

# CS4248 Assignment 1: Regexs and Language Models

By A0184679H

## 1. Declaration of Original Work

By entering my student ID below, I certify that I completed my assignment independently of all others (except where sanctioned during in-class sessions), obeying the class policy outlined in the introductory lecture. In particular, I am allowed to discuss the problems and solutions in this assignment, but have waited at least 30 minutes by doing other activities unrelated to class before attempting to complete or modify my answers as per that Pokemon Gó rule.

## 2. References

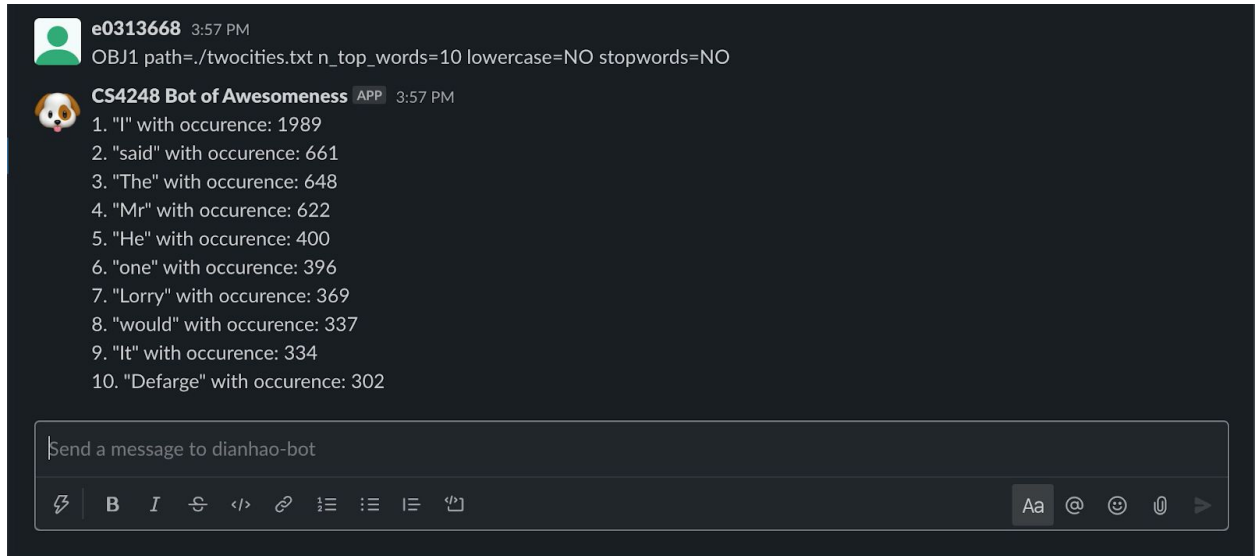
I give credit where credit is due. I acknowledge that I used the following websites or contracts to complete this assignment.

- Geeks For Geeks: [Removing stop words with NLTK in Python - GeeksforGeeks](#) for learning about the set of stop words
- Python's Collections documentation: [collections — Container datatypes — Python 3.9.1 documentation](#) for learning collections Counter methods
- Python's Regular Expression documentation: [re — Regular expression operations — Python 3.9.1 documentation](#) for learning re methods
- Matplotlib documentation: [matplotlib.pyplot — Matplotlib 3.3.3 documentation](#) for plotting a graph
- StackOverflow, for debugging purposes
- [Text Generation Using N-Gram Model | by Oleg Borisov | Towards Data Science](#) to learn how to build n-gram model
- [text mining - How to find the perplexity of a corpus - Cross Validated \(stackexchange.com\)](#) for implementation of perplexity formula in python
- [3.pdf \(stanford.edu\)](#) SLP textbook for additional reference
- CS4248 lecture notes for better understanding of contents

