CZ4045 Assignment Report

Chen Chongsong
Huangfu Qingchuan
Li Huimin
Liu Fangbing
Zhang Bin
{chen1129,c160135,c160029,liuf0029,bzhang015}@e.ntu.edu.sg
SCSE, Nanyang Technological University
Group Number: G19C

1 INTRODUCTION

This LaTeX report serves as a technical explanation of our group's CZ4045 project. We analyze the given Yelp dataset [2] using two Python packages, namely NLTK and SpaCy, and finally obtain reasonable results. The details of our analysis are illustrated in Section 3, 4, and 5.

Each of our 5 team members contributes to the project and the report dedicatedly:

Chen Chongsong: Section 3.3, 4, 5
Huangfu Qingchuan: Section 3.2
Li Huimin: Section 3.4
Liu Fangbing: Section 3.1
Zhang Bin: Section 3.5

2 REVIEW SAMPLING

We randomly sample 5 reviews from the dataset, and use them frequently for tasks described in Section 3 to 5. It's obvious that Review 2, 4, and 5 are relatively longer.

Review 1:

> I love love their Kalbi, I always order it the sauce is what makes it really good..hmm think I wanna eat that today.i didn't like their noodles to sweet..

Review 2:

> We found JBs though all the hype on instagram and was excited to try the international burgers and Poutine. I got the Burgetta burger and the JB House poutine. The Poutine was tasty with all its toppings and the burger patty was good, but there was barely any toppings and much smaller than anticipated. I reread the description of the burgetta burger and realized that I didn't even get the fried eggplant! I felt very mislead by the advertising and disappointed. I would give this company an A for creativity but I will not be coming back for an undersized meal with an oversized pricetag.

Review 3:

> Good sushi, fast service! ... Not the best i tasted, but surely works if you crave for sushi, and thats the only place nearby, you can go to kill the Sushi craving.

Review 4:

> This location was a surprise to me, even though I have been here a million times. The location is also incredibly busy! I was surprised how busy it was. Nevertheless the wait time wasn't anything crazy. We were able to find a table for 2 within a reasonable amount of time. We ordered a few items, from each of the different food groups: pork, beef, chicken, you get the idea. With all the food groups covered, we had a really enjoyable time. The food was good, and the service was speedy. The only drawback might have been the noise level, but there's not much you can do about that with a full restaurant on a very busy day.

Review 5:

> I bought a wine and bar fridge in August, I was told that the wine fridge had two temps, it only had one. Delivery was a joke, only had one of the items and wouldn't switch the door handles, spent more time taking off work waiting for delivery. Then wine fridge hasn't worked since, Spencers said it was out of their hands, the manufacturer sent an appliance person to fix it, still hasn't worked. Have called Spencers numerous times and gone in person to no avail. Months later, I still can't use it unless I like frozen wine. (bottles have been ruined) Gary, the sales person who said he'd check with the manager and he'd call me next day...over two weeks later STILL waiting. But he did ask me if I knew how to turn the temp knob. Seriously. Do not deal with these people. They get their money and don't care and are insulting to women. I have filed with the BBB.

3 PART I: DATASET ANALYSIS TASKS

3.1 Writing Style

All the texts are reviews uploaded by customers. As they deliver their evaluation and points of view on restaurants, they may not consider about grammars and spelling seriously before posting. As a result, errors of reviews commonly exist in the dataset. In this part, I select two sentences as examples and analyze their writing styles. Generally, three aspects will be discussed, which are spelling, grammar and punctuation.

Review 1 of section 2: There is no spelling error. However, the review contains non-standard words, for instance, "wanna" which is spoken form and "hmm" which is a modal particle.

There are some problems with grammar as well. For example, in the first sentence, there is two consecutive verbs 'love'. Moreover, the first word of a sentence is not capitalized sometimes, like the last sentence, 'i didn't like their noodles to sweet..'. Punctuation is also not proper, there often double dots at the end of a sentence, such as the last sentence.

