CZ4045 Natural Language Processing

|  |  |  |
| --- | --- | --- |
| Li Bingzi | U1722793H | li0001zi@e.ntu.edu.sg |
| Li Guanlong | U1722033H | gli009@e.ntu.edu.sg |
| Wu Ziang | U1722432E | zwu018@e.ntu.edu.sg |
| Yong Hao | U1722282A | yo0001ao@e.ntu.edu.sg |
| Zhang Yuehan | U1722863L | yzhang145@e.ntu.edu.sg |

# **1 Data Analysis**

## **1.1 Writing Style**

There are three writing styles listed above: informal style like the reviews, formal writing style like the news articles, and technical writing style like the Python Library documentation.

**The reviews:**

Review[line1, ‘8aoJJdKEO3ypoZNszpPu7Q’] "We had my Mother's Birthday Party here on 10\/29\/16. What a Great time we all had. The food, music and waiters were Great!!! Thanks Lyles!!!"

Review[line2, ‘J5NOCLdhuhor7USRhtYZ8w’] "Good Korean grill near Eaton Centre. The marinate is good. We got beef, ox liver, salmon, fish fillet, chicken, pork, pork belly. The fish fillet was bland and liver was meh. Salmon and chicken was really flavourable. Such a fun place to eat at for a date or group of friends. Even alone. No judgments here. \nThe staff is attentive, nice and considerate. Bigger groups will most likely be seated on the second floor which is way bigger.\nCaution: will smell like BBQ grill after."

Review[line48, ’Oo8BMFOmIYxzt3fyzTA-yg’] "By far the most affordable sushi place on the west side :) I love their Hawaii Roll!! The slivers of almond get me every time!! I also love their bento special. If you're not particularly hungry, don't get it! It's a lot of food. You can get a lot of food for under $10. Definitely a place to try :) ahem ahem walk over to the donut shop next door and have a boba for dessert. Just sayin!!"

**News article:**

The Straits Times[‘Japan advises hundreds of thousands to evacuate as powerful Typhoon Hagibis approaches’]: Typhoon Hagibis, which means "speed" in the Philippine language Tagalog, is due to make landfall on Japan's main island of Honshu late on Saturday, a month after one of the strongest typhoons to hit the country in recent years destroyed or damaged 30,000 houses and caused extensive power outages.[1]

**Python Library Documentation:**

See the [plot()](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot) documentation for a complete list of line styles and format strings. The [axis()](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.axis.html#matplotlib.pyplot.axis) command in the example above takes a list of [xmin, xmax, ymin, ymax] and specifies the viewport of the axes.[2]

*1.1.1 Characteristics of the Reviews Text:*

*1.1.1.1 Informal Grammar*

Incorrect sentence structure and misspelling

E.g No subject in “Such a fun place to eat at for a date or group of friends. Even alone.”, and “group of friends” should be “a group of friends”.

*1.1.2 Informal Expressions*

*1.1.2.1 Informal Use of Capital Letters and Punctuation for Emphasis*

In the first review text, the author uses capital letters and punctuation to emphasize his/her mood, which is not applicable in the formal grammar. (e.g. ‘Great’ and ‘!!!’ in “The food, music and waiters were Great!!!”)

*1.1.2.2 Oral Expressions*

E.g ‘meh’ in “The fish fillet was bland and liver was meh.”

*1.1.2.3 Internet Slang and Emojis*

E.g ‘:) ’, ‘ahem ahem’, and ‘sayin!!’ in the

third reviews

*1.1.3 Other Characteristics*

*1.1.3.1 Sentence Length*

Average sentence length of the reviews are much shorter than that of news articles. The sentence structure of the online reviews is usually incomplete and has fewer long clauses, which is difficult to parse.

*1.1.3.2 Vocabulary*

The vocabulary used in the reviews are much less advanced than that of news articles. As mentioned above, it also includes a set of informal expressions, making NLP difficult. However, when comparing to the expressions in technical writing style, the vocabulary in reviews does not have technical expressions like math formula and the terminology, which is convenient for NLP.

*1.1.3.3 Web Text Characteristics*

There are tokens like newline and escape characters in the text of the reviews.