## Part 1

### Objective 1 - Tokenization, Zipf's Law

#### A. No stopwords, no lowercase:



The screenshot shows a chat window with a dark background. At the top, a user with a green profile picture and ID 'e0313668' sends a message at 3:57 PM: 'OBJ1 path=./twocities.txt n\_top\_words=10 lowercase=NO stopwords=NO'. Below this, a bot with a dog profile picture and name 'CS4248 Bot of Awesomeness' (with an 'APP' badge) responds at 3:57 PM with a numbered list of 10 words and their occurrence counts. At the bottom is a text input field with the placeholder 'Send a message to dianhao-bot' and a rich text toolbar with icons for bold, italic, link, code, linkify, bulleted list, numbered list, indent, and image, followed by a 'Aa' button and other communication icons.

e0313668 3:57 PM  
OBJ1 path=./twocities.txt n\_top\_words=10 lowercase=NO stopwords=NO

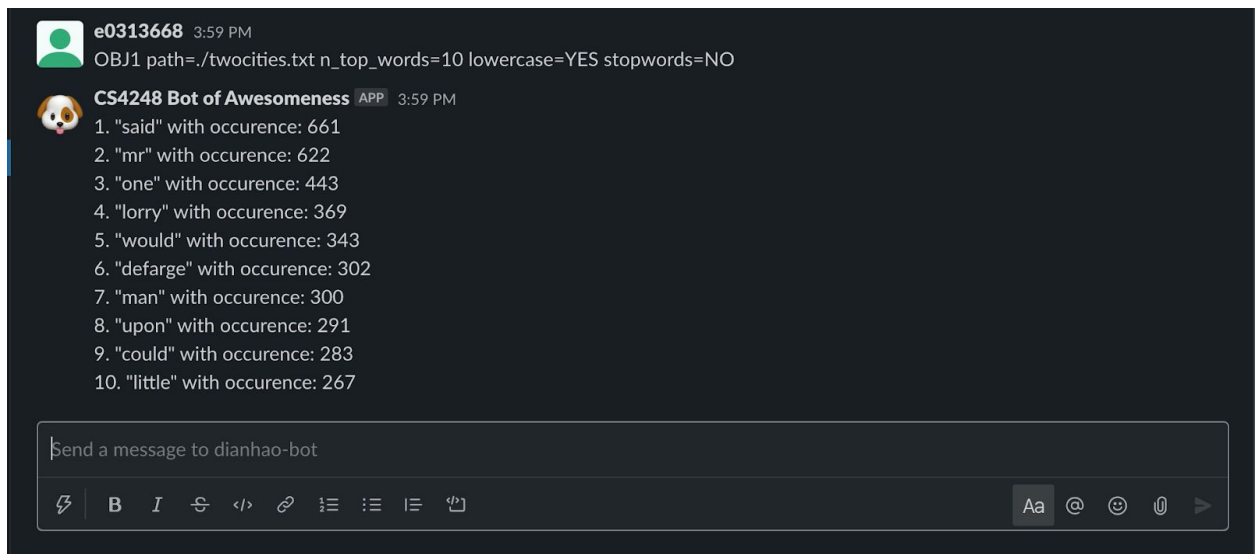
CS4248 Bot of Awesomeness APP 3:57 PM

1. "I" with occurrence: 1989
2. "said" with occurrence: 661
3. "The" with occurrence: 648
4. "Mr" with occurrence: 622
5. "He" with occurrence: 400
6. "one" with occurrence: 396
7. "Lorry" with occurrence: 369
8. "would" with occurrence: 337
9. "It" with occurrence: 334
10. "Defarge" with occurrence: 302

Send a message to dianhao-bot

⚡ B I 🔗 </> 🔗 ☰ ☷ |≡ 📎 Aa @ 😊 📎 ➡

#### No stopwords, all lowercase:



The screenshot shows a chat window with a dark background. At the top, a user with a green profile picture and ID 'e0313668' sends a message at 3:59 PM: 'OBJ1 path=./twocities.txt n\_top\_words=10 lowercase=YES stopwords=NO'. Below this, a bot with a dog profile picture and name 'CS4248 Bot of Awesomeness' (with an 'APP' badge) responds at 3:59 PM with a numbered list of 10 words and their occurrence counts. At the bottom is a text input field with the placeholder 'Send a message to dianhao-bot' and a rich text toolbar with icons for bold, italic, link, code, linkify, bulleted list, numbered list, indent, and image, followed by a 'Aa' button and other communication icons.