Review 2 of section 2: Firstly, there are spelling errors, such as in the first sentence, "We found JBs though all the hype on Instagram". Here, "though" should be "through". In addition, many words are not capitalized properly, such as "Burgetta" and "Poutine". There are several cases, where the first characters of these words are capitalized when the words are in the middle of a sentence.

There are also some problems with grammar. Some words do not follow standard inflectional morphology. For example, in Review 2, the reviewer writes "I felt very mislead by". This sentence is in passive voice so the verb "mislead" should be in passive form "misled". Also, some sentences of this review miss pronouns, such as "I would give this company an A for creativity". This sentence lacks a "their"/ "the" before "creativity". As for punctuation, a comma is missed before "but" in the last sentence.

3.2 Sentence Segmentation

Segmentation method: We have used spaCy to segment all the reviews into sentences. According to spaCy documentation, the sentence boundary is determined based on Dependency Parse. We applied our custom function:

sentence_text(text, nlp)

to obtain the segmented sentences output. "*en_core_web_sm*" denotes the core English base corpus parameter of spaCy. doc.sents returns the sentences of the processed nlp document of an input text, given "*en_core_web_sm*" parameter.

```
nlp = spacy.load("en_core_web_sm")
def sentence_text(text, nlp):
    doc = nlp(text)
    sentences = [sent.string.strip() for sent in doc.
     sents]
    return sentences
```

Test model behavior: We have performed sentence segmentation on the 5 reviews sampled in section 2, to gain insights of how Spacy's sentence segmentation method behaves.

For **Review 1**, the program output is as follows:

```
1: [
'I love love their Kalbi, I always order it',
'the sauce is what makes it really good..',
"hmm think I wanna eat that today.i didn't like their
    noodles to sweet.."
]
```

We can see that it's a correct result based on the semantics of the sentence. In the original text, even when the first 2 sentences has no separators in between, the model is able to identify that the latter should be a standalone sentence. This matches our expectation that the model performs Dependency Parse.

Please refer to Appendix for all the sentence segmentation results of our 5 sample texts.