## **1.2 Sentence Segmentation**

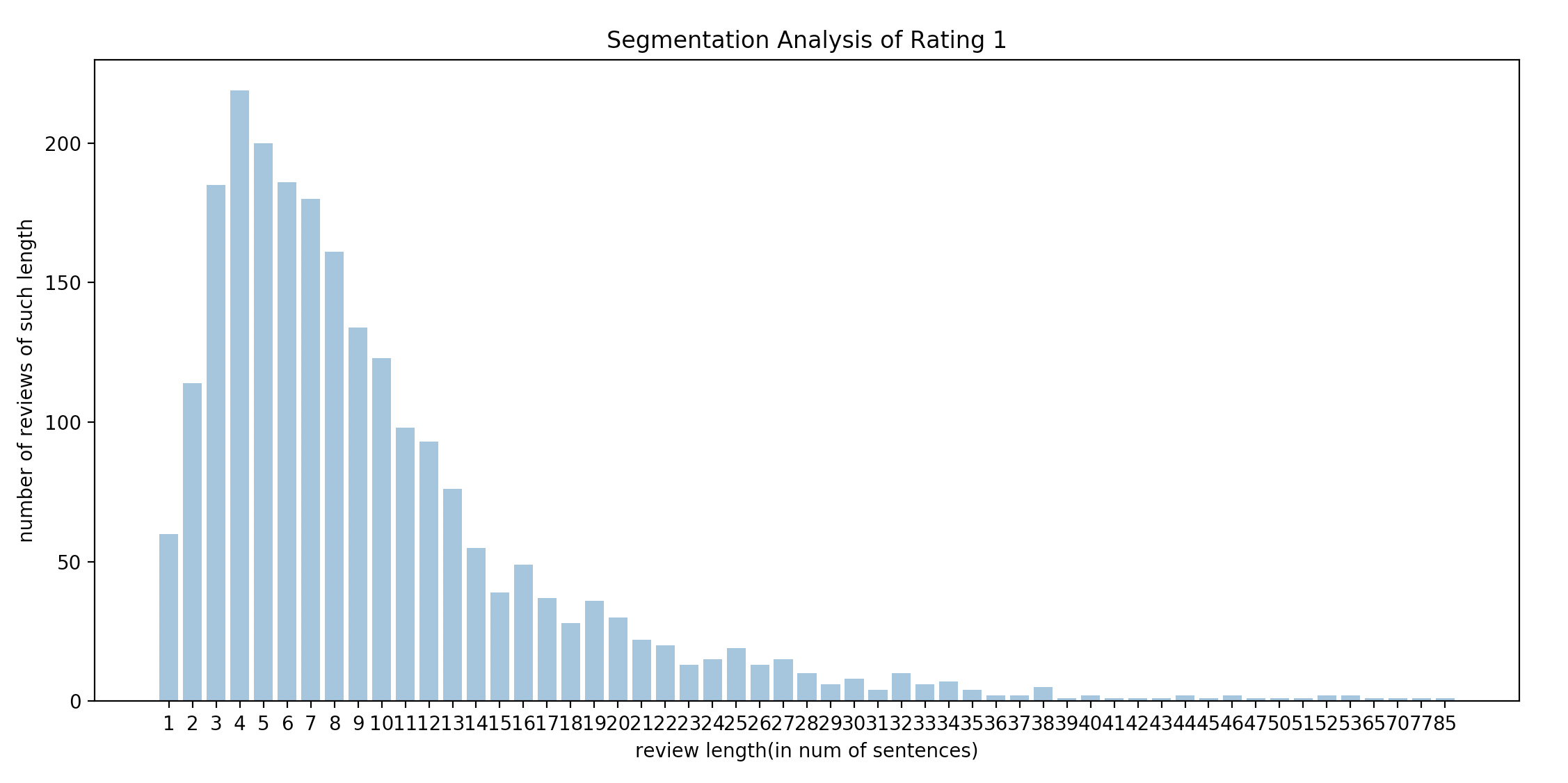


Fig 1: Segmentation result of 1 star comments

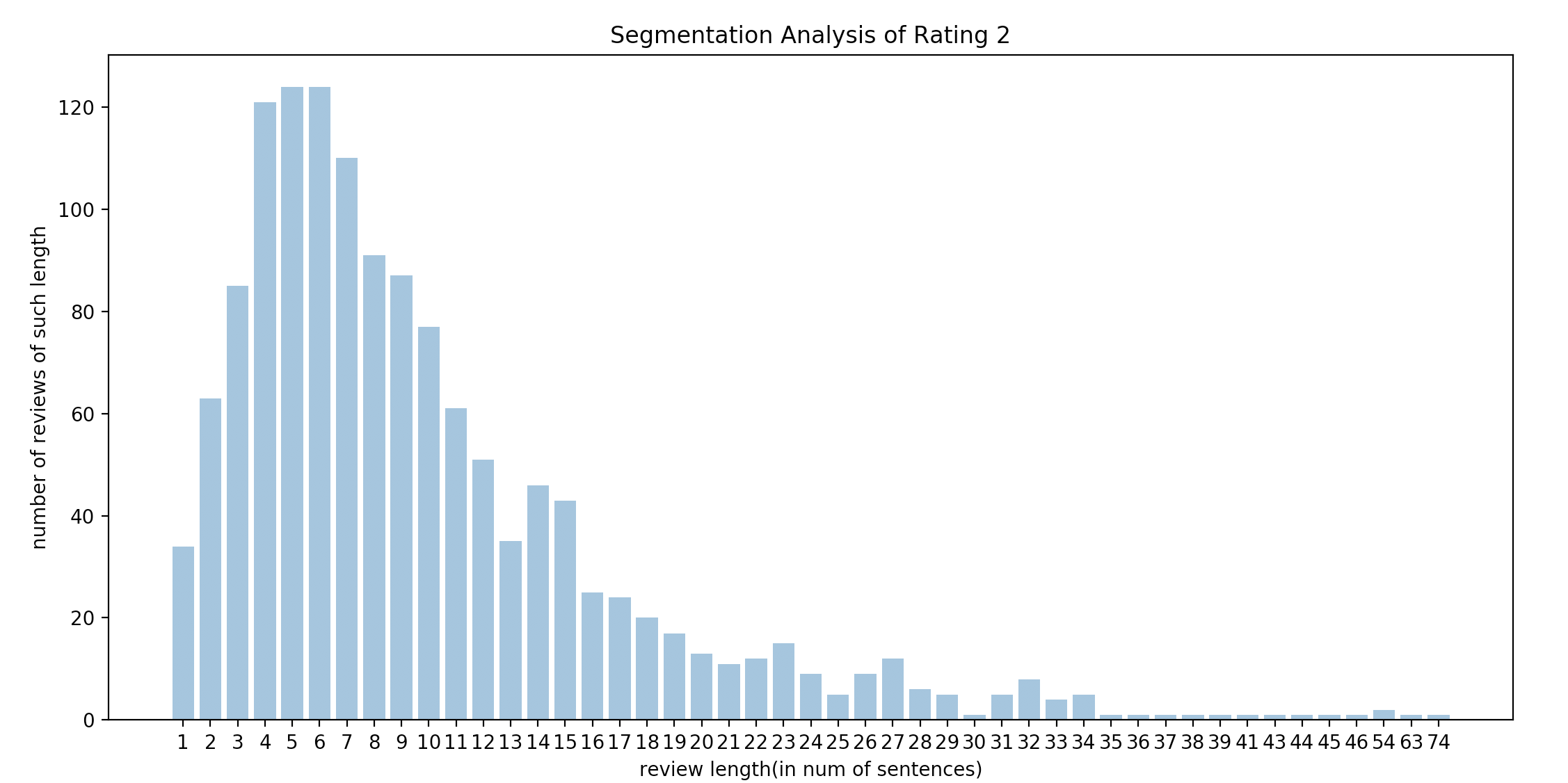


Fig 2: Segmentation result of 2 star comments

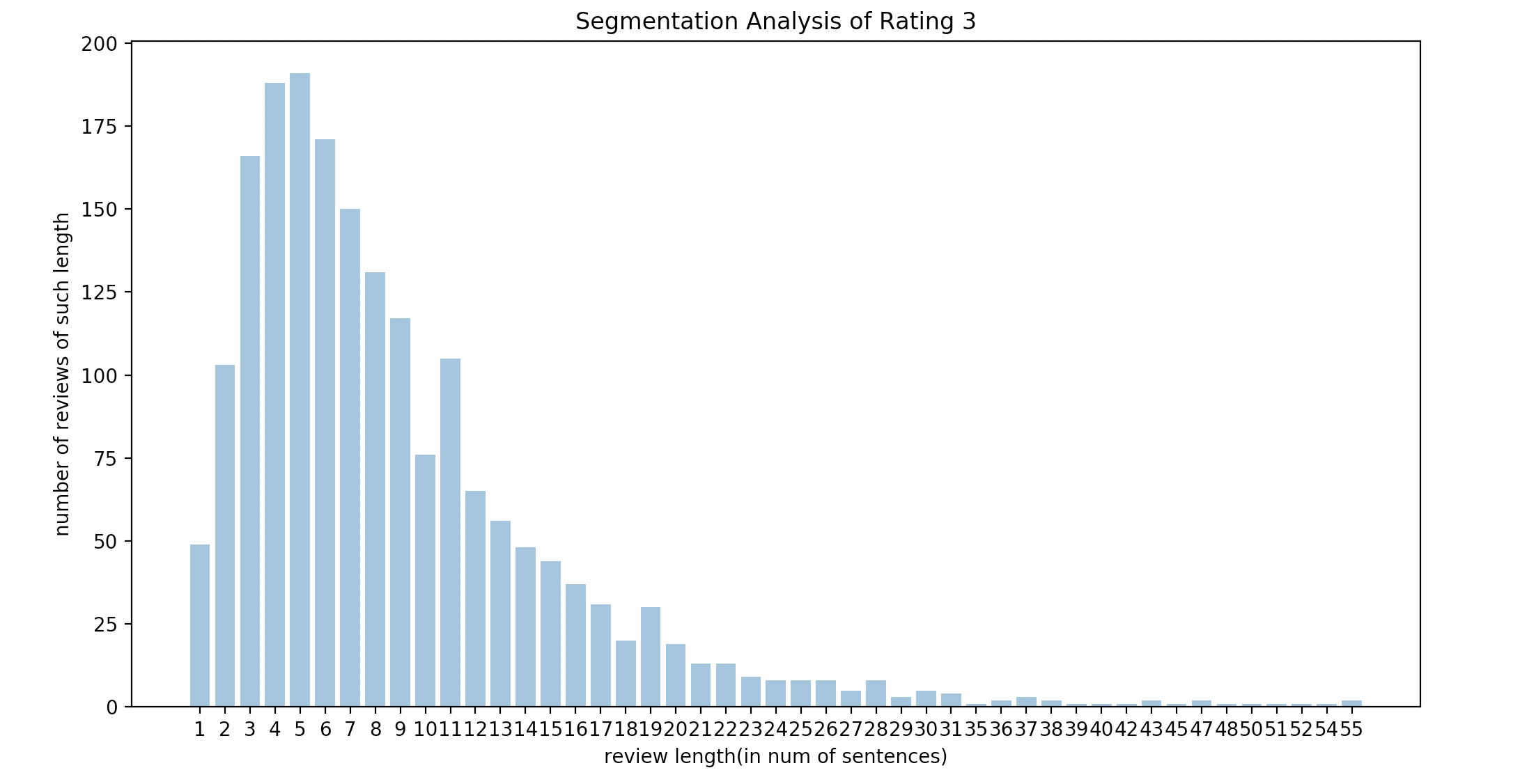


Fig 3: Segmentation result of 2 star comments

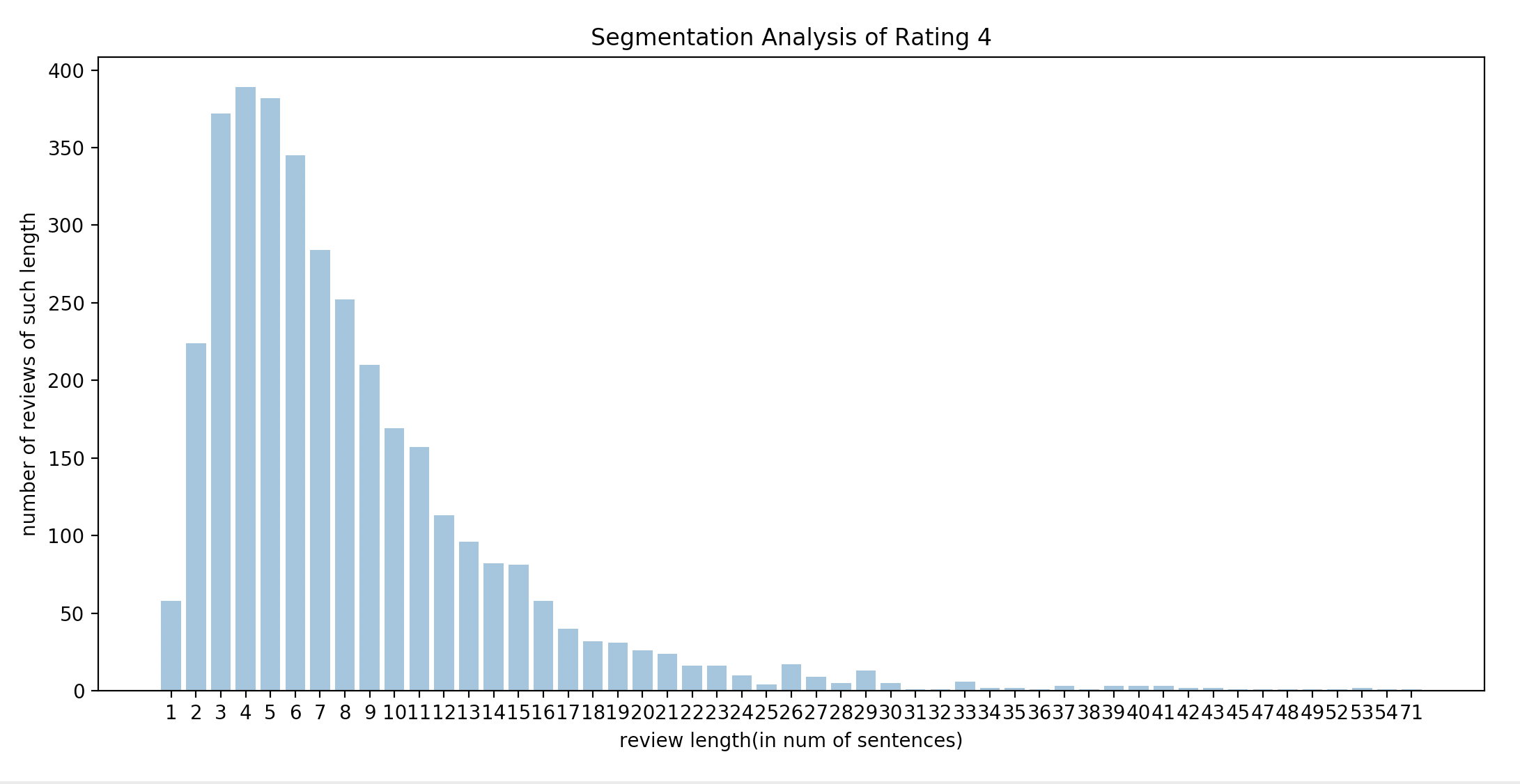


Fig 4: Segmentation result of 4 star comments

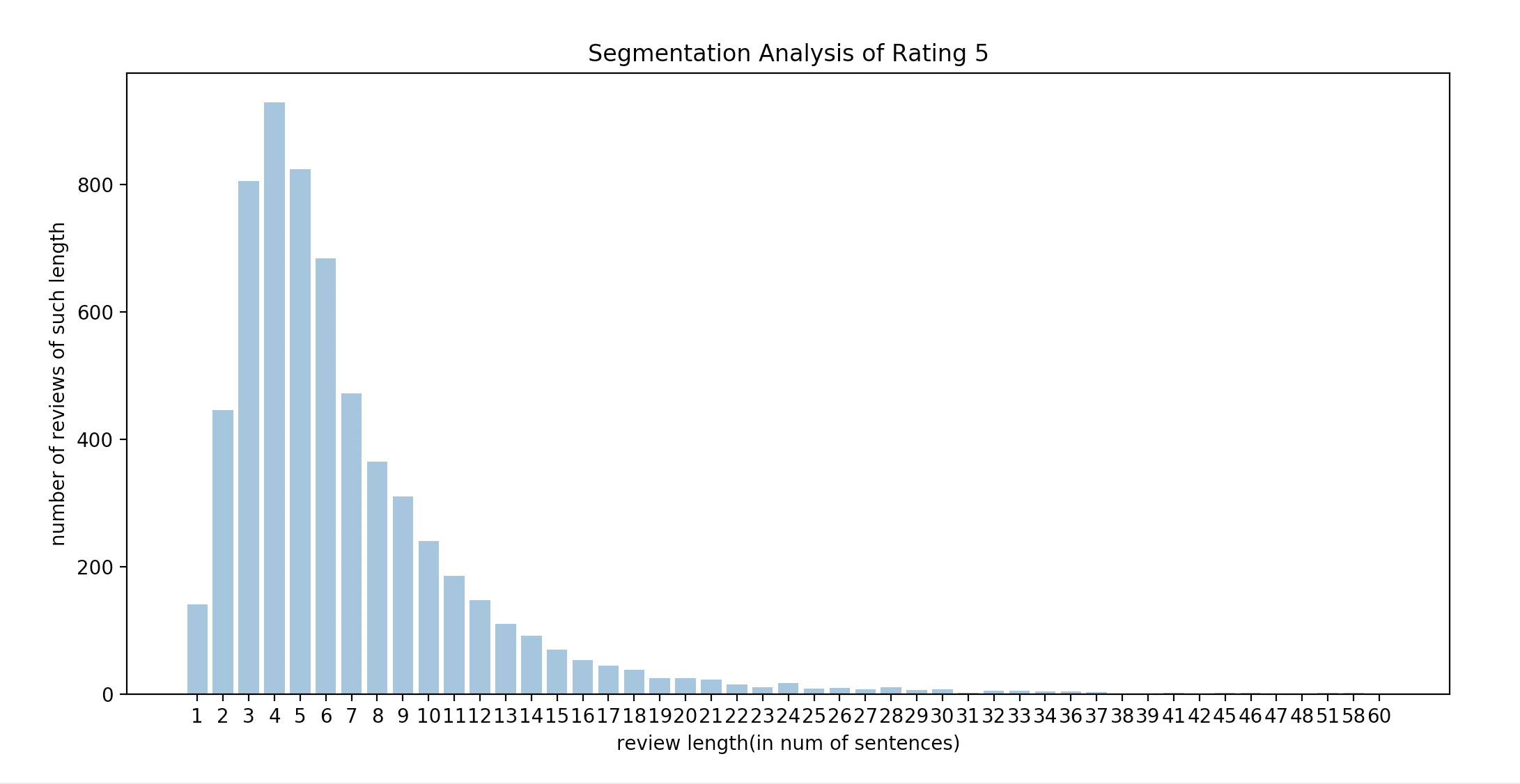


Fig 5: Segmentation result of 5 star comments

*1.2.1 Discussions:*

- Data distribution in all the graphs roughly follows the pattern of Right-Skewed Normal Distribution.

- Most frequent length occurs at 4 or 5. This shows that no matter what the rating is, most people tend to write moderate length comments. However, there are some differences between the distribution of five graphs.

- The range of the x axis are [1,85], [1,74], [1,55], [1,71], [1,60]. We can reach the conclusion that there is higher possibility for the reviewers to write very long negative comments. The neutral comments are usually shorter.

- The range of y axis for each rating are [0,219], [0,124], [0,191], [0,389], [0,929]. The total number of comments for each rating are: 2306, 1372, 1904, 3559, 6159. There are more high-rating comments than low-rating ones.