e0313668 3:59 PM  
OBJ1 path=./twocities.txt n\_top\_words=10 lowercase=YES stopwords=NO

CS4248 Bot of Awesomeness APP 3:59 PM

1. "said" with occurrence: 661
2. "mr" with occurrence: 622
3. "one" with occurrence: 443
4. "lorry" with occurrence: 369
5. "would" with occurrence: 343
6. "defarge" with occurrence: 302
7. "man" with occurrence: 300
8. "upon" with occurrence: 291
9. "could" with occurrence: 283
10. "little" with occurrence: 267

Send a message to dianhao-bot

⚡ B I 🔗 </> 🔗 ☰ ☷ |≡ 📎 Aa @ 😊 📎 ➡

With stopwords, no lowercase:

The screenshot shows a Telegram chat interface. At the top, a user with ID e0313668 sends a message at 4:00 PM: "OBJ1 path=./twocities.txt n\_top\_words=10 lowercase=NO stopwords=YES". Below this, a bot named "CS4248 Bot of Awesomeness" (with a dog icon) responds at 4:00 PM with a list of 10 words and their occurrence counts. The list is as follows:

Rank	Word	Occurrence
1.	"the"	7587
2.	"and"	4925
3.	"of"	4106
4.	"to"	3603
5.	"a"	2864
6.	"in"	2546
7.	"I"	1989
8.	"his"	1942
9.	"that"	1846
10.	"was"	1767

Below the list is a text input field with the placeholder "Send a message to dianhao-bot" and a rich text editor toolbar with icons for bold, italic, link, unlink, list, and image.

With stopwords, all lowercase:

The screenshot shows a Telegram chat interface. At the top, a user with ID e0313668 sends a message at 4:45 PM: "OBJ1 path=./twocities.txt n\_top\_words=10 lowercase=YES stopwords=YES". Below this, the same bot "CS4248 Bot of Awesomeness" responds at 4:45 PM with a list of 10 words and their occurrence counts. The list is as follows:

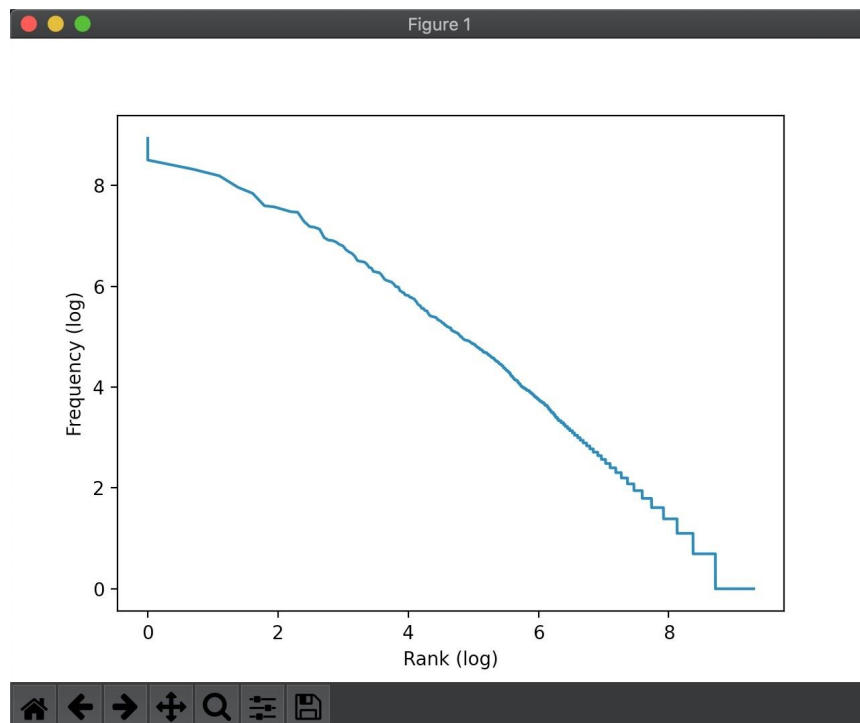
Rank	Word	Occurrence
1.	"the"	8241
2.	"and"	5071
3.	"of"	4143
4.	"to"	3653
5.	"a"	3017
6.	"in"	2665
7.	"it"	2082
8.	"his"	2011
9.	"i"	1990
10.	"that"	1956

Below the list is a text input field with the placeholder "Send a message to dianhao-bot" and a rich text editor toolbar with icons for bold, italic, link, unlink, list, and image.