Figure 1: Distribution of Sentence Counts: 1 star

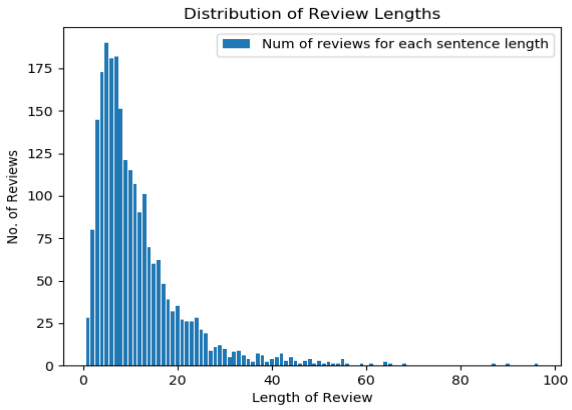


Figure 2: Distribution of Sentence Counts: 2 star

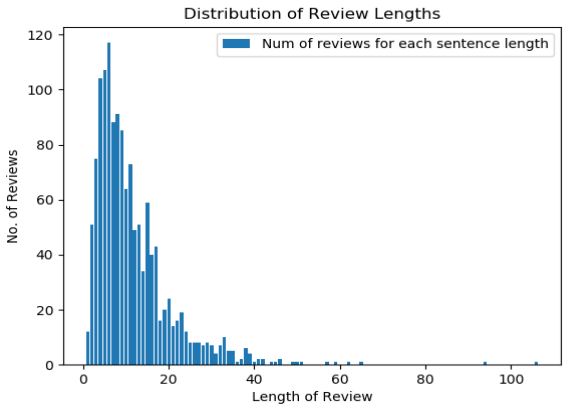


Figure 3: Distribution of Sentence Counts: 3 star

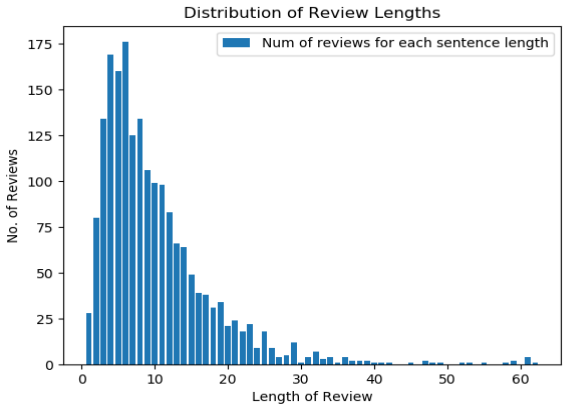


Figure 4: Distribution of Sentence Counts: 4 star

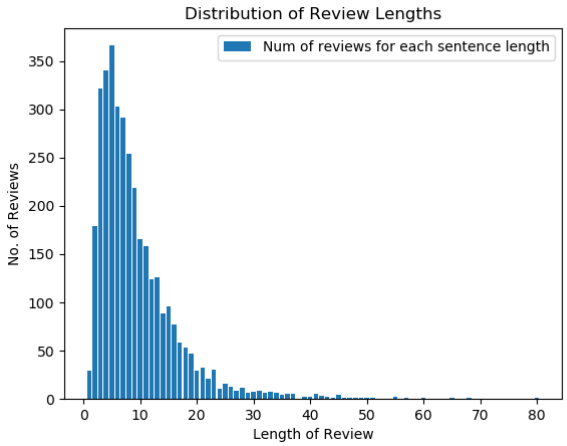
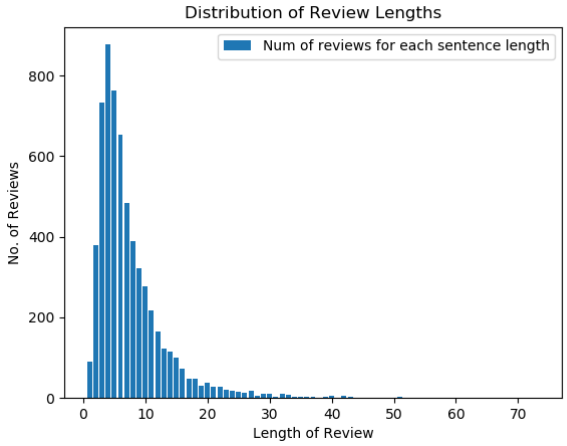


Figure 5: Distribution of Sentence Counts: 5 star



Distribution Plotting: Reviews with the same rating are grouped together. Here are the 5 results of the occurrence of each sentence count of a review of a certain rating (Figure 1, 2, 3, 4, 5). From those images, we found out that with different rating stars, the distributions of the sentence count are similar. They all follow an exponential decay pattern. Generally speaking, the decay rates of extreme ratings (1 and 5) are higher than moderate ratings (3).

3.3 Tokenization and Stemming

We use NLTK to tokenize all the reviews (with/without stemming), and for stemming, we use the Porter Stemmer [1] implemented by NLTK. Thereafter, numbers of words (tokens) are counted for each review.

Frequent Tokens: The top 20 frequent tokens (before and after stemming) are listed in Table 1. Please refer to the appendix section for a full list of 199 stop words adopted in this task. By analyzing the results in Table 1, we have the following findings.

- The implementation of stemmer is generally correct. For example, *very* → *veri*, *service* → *servic*, *ordered* → *order*.
- Most words (e.g. *food, good, service, order*) listed in the results are as expected, since the dataset mainly targets at food and restaurants.
- We also notice several weird stems in the table. For example, *"was"* are stemmed into *"wa"*, hence not being recognized as stop words

Table 1: Top 20 frequent tokens from review

No Stemming		With Stemming	
Token	*Frequency*	*Token*	*Frequency*
food	8580	wa	31504
place	8236	thi	13927
good	7919	place	9408
great	6295	food	8731
service	6027	good	8055
like	5534	veri	6825
get	5216	time	6588
time	5172	get	6390
back	4717	servic	6338
go	4145	great	6302
really	3731	like	6231
$	3706	order	6150
got	3171	go	5993
order	2820	back	4744
nice	2756	tri	3978
ordered	2620	come	3784
came	2614	realli	3731
best	2605	$	3706
always	2587	love	3291
chicken	2515	onli	3181

Distribution Plotting: The data distributions are shown in Figure 6 and Figure 7 respectively. For both cases, length 30 has a maximum number of reviews, and the max review lengths are 1194. In theory, the two distributions should be the same. In actual experiments, however, the numbers are slightly different for a small portion of reviews, due to the implementation of stemmer.