*1.2.2 Verification:*

*1.2.2.1 Line 6*

* review\_id: "Yj6KMH5yqZbNNW7XI7sZGA"
* Segmented text: ["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.."]
* num\_sentence: 1

*1.2.2.2 Line 9*

* review\_id:"dunRtl-WLvrM9ZMZUJnEHw"
* Segmented text: ["The food was amazing.", "The filet and lobster tail was perfect.", "Its hard to beat a great steak and lobster tail for under $30.", "Also the service was on point."]
* num\_sentence: 4

*1.2.2.3 Line 16*

* review\_id:"wE9JR70uPhdSW9V1MVUF-A"
* Segmented text: ["Visited for the first time today.", "Red door signature facial.", "Jordan was amazing.", "My skin looks and feels smoother and moisturized.", "The atmosphere is so elegant and relaxing .", "I'll be be back again soon!", "!"]
* num\_sentence: 7

*1.2.2.4 Line 345*

* review\_id:"y3zVxexdN1dHbInJwhI-sg"
* Segmented text: ["Worst customer service I have ever received.", "This is the first review on yelp I have left that is not 5 stars.", "I moved into a new house a promptly had an issue with the pool.", "I did not have a pool company so relied on my home warranty company First American and they send Aarons Pool.", "It was a disaster from start to finish.", "They had a no-show for my first scheduled appointment after waiting over a week.", "The second appointment he entered my backyard without my permission and supposedly replaced a capacitor.", "The company REFUSED to have someone go back to my house show me the repair!", "After being unable to resolve my complaints with the technician I called the office.", "The office refused to send the tech back out to prove to me the work had been done and HUNG UP ON ME.", "The capacitor did NOT fix my issue IF it was replaced.", "The warranty company First American sent out another company who diagnosed and replaced the pool pump all under warranty.", "Aaron's Pool was obviously only interested in making a quick dollar from the warranty company and not gaining a potential lifelong customer."]
* num\_sentence: 13

*1.2.2.5 Line 3103*

* Review\_id: "Ffw8X9PRw3YWo-nM0wWHJQ"
* Segmented text: ["I went to Mayworth with a neighbor and I truly enjoyed myself.", "I remember visiting this establishment when it was Center Street Tavern ( probably said the name wrong) years ago.", "Nevertheless, I truly enjoyed Mayworth.", "From the time we walked in until the time we left, Dana took care of us.", "Her hospitality was outstanding.", "The pub is very inviting and cozy.", "I really wanted to stay longer, but I had things to do.", "I order the Jamaican jerk wings entree with a side of sweet potato waffle fries.", "The Jamaican Jerk was not to my liking.", "I felt it wasn't spicy or seasoned at all.", "However, the fries were so delicious.", "Dana suggested mayo with it, and I said no thanks, but do you have something else?", "She had me to try the chipotle sauce and man was it good.", "Afterwards, Dana asked me about my wings.", "I had to be honest, I just told her I didn't like them at all.", "She immediately said she would get me another flavor wing if I wanted.", "I said yes, and requested the hot wings.", "When the owner found out that I wasn't satisfied he immediately came over to our table to talk with me.", "Again, I told him about the wings.", "He assured me that he would take care of me.", "The wings came out and boy were they good!", "I really enjoyed them.", "The owner comes back over and asked if I was pleased with the wings and I said yes and thanked him for taking care of me.", "This owner went above and beyond what most owners would do.", "I never got a bill so I had to asked Dana for it.", "To my surprise there was no billed because the owner took care of it.", "Most customer would say thank you right off, but not me!", "I wanted to fuss with the waitress about the owner taking care of my bill!", "He didn't have to do it, but he did and for that courtesy I'm so grateful.", "He will have my business from now on.", "This is the kind of establishment you want to embark on.", "You know you're going to be cared for here.", "Thank you for the love shown."]
* num\_sentence: 33

The first review(line 6) is a typical failure case of sentence segmentation. It should have multiple sentences, but the result is only one. The reason is the misuse of punctuation and uppercase/lowercase. Line 9, Line 345 and Line 3013 reviews are correctly segmented, these reviews, especially long reviews tends to have better grammar. Line 16 is a typical failure case that the sentence tokenizer fails if there are informal expressions, like ‘!!’ on the sentence boundary.

The tool we used for sentence segmentation is sent\_tokenize() from Python NLTK. This method comes from Punkt Sentence Tokenizer:

|  |
| --- |
| from nltk import sent\_tokenize # using sent\_tokenize() to split a review text into a list of sentences.  ... for review in datastore:  review['text'] = sent\_tokenize(review['text']) |

Punkt Sentence Tokenizer uses an unsupervised algorithm to build a model of tokens that divides a sentence, trained on a large plain text dataset of the target language. Unsupervised algorithm is a kind of algorithm that can find unknown pattern in a large dataset without existing labels in the training set.

The method sent\_tokenize() uses the model to segment the sentence. Therefore, the tokenizer fails if the sentence boundary is rare like “!!!!!”.

## **1.3 Tokenization and Stemming**

## *1.3.1 Analysis on Review Length*

## The following code snippet was executed to convert each review to a list of tokens:

tokenized\_reviews = []  
for review in reviews:  
 tokens =

nltk.tokenize.word\_tokenize(review)  
tokenized\_reviews.append(list(tokens))

A Porter Stemmer was utilized to perform stemming on each token, as shown in the following code snippet:

ps = nltk.PorterStemmer()  
stemmed\_reviews = []  
for tokenized\_review in tokenized\_reviews:  
 stemmed\_review = []  
 for token in tokenized\_review:  
 stem = ps.stem(token)  
 stemmed\_review.append(stem)

stemmed\_reviews.append(stemmed\_review)

The number of reviews of each length against the length of the review in number of tokens for both tokenization with stemming and without stemming were plotted, as shown in Figure 6 and Figure 7 below:

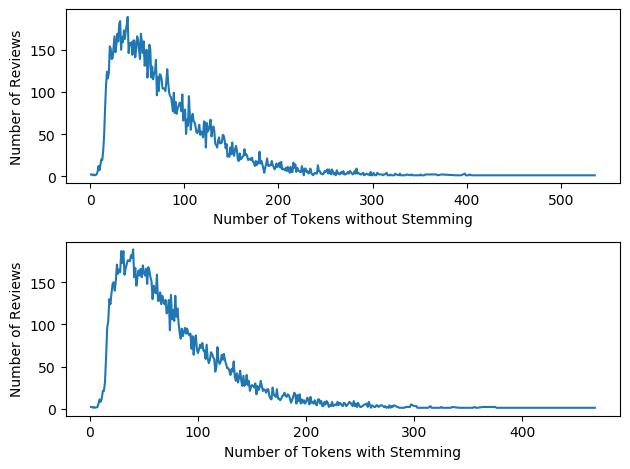


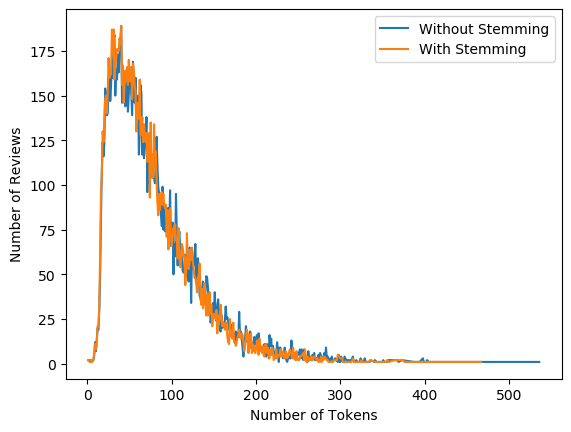
Figure 6: Results for counting number of reviews of each length

Figure 7: Results for counting number of reviews of each length combined in the same plot

It can be found that:

- After stemming is applied, the maximum number of tokens in a review changes from 536 to 467, while the minimum number of tokens in a review remains to be 1.

- Compared with the number of reviews of each length for no stemming, the curve for stemming is generally more compact towards the left hand side.When number of tokens in a sentence is around 50-80, the number of reviews of each length for stemming is generally higher than the one without stemming.

- After counting the occurrence of each word and selecting the top 20 frequent words for both tokenization with stemming and without stemming, the top 20 frequent words were plotted in a histogram, as shown in Figure 7 and Figure 8.

*1.3.2 Analysis on Word Frequency*

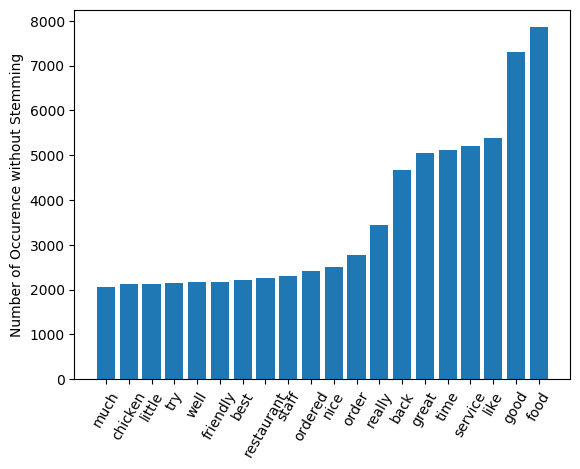


Figure 8: Histogram for top 20 frequent words when stemming is not performed

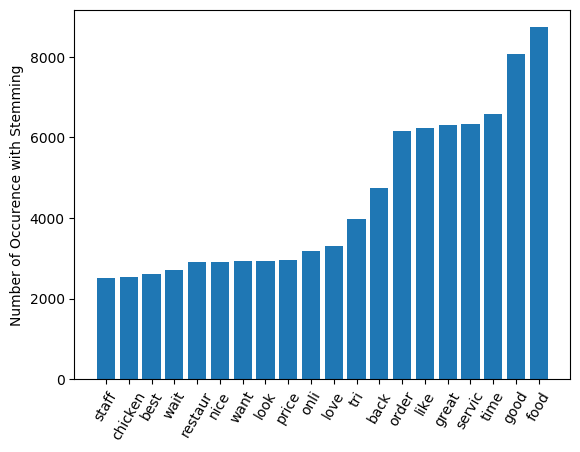


Figure 9: Histogram for top 20 frequent words after stemming is performed

The words that are expected to be popular include: ‘food’, ‘good’, ‘like’, ‘service’, ‘great’, ‘order’, ‘staff’, ‘friendly’, ‘restaurant’, ‘price’, those words are expected to be highly associated with the context of reviews on food and restaurant.

The words that are not expected to be so popular include: ‘time’, ‘back’, ‘look’. Although those words are also commonly used in reviews, their are not expected to be so popular, ‘time’ and ‘back’ are even more popular than ‘restaurant’ both for with or without stemming, and ‘look’ is more popular than ‘restaurant’ after stemming is performed.

In general, the stemming leads to an increase in the number of occurrences for most of the words since the words with the same stem but different affixes are also counted. Some of the words such as order, like, time even have an increase in rank.

## **1.4 POS Tagging**

The following code was executed to perform POS tagging on 5 random sentences:

review\_random\_df['tokenize'] = review\_random\_df['text'].apply(nltk.word\_tokenize)

review\_random\_df['pos\_tag'] = review\_random\_df['tokenize'].apply(nltk.pos\_tag)

However, since the random selected sentences all contain a fairly long text, five relatively short sentences were selected to carry out the POS tagging.

The following are POS tagging results for the 5 selected sentences:

Some**/DT** great**/JJ** Hawaiian**/JJ** food/**NN** and/**CC** friendly/**JJ** staff/**NN** !/**.**

service/**NN** was /**VBD** just/**RB** as/**RB** good/**JJ** as/**IN** the/**DT** food/**NN** !/**.** !/**.** !/**.**

I/**PRP** 'm/**VBP** sad/**JJ** that/**IN** this/**DT** restaurant/**NN** is/**VBZ** closed/**VBN** .../:

Clean/**NNP** ,/**,** spacious/**JJ** ,/**,** quite/**RB** ,/**,** and/**CC** nice/**JJ** breakfast/**NN**

One/**CD** of/**IN** the/**DT** best/**JJS** sushi/**NN** places/**NNS** in/**IN** Mississauga/**NNP** ./**.**

From the results above, it can be concluded that the results produced by the POS tagger is already fairly accurate. For example, some words like Mississauga was correctly tagged as NNP, and all the special characters such as ‘!’ and ‘,’ were not tagged.