The corpus I utilised is "A Tale of Two Cities" by Charles Dickens, with a file size of 791 KB. I analysed the result with 4 different conditions as shown above. According to the screenshots, when stopwords are not counted, the word with the most frequency is only 661 for "said" when all characters are lowercase while it is 1989 for "I" when the lowercase condition is false. This is expected because the word "i" a stopword.

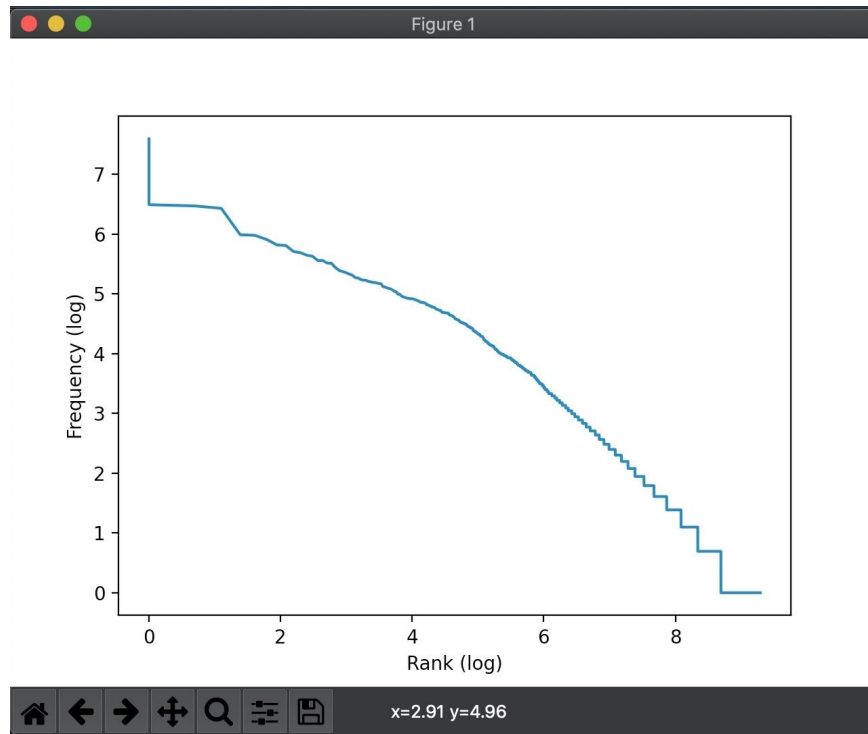
When stopwords are counted, the word with the most frequency is “the” with 7587 occurrences for all lowercase, and also “the” with 8241 occurrences when the lowercase condition is not enforced. The results for both are almost the same, as the words in these results tend to start the beginning of a sentence, and are often stopwords, such as “I”, “that”. One observation is that the word “was” appeared as one of the most frequent words when lowercase, and it is expected because “was” is used to start a sentence, in particular, a question. The result is accepted.