Figure 6: Distribution of Token Counts (no stemming)

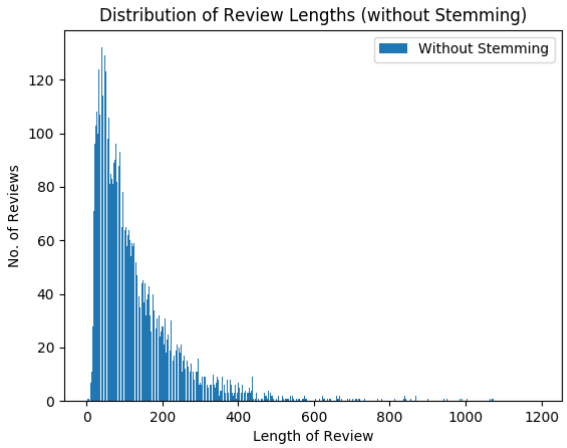
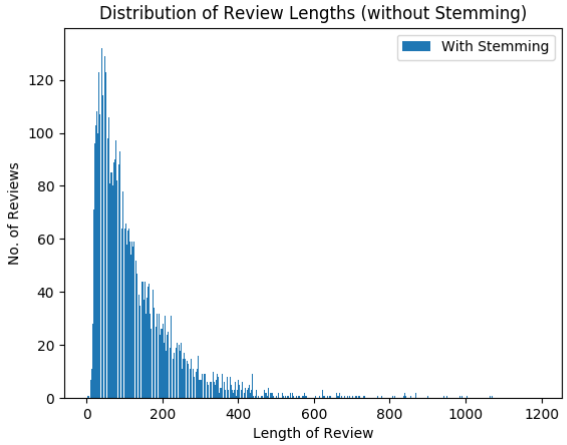


Figure 7: Distribution of Toke Counts (with stemming)



3.4 POS Tagging

Following tokenization, part-of-speech tagging is the next step in the typical NLP pipeline. In this process, we firstly extracted the tokens from segmented sentences using *nltk.tokenize.word_tokenize()* method and filtered out punctuations (as the tokenizer treats most punctuation characters as separate tokens). Then we used a POS-tagger provided by NLTK, which processes a sequence of input words and attaches a part-of-speech tag to each word.

Here we randomly selected one sentence from each review to apply POS tagging. The sample sentences, tagging results and detailed discussion are shown below:

```
# Sample sentence from review 1
>>> text = word_tokenize("I love love their Kalbi, I
    always order it the sauce is what makes it really
    good.")
>>> nltk.pos_tag(text)
[('I', 'PRP'), ('love', 'VBP'), ('love', 'VB'), ('their',
     'PRP$'), ('Kalbi', 'NNP'), ('I', 'PRP'), ('always',
     'RB'), ('order', 'NN'), ('it', 'PRP'), ('the', 'DT'
    ), ('sauce', 'NN'), ('is', 'VBZ'), ('what', 'WP'), (
    'makes', 'VBZ'), ('it', 'PRP'), ('really', 'RB'), ('
    good', 'JJ')]
```

