tag_Unigram	POS_Tag_Combine
0	No	DT	DT
1	judgement	NN	NN
2	here	RB	RB
3	,	,	,
4	as	IN	IN
5	I	PRP	PRP
6	like	IN	VBP
7	to	TO	TO
8	lick	VB	VB
9	my	PRP\$	PRP\$
10	plates	NNS	NNS
11	clean	JJ	VBP
12	too	RB	RB
13	!	.	.

Table 6.2.2: POS tagging results for the second randomly selected sentence

----- Sentence 2 -----			
	word_x	POS_Tag_Unigram	POS_Tag_Combine
0	It	PRP	PRP
1	was	VBD	VBD
2	a	DT	DT
3	blast	NN	NN
4	.	.	.

Table 6.2.3: POS tagging results for the third randomly selected

sentence			
----- Sentence 3 -----			
word_x	POS_Tag_Unigram	POS_Tag_Combine	
0	The	DT	DT
1	night	NN	NN
2	manager	NN	NN
3	started	VBD	VBD
4	shouting	VBG	VBG
5	at	IN	IN
6	me	PRP	PRP
7	and	CC	CC
8	even	RB	RB
9	told	VBD	VBD
10	me	PRP	PRP
11	that	IN	IN
12	my	PRP\$	PRP\$
13	room	NN	NN
14	was	VBD	VBD
15	not	RB	RB
16	his	PRP\$	PRP\$
17	issue	NN	NN
18	.	.	.

Table 6.2.4: POS tagging results for the fourth randomly selected sentence

----- Sentence 4 -----			
word_x	POS_Tag_Unigram	POS_Tag_Combine	
0	calls	NNS	NNS
1	were	VBD	VBD
2	extra	JJ	JJ
3	charge	NN	NN
4	.	.	.

Table 6.2.5: POS tagging results for the fifth randomly selected

sentence			
----- Sentence 5 -----			
word_x	POS_Tag_Unigram	POS_Tag_Combine	
0	We	PRP	PRP
1	dropped	VBD	VBD
2	in	IN	IN
3	during	IN	IN
4	show	NN	NN
5	time	NN	NN
6	and	CC	CC
7	since	IN	IN
8	the	DT	DT
9	box	NN	NN
10	office	NN	NN
11	was	VBD	VBD
12	closed	VRN	VRN
13	,	,	,
14	we	PRP	PRP
15	decided	VBD	VBD
16	to	TO	TO
17	sneak	VB	VB
18	into	IN	IN
19	the	DT	DT
20	main	JJ	JJ
21	show	NN	NN
22	.	.	.