## B. With stopwords



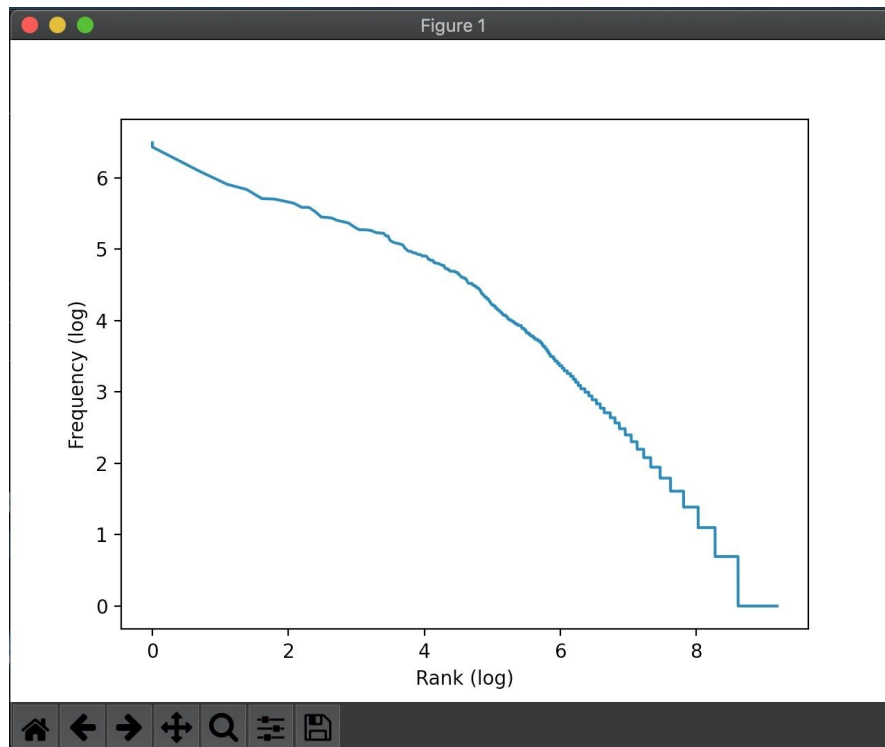
When stopwords are included, the graph of frequency vs rank (in logs) follows Zipf's Law.

### C. Without stopwords



When stopwords are removed, the wordcounts are lower compared to the previous graph, but it is still consistent with Zipf's Law.

### D. Without stopwords, lowercase



When stopwords are excluded and the corpus is converted to lowercase, Zipf's Law still holds, albeit the frequency of the words are further decreased due to stop words like "I" are not counted.

## Objective 2 - How's the Weather?

The strategy of distinguishing between a valid query and an invalid input lies in the users' intentions. A valid query can be categorized into two kinds: a question, and a sentence.

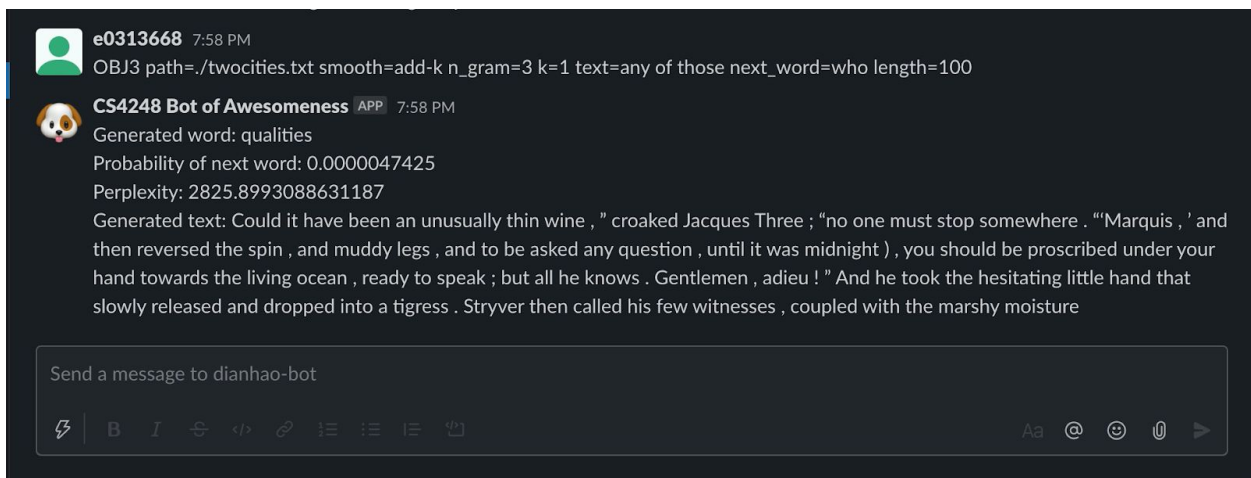
- Question:
  - Observation 1: Certain asking words must exist in a question, such as "what, how, do you know". I prepared a list of these valid words to handle most kinds of asking words.
  - Observation 2: Certain words concerning weather must exist in the query, such as "weather, temperature, foggy, raining". However, these words may have suffix, so I used normalization to reduce the words to its simplest form, and drop alphabets if necessary, such as "freez" to handle "freezing" and "freezy".
  - Observation 3: A location must exist in the question, and either be Singapore, Cairo or London. I used Python's inbuilt regex function to ignore uppercasings (as majority users do not

concern uppercase in text messages), and used regex capturing groups to determine the location that the user is querying.