From this example, we can find that the first "love" is labelled as "VBP" (verb, present tense) not just "VB" and "makes" is "VBZ" (verb, present tense, 3rd person singular). These show that the tagger not only provides the part-of-speech labels, but also indicates other grammatical categories like tense and number. In this case, the word "order" should be labelled as "VBP" or "VB" instead of "NN" here, so the tagger classifies 16 out of 17 words correctly.

```
# Sample sentence from review 2
>>> text = word_tokenize("I reread the description of the
     burgetta burger and realized that I didn't even get
     the fried eggplant!")
>>> nltk.pos_tag(text)
[('I', 'PRP'), ('reread', 'VBP'), ('the', 'DT'), ('
    description', 'NN'), ('of', 'IN'), ('the', 'DT'), ('
    burgetta', 'NN'), ('burger', 'NN'), ('and', 'CC'), (
    'realized', 'VBD'), ('that', 'IN'), ('I', 'PRP'), ('
    didnt', 'VBP'), ('even', 'RB'), ('get', 'VB'), ('the
    ', 'DT'), ('fried', 'JJ'), ('eggplant', 'NN')]
```

From the documentation provided by NLTK, we find that "IN" stands for "preposition or conjunction, subordinating". Here, both "of" and "that" are classified as "IN" but "of" acts as a preposition while "that" is used as a conjunction. In this case, the tagger classifies 18 out of 18 words correctly.

```
# Sample sentence from review 3
>>> text = word_tokenize("Good sushi, fast service!")
>>> nltk.pos_tag(text)
[('Good', 'JJ'), ('sushi', 'NN'), ('fast', 'JJ'), ('
    service', 'NN')]
```

The tagger classifies 4 out of 4 words correctly.

```
# Sample sentence from review 4
>>> text = word_tokenize("With all the food groups
    covered, we had a really enjoyable time. ")
>>> nltk.pos_tag(text)
[('With', 'IN'), ('all', 'PDT'), ('the', 'DT'), ('food',
     'NN'), ('groups', 'NNS'), ('covered', 'VBD'), ('we',
     'PRP'), ('had', 'VBD'), ('a', 'DT'), ('really', 'RB
    '), ('enjoyable', 'JJ'), ('time', 'NN')]
```

The tagger classifies 12 out of 12 words correctly.

```
# Sample sentence from review 5
>>> text = word_tokenize("But he did ask me if I knew how
     to turn the temp knob. ")
>>> nltk.pos_tag(text)
[('But', 'CC'), ('he', 'PRP'), ('did', 'VBD'), ('ask', '
    VB'), ('me', 'PRP'), ('if', 'IN'), ('I', 'PRP'), ('
    knew', 'VBD'), ('how', 'WRB'), ('to', 'TO'), ('turn'
    , 'VB'), ('the', 'DT'), ('temp', 'NN'), ('knob', 'NN
    ')]
```

The tagger classifies 14 out of 14 words correctly.

From the five example sentences above, we can conclude that the POS-tagger of NLTK works quite well in most cases. However, while dealing with those words which can both appear as verbs and nouns, the classification accuracy still needs to be improved.

3.5 Most Frequent Adjectives

Table 2: Top 10 most frequently used adjectives for each rating star

rank	1 star	2 star	3 star	4 star	5 star
1	good	good	good	good	great
2	other	other	other	great	good
3	bad	great	great	nice	best
4	more	better	nice	other	friendly
5	worst	more	little	little	amazing
6	first	bad	more	delicious	delicious
7	new	nice	small	friendly	nice
8	last	first	better	more	other
9	rude	little	bad	fresh	fresh
10	horrible	small	decent	best	awesome

Table 3: Top 10 most indicative adjectives for each rating star

rank	1 star	2 star	3 star	4 star	5 star
1	worst	better	good	good	great
2	rude	bad	ok	great	amazing
3	horrible	ok	decent	tasty	best
4	bad	disappointed	average	nice	friendly
5	terrible	same	other	delicious	delicious
6	poor	bland	okay	little	awesome
7	last	dry	small	fresh	excellent
8	awful	chinese	little	small	helpful
9	worse	okay	better	korean	wonderful
10	same	disappointing	nice	huge	professional

We used the exact formula provided in the lab manual to derive the "indicativeness" for each adjective.

Table 2 is based on a counter over each category and the result here is not satisfactory. As the reader may have noticed, the most frequent adjective for each star except 5 is the word "good". This is not reasonable. One possible explanation could be the origin post is actually suggesting "not good" but the negation was truncated. Another possibility could be that the comment is generally negative but one may say something good at the beginning to be more polite. There are also words like "other" which do not show any opinions but are commonly used in English language included in the chart.