## **1.5 Most Frequent Adjectives for each Rating**

To analyse the most frequently used adjectives and the most indicative adjectives in regard to the rating level of the reviews, we have adopted the results from 1.4 which are the tagged tokens of each review and found their frequencies using the nltk module named Frequency Distribution. The Frequency Distribution module counts the number of times a certain word appeared in the list of tokens and calculates the respective percentage of the word.

We have firstly extracted the adjectives from the reviews of each rating and counts the most frequent adjectives afterwards. The related code is given below:

|  |
| --- |
| data\_length = len(review\_df) for stars in range(1, 6):  output = {'rating': stars, 'adjs':[]}  for i in range(0, data\_length):  review = review\_df.iloc[i]  if review['stars']==stars:  tagged = review['pos\_tag']  review\_adjs = [word for word, tag in tagged if tag in ('JJ')]  output['adjs'] = output['adjs'] + review\_adjs  adj\_fdist = FreqDist(output['adjs'])  print(adj\_fdist.most\_common(5)) |

To analyse the “indicativeness” of adjectives, we adopted the formula given in the assignment manual. In addition to measure the frequencies in respect to each rating, we have also measured the frequencies of each adjective in the context of the entire review. The related code is given below:

|  |
| --- |
| data\_length = len(review\_df) data\_tokens = [] for i in range(0, data\_length):  data\_tokens = data\_tokens + review\_df.iloc[i]['tokenize'] data\_fdist = FreqDist(data\_tokens) |

The result of this section is given below:

First part: Select adjectives and find their probability in regard to their ratings (With the form (word, frequency))

Star 1: [('good', 633), ('other', 532), ('bad', 433), ('new', 319), ('first', 313)]

Star 2:[('good', 742), ('other', 381), ('great', 284), ('bad', 236), ('nice', 209)]

Star 3:[('good', 1453), ('other', 531), ('great', 474), ('nice', 419), ('little', 375)]

Star 4:[('good', 2494), ('great', 1480), ('nice', 775), ('other', 740), ('little', 684)]

Star 5:[('great', 2586), ('good', 1944), ('friendly', 910), ('delicious', 887), ('nice', 832)]

Second part: Select adjectives and find their “indicativeness” in regard to their ratings (With the form (word, “indicativeness”)):

Star 1:[('other', 0.10535791798617213), ('bad', 0.10205823098735854), ('good', 0.09300169002719595), ('new', 0.06812092434855212), ('last', 0.06447757868340563)]

Star 2:[('good', 0.1800886843953514), ('other', 0.10107238349073752), ('bad', 0.0685405989872825), ('great', 0.052983999382220925), ('same', 0.05001135535070566)]

Star 3:[('good', 0.28469250735569845), ('other', 0.1003930653574356), ('nice', 0.07713290687241875), ('little', 0.07043144019426874), ('great', 0.06875275208462026)]

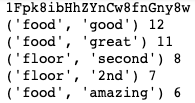
Star 4:[('good', 0.27599695834442534), ('great', 0.15498514352582435), ('nice', 0.08276697642903924), ('little', 0.07413324007214017), ('other', 0.07306323023357603)]

Star 5:[('great', 0.2344860461558968), ('good', 0.13827683872780774), ('delicious', 0.08037619600838287), ('friendly', 0.07707134327823334), ('nice', 0.06465089293410879)]

# **2 <Noun-Adjective Summarizer>**

## <Noun-Adjective> pairs are extracted from reviews of 5 randomly selected business, 100 reviews each. Each pair has a count to denote the frequency. Five most frequent <Noun-Adjective> pairs for each business are selected to discuss the correctness of this summarizer corresponding to the reviews text.

**Business\_id[‘1Fpk8ibHhZYnCw8fnGny8w’]**  **Business Name: Baro**

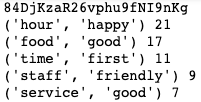
****

By observing the Noun-Adjective pairs, this business can be summarized to be a restaurant with delicious food. Among the 5 most frequently appeared pairs, (food, good) (food, great) (food, amazing) take up 3 positions.

After reading some of the reviews, we find that most customers do enjoy the food here and this matches the result generated by the summarizer.

**Businuss\_id[‘84DjKzaR26vphu9fNI9nKg’]**

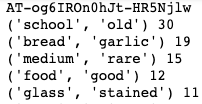
**Business Name: Sierra Gold**

****

The business is a bar, such that the pair: (hour, happy) has a very high frequency in the reviews. Although the top 5 frequent pairs are positive, a number of the reviews are actually negative. However, negative reviews are mostly describing some complex situations which cannot be extracted to noun-adjective pairs.

**Business\_id[‘AT-og6IROn0hJt-HR5Njlw’]**

**Business Name: Octagon**



This business is a very well-known steak house existed for a long time. (school, old) is the most frequently appeared Noun-Adjective pair. Many customers’ review mentioned that they have hard this long time ago and came because its famous reputation. Although upon seeing this pair, we can understand what it wants to express, a problem is old-school can actually be written as one word (Adjective) that describes a style.

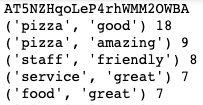
Also, (medium, rare) appeared many times since this business is a steak house, medium-rare is somehow combined with the order of steak. But the same problem as ‘old school’ is medium-rare is actually one word (Adjective) that describes how a steak will be cooked.

However, from observations of the summarizer, we can see that (food, good) has lower rank compared with others. And it is true that after seeing some of the reviews we found that food here does not reach many customers’ expectations.

In this way, we say that this summarizer overall correctly generalizes the key characteristic of this business. But the Noun-Adjective form Adjective word cannot be actually correctly identified.

**Business\_id[‘AT5NZHqoLeP4rhWMM2OWBA’]**

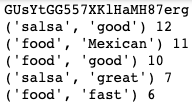
**Business Name: Novanta**



This is a Pizza restaurant, the adj-noun pairs reflect about the business.

**Business\_id[‘GUsYtGG557XKlHaMH87erg’]**

**Business Name: Chico's**

****

The business is a Mexican food restaurant. The comments do mention a lot about the good salsa in this restaurant. And the food is fast and good. The noun-adj summarizer correctly generates keywords of the business.

# **3 Application**

This application aims to understand how reviewers like the restaurant by sentiment analysis over text. We begin with some pretrained sentiment analysis tools provided in various libraries.