- Observation 5: The sentence must end with a question mark.
- Observation 4: It is also valid for the question to be juxtaposed, such as "can you tell me weather in Singapore?" and "can you tell me Singapore's weather"? My approach is to generate all regexes combinations of the above observations, and check if any of them is a match.
- Sentence:
  - Observation 1: For a valid query, it must have some kind of instructional word, eg "tell me" or "ask".
  - Observation 2: It also requires a word that describes the weather context such as "humid" or "weather".
  - Observation 3: The location is required.
  - Observation 4: The sentence can be inversed too, such as "tell me the weather in singapore", and "tell me singapore's weather".
  - Observation 5: To handle non valid queries, a sentence must include an action word and a weather word to form a valid response. For ambiguous queries like "Singapore weather", the bot will choose not to answer the query, but "Singapore weather?" is a valid response in this case.

### Objective 3:

Some captured outputs:

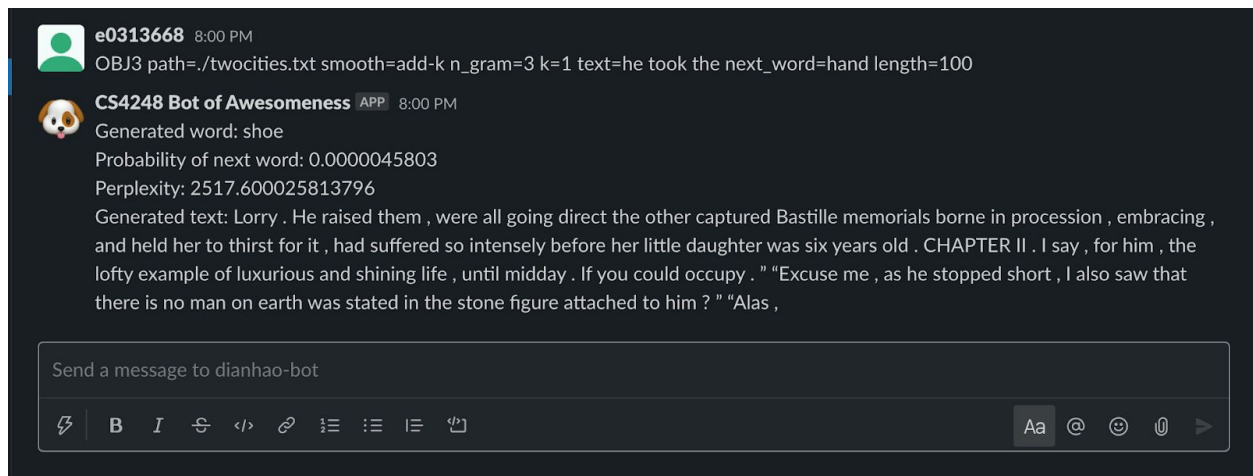


The screenshot shows a chat window with a dark background. At the top, a user with the ID 'e0313668' sends a message at 7:58 PM: 'OBJ3 path=./twocities.txt smooth=add-k n\_gram=3 k=1 text=any of those next\_word=who length=100'. Below this, a bot with the ID 'CS4248 Bot of Awesomeness' responds at 7:58 PM. The bot's response includes: 'Generated word: qualities', 'Probability of next word: 0.0000047425', 'Perplexity: 2825.8993088631187', and a paragraph of generated text: 'Generated text: Could it have been an unusually thin wine , " croaked Jacques Three ; "no one must stop somewhere . "Marquis , ' and then reversed the spin , and muddy legs , and to be asked any question , until it was midnight ) , you should be proscribed under your hand towards the living ocean , ready to speak ; but all he knows . Gentlemen , adieu ! " And he took the hesitating little hand that slowly released and dropped into a tigress . Stryver then called his few witnesses , coupled with the marshy moisture'. At the bottom of the chat window, there is a text input field with the placeholder 'Send a message to dianhao-bot' and a rich text editor toolbar with icons for bold, italic, link, unlink, list, and image.