Table 3 makes much more sense comparing to its previous counterpart thanks to the formula. In this table, negative adjectives appear more often in the 1-2 stars columns whereas the the highly-rated side almost remain the same. If we were to make improvements, maybe we could try to do it with stemming to merge the case like "disappointed" and "disappointing".

4 PART II: <NOUN, ADJECTIVE> PAIR SUMMARIZER

4.1 Adjectival Modifier and Adjectival Complement

We implement the summarizer using APIs of the SpaCy library. An adjective can be either an adjectival modifier or an adjectival complement. Our summerizer treats the two types above separately.

Adjectival Modifier: When dealing with phrases containing adjectival modifiers, for example:

"a red apple"

we use SpaCy to parse the adjective "*red*" as an *amod* depending on the noun "*apple*".

Adjectival Complement: For sentences with adjectival complements, for example:

"The apple is red."

we use SpaCy to parse the adjective "*red*" as an *acomp* depending on the verb "*is*". Meanwhile, the noun "apple" is be parsed as a nsubj depending on the verb "*is*".

4.2 Conjunction

The Yelp dataset includes texts containing conjunctions, for example "Noun_A and Noun_B" or "Adj_A and Adj_B". SpaCy parses them into a form of "$A \rightarrow B$", where B is a *conj* depending on A. Thereafter, we identifies A and B as two different adjectives, and add them into the final summary.

4.3 Compound Noun

To deal with compound nouns, such as "chicken wing", SpaCy parses them into the form of "$wing \rightarrow chicken$", where the word *wing* is a *compound* depending on *chicken*. Our summariser identifies these compounds and concatenate them into noun phrases, like "chicken wing" and "chilli crab".

4.4 Adverbs for Adjectives

For adverbs modifying adjectives, for example "extremely big", SpaCy parses them into the form of "$big \rightarrow extremely$", where the adverb *extremely* is an *advmod* depending on *big*. We then concatenate these adverbs with corresponding adjectives and finally produce adjective phrases.

4.5 Discussion of Example Businesses and Sentences

We randomly choose 5 businesses, and count the most common <noun, adjective> pairs for each of the businesses. The results are shown in Table 4. The result of the summary is reasonable in general, but useless pairs are generated occasionally, such as $(day, next)$ and $(time, first)$.

Table 4: Top 5 most common <noun, adjective> pairs

Business ID	Pair	Frequency
ZBE-H_aUlicix_9vUGQPIQ	(food, good)	10
	(time, first)	8
	(food, great)	7
	(manager, general)	7
	(place, great)	5
e-YnECeZNt8ngm0tu4X9mQ	(bbq, korean)	14
	(food, good)	10
	(food, korean)	10
	(service, great)	6
	(food, more)	5
j7HO1YeMQGYo3KibMXZ5vg	(food, hawaiian)	14
	(noodles, fried)	10
	(rice, fried)	9
	(food, good)	8
	(portions, huge)	8
7e3PZzUpG5FYOTGt3O3ePA	(rib, prime)	10
	(service, excellent)	7
	(course, main)	7
	(service, great)	6
	(service, friendly)	5
vuHzLZ7nAeT-EiecOkS5Og	(day, next)	16
	(service, great)	7
	(pool, green)	6
	(motor, new)	5
	(time, first)	5

Also, to demonstrate the effectiveness of our summariser, we perform experiments on two sentences of the dataset, which contains Adjectival Modifier and Adjectival Complement respectively. The results show that conjunctions, compounds, and adverb modifiers are handled correctly:

Sentence 1: The crab and chicken wings were also great and extremely big.
Results: (crab, great), (crab, extremely big), (wings, great), (wings, extremely big)

Sentence 2: A popular and nice dish!
Results: (dish, popular), (dish, nice)

5 PART III: APPLICATION - NAMED ENTITY RECOGNITION

Named entity recognition (NER) a typical pre-processing step towards information extraction, where named entities are located and classified into categories such as the organizations, time expressions, locations, names of persons, monetary values, etc. We experiment on several sentences using SpaCy, and the NER results are listed in Table 5.