**NLTK Vader Sentiment Analysis**

Vader tool uses the Bag of Words approach, a dictionary of words assigned with sentiment value from -1 to 1 for degree of positive and negative, and a heuristics function for depending adverb phrase.

For example, the negative adjective “dumb” gives the sentiment of -0.55, and the pharse “so dumb” adds the intensity and gives an overall -0.72.

Another example, although the positive verb “like” raise the sentiment to 0.36, heuristics function flips the sentiment of “do not like” because “like” in this case follows a negation. Vader’s heuristics function takes care of some subtle details in the sentence, such as the negation here.

One disadvantage of this approach is that it could not estimate the intensity for unseen word and classify it as neutral by default.

**Flair Sentiment Analysis**

Flair’s sentiment classifier, developed by Zalando Research group, is based on a character-level Long Short-Term Memory (LSTM) neural network.

LSTM neural network has the feedback connections and can not only process single data point but also the entire sequences of data. Flair takes sequences of letters and words into account when evaluating the overall sentiment of the sentences.

Similar to Vader, negations and adverb phrase as intensifier are handled. An advantage of Flair is that it can predict a sentiment for unseen words, such as misspelt words.

**Stanford NLP Parser**

Stanford NLP Parser applies the POS tagging to the input sentence and returns the dependency tree and typed dependencies tree. POS tagging focuses on the detailed relationships between words and phrases with high accuracy.

Stanford NLP Parser has advantage over detecting negation expressions, such as “bad”, “really bad”, and“not good”, with typed dependencies. However, Some sophisticated process would be required to generalize word-level POS tagging dependencies to sentence-level or text-level sentiment.

We further develop our application based on the two libraries evaluated, NLTK Vader and Flair.

First, sentiment analysis over the entire text, that is model evaluates the sentence-level relationship and then gives overall text sentiment, is experimented. It performs well over the text at the expense of long execution time, 2.54s for a 301-word which makes it unscalable to the large dataset.

Sentence-level sentiment analysis is in fact not necessary, because of the nature of the review text. It is commonly observed that review is composed of straightforward and logic-independent sentences. Besides, reviewers tend to spend more effort on things they care about most, i.e. write several long sentences to describe.

This motivates us to the weighted-sentiment value. Review test is first segmented into sentences and record the word count for each sentence. Sentiment analysis tools, Vedar and Flair, give the sentiment values of the sentences. Sentiment value for overall text is then calculated from the sentiment values of the sentences weighted average by word counts.

To examine the accuracy of two models and our weighted-analysis approach, we compare our labels to the star rated by the reviewer. Two review texts below are examples where Vader and Flair mislabel the positive and negative.

Examples over 20 review samples are recorded in the spreadsheet appendix to this report.

**Example 1 - Failure of Flair**

“*Love this place downtown but the Scottsdale location has no manners. Sat at bar for 10 min while bartender ignored us. No menu, no water. We walked out and they could have cared less.*”

- Reviewer Star: 1

- Vader

- Weighted Average: -0.060673529

- Sentences: [-0.0516, -0.3182, -0.5267, 0.4215]

- Flair

- Weighted Average: 0.996961765

- Sentences: [0.9997, 0.9987, 0.9868, 0.9962]

**Example 2 - Failure of Vader**

"*I have no idea what the owner's problem is, but he's incredibly rude. His wife, on the other hand is super nice (an odd mix). Great space, rude owner and the coffee is average.*"

- Reviewer Star: 1

- Vader

- Weighted Average: 0.015841667

- Sentences: [-0.7839, 0.8225, 0.2732]

- Flair

- Weighted Average: -0.473241667

- Sentences: [-0.9944, -0.9265, 0.9997]

Possible explanation for failure observed above are:

- The possible reason could be the reviewer’s definition for the level of satisfaction represented by each star given by themselves is different. For example, customer A and customer B both write the same sentence, “Great food but rude service”, in the review, however the star given by A is 2 while B is 1. It occurs could just because B is more cared about the service provided while A may focus more on the food.

- Weighted average is calculated in a way such the weight for each sentence is calculated based on their length in terms of words. Some keyword may not be sufficiently highlighted in this case.

- The fact the model’s training data set is from IMDB’s dataset, a movie review website’s dataset, which might not perfectly reflect the conditions of restaurant reviews.

# **4 Contribution**

Li Bingzi

* Writing style analysis
* Sentence segmentation
* Plot analysis

Li Guanlong

* Tokenization analysis
* Stemming analysis

Wu Ziang

* POS tagging
* Application

Yong Hao

* Most frequent adjective for each rating
* Application

Zhang Yuehan

* <None-Adjective> Summarizer

# **5 Reference**

[1] W. Sim, “Typhoon Hagibis: 2 dead, millions told to flee in Japan; quake triggers fresh mudslide warning,” The Straits Times, 12 Oct 2019. Available: [https://www.straitstimes.com/asia/east-asia/japan-advises-hundreds-of-thousands-to-evacuate-as-powerful-typhoon-approache](https://www.straitstimes.com/asia/east-asia/japan-advises-hundreds-of-thousands-to-evacuate-as-powerful-typhoon-approaches)

[2] Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

[3] Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long short-term memory". Neural Computation. 9 (8): 1735–1780. doi:10.1162/neco.1997.9.8.1735. PMID 9377276.

## **5.1 Open Source Project**

Flair: State-of-the-Art Natural Language Processing

<https://research.zalando.com/welcome/mission/research-projects/flair-nlp/>

Stanford typed dependencies using coreNLP

<https://stackoverflow.com/questions/56527814/stanford-typed-dependencies-using-corenlp-in-python>

# 

# Appendix A: List of Stop Words

Stop words provided by nltk:

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,

Additional stop words utilized:

The, n't, I, 's, We, get, would, It, one, place, They, go, This, 've, got, us, my, could, also, even, 'm, always, came, come, still, made, said, going, know, If, day, 're, two, 2, say, take, way, ever, give, told, eat, minutes, around, So, asked, see, There, since, 'll, took, She, He, You, next, But, 3, And, 5, When, every, A, something, tried, ca, Not, everything, 'd, called, done, coming, things, wa, thi, veri, realli, becaus, alway, ha, ani, use, reallyThis, My, make, went, ask, definit, definitely, work, thing