e0313668 7:58 PM  
OBJ3 path=./twocities.txt smooth=add-k n\_gram=3 k=1 text=any of those next\_word=who length=100

CS4248 Bot of Awesomeness APP 7:58 PM  
Generated word: qualities  
Probability of next word: 0.0000047425  
Perplexity: 2825.8993088631187  
Generated text: Could it have been an unusually thin wine , " croaked Jacques Three ; "no one must stop somewhere . "Marquis , ' and then reversed the spin , and muddy legs , and to be asked any question , until it was midnight ) , you should be proscribed under your hand towards the living ocean , ready to speak ; but all he knows . Gentlemen , adieu ! " And he took the hesitating little hand that slowly released and dropped into a tigress . Stryver then called his few witnesses , coupled with the marshy moisture

Send a message to dianhao-bot



- Optional implementation: backoff
  - If backoff is enabled, when an unknown context of n-gram is introduced, the (n - 1) gram will be used, and the backoff is carried out until the (n - xth) gram is found or the unigram is used.

## Part 2

### 1. Subtraction Regular Expressions

$$(?=([a]+) - ([a]+) = ([a]+))\backslash 2\backslash 3 - \backslash 2 = \backslash 3$$

### 2. Language Models

- True. Let the length of the text corpus be a, therefore the perplexity of the language model is calculated by

$$PP(\text{test corpus}) = \sqrt[a]{\prod_{i=1}^a \frac{1}{p(x_i | x_{i-1})}}$$

and because each word in the test is seen at least once in training and for all sentences  $p(x_1 \dots x_n) > 0$ , 0 will not appear as a denominator, and the fraction  $1 / 0$  will not happen. Therefore the perplexity will always be less than  $< \infty$ .



b. False.

$$\begin{aligned} \text{PP (test corpus)} \\ &= \sqrt[N]{\prod_{i=1}^N \frac{1}{q(f_i | x_{i-1})}} \\ &= \sqrt[N]{\prod_{i=1}^N \frac{1}{\frac{c(x_{i-1}, f_i)}{c(x_{i-1})}}} \end{aligned}$$

For the PP to be largest,

$$\prod_{i=1}^N \frac{1}{\frac{c(x_{i-1}, f_i)}{c(x_{i-1})}}$$

should approach  $\frac{1}{N}$  such that the PP can have a value of  $N$ ;

therefore the fraction  $\frac{c(x_{i-1}, f_i)}{c(x_{i-1})}$  should have a value of  $\frac{1}{N}$ .

However, the discount  $c^*(x_{i-1}, f_i) = c(x_{i-1}, f_i) - 0.5$  caused the fraction  $\frac{1}{\frac{c(x_{i-1}, f_i) - 0.5}{c(x_{i-1})}}$  to yield a greater value than  $\frac{1}{\frac{c(x_{i-1}, f_i)}{c(x_{i-1})}}$  as shown below:

$$\frac{1}{\frac{c(x_{i-1}, f_i) - 0.5}{c(x_{i-1})}} = \frac{c(x_{i-1})}{c(x_{i-1}, f_i) - 0.5} > \frac{c(x_{i-1})}{c(x_{i-1}, f_i)}$$

therefore the perplexity

$$\sqrt[N]{\prod_{i=1}^N \frac{1}{q(f_i | x_{i-1})}} > N+1.$$

c. True.

$$\begin{aligned}
 & \sqrt[n]{\prod_{i=1}^n \frac{a}{q(x_i | x_{i-1})}} \\
 &= \sqrt[n]{\prod_{i=1}^n \frac{a}{\frac{1}{N+1}}} \\
 &= \sqrt[n]{(N+1)^n} \\
 &= N+1
 \end{aligned}$$