The NER results given by SpaCy are generally accurate, but there are still some false positives generated. For example, if the first letter of "poutine" is capitalised by mistake, the word will be classified as a person's name

Table 5: Results of Named Entity Recognition

Sentence	Named Entity
This location was a surprise to me, even though I have been here a million times. (from Review 4)	million (CARDINAL)
I bought a wine and bar fridge in August, I was told that the wine fridge had two temps, it only had one. (from Review 5)	August (DATE) two (CARDINAL) one(CARDINAL)
Gary, the sales person who said he'd check with the manager and he'd call me next day. (from Review 5)	Gary (PERSON) next day (DATE)
They didn't have any menus in English when we arrived, but, luckily Google Translate saved my day.	English (LANGUAGE) Google Translate (PRODUCT)
The Poutine was tasty with all its toppings and the burger patty was good. (from Review 2)	Poutine (PERSON)

6 CONCLUSION

By completing the assignment and this report, our group members gain hands-on experiences on NLP packages and a deeper understanding on various NLP tasks introduced in CZ4045 lectures.

REFERENCES

[1] Martin F. Porter. 1980. An algorithm for suffix stripping. *Program* 14 (1980), 130–137.
[2] Yelp. 2019. *Yelp Dataset Challenge.* Retrieved Nov 1, 2019 from https://www.yelp.com/dataset/challenge

A APPENDICES

A.1 Sentence Segmentation Results for Sample Texts

```
1: ['I love love their Kalbi, I always order it', 'the sauce is what makes it really good..', "hmm think I wanna eat that today.i didn't like their noodles to sweet.."]
2: ['We found JBs though all the hype on instagram and was excited to try the international burgers and Poutine.', 'I got the Burgetta burger and the JB House poutine.', 'The Poutine was tasty with all its toppings and the burger patty was good, but there was barely any toppings and much smaller than anticipated.', "I reread the description of the burgetta burger and realized that I didn't even get the fried eggplant!", 'I felt very mislead by the advertising and disappointed.', 'I would give this company an A for creativity but I will not be coming back for an undersized meal with an oversized pricetag.']
3: ['Good sushi, fast service! ...', 'Not the best i tasted, but surely works if you crave for sushi, and thats the only place nearby, you can go to kill the Sushi craving.']
4: ['This location was a surprise to me, even though I have been here a million times.', 'The location is also incredibly busy!', 'I was surprised how busy it was.', "Nevertheless the wait time wasn't anything crazy.", 'We were able to find a table for 2 within a reasonable amount of time.', 'We ordered a few items, from each of the different food groups: pork, beef, chicken, you get the idea.', 'With all the food groups covered, we had a really enjoyable time.', 'The food was good, and the service was speedy.', "The only drawback might have been the noise level, but there's not much you can do about that with a full restaurant on a very busy day."]
5: ['I bought a wine and bar fridge in August, I was told that the wine fridge had two temps, it only had one.', "Delivery was a joke, only had one of the items and wouldn't switch the door handles, spent more time taking off work waiting for delivery.", "Then wine fridge hasn't worked since, Spencers said it was out of their hands, the manufacturer sent an appliance person to fix it, still hasn't worked.", 'Have called Spencers numerous times and gone in person to no avail.', "Months later, I still can't use it unless I like frozen wine.", '(bottles have been ruined)', "Gary, the sales person who said he'd check with the manager and he'd call me next day... over two weeks later STILL waiting.", 'But he did ask me if I knew how to turn the temp knob.', 'Seriously.', 'Do not deal with these people.', "They get their money and don't care and are insulting to women.", 'I have filed with the BBB.']
```

A.2 Stop Words

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't", '.', ',', '!', '?', "'s", '...', '(', ')', "n't", '``', 'also', 'even', 'would', 'us', 'could', 'one', "'ve", '-', "''", "'m"]
```