CS4248 Assignment 1: Regexes and Language Models

By A0236430N

This is a sample writeup.pdf file, to illustrate the expected format. You may choose to use this source file but need not to. Just follow the instructions required in the Assignment 1 instructions.

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 the Pokémon Go rule.

Signed, A0236430N

2. **References**.

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

- https://regex101.com/
- Lecture notes

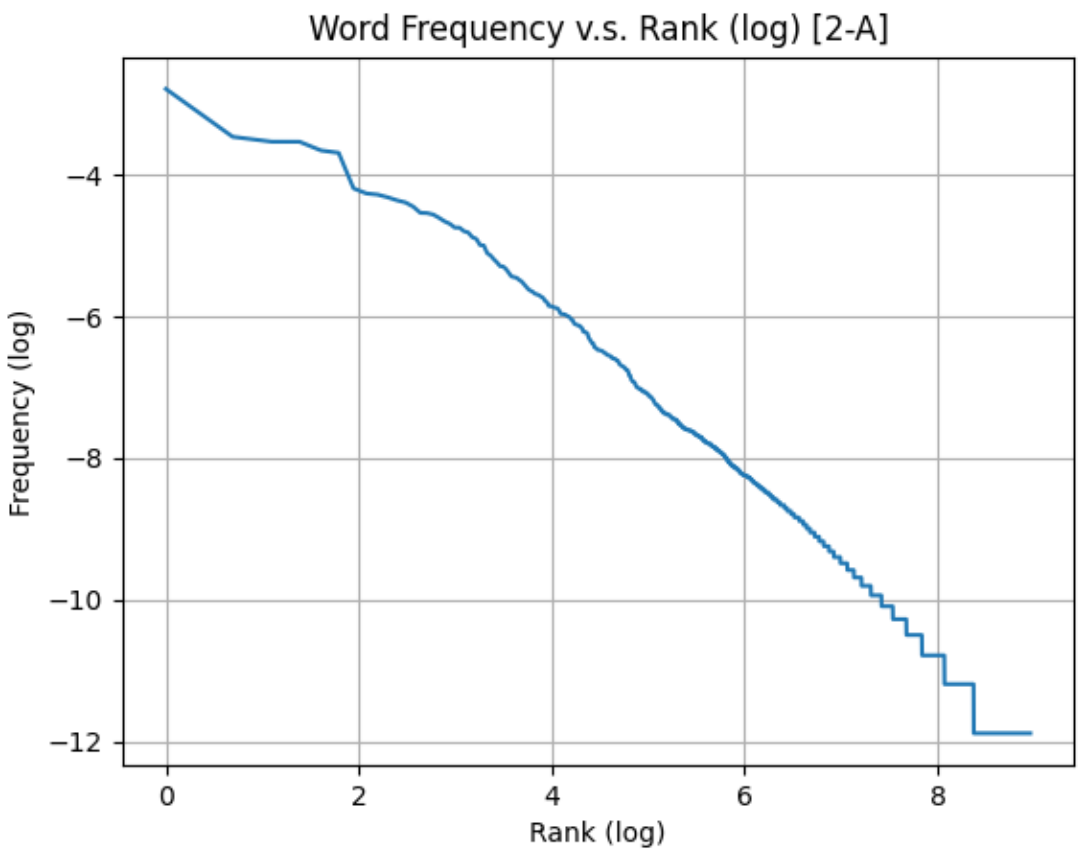
Part 1. Programming

Objective 1 — Regular Expressions

C. Descriptions of relations between R3, R4, and R1 in FSA.
   Answer here:
   $R3 = R1 \cap R2$ and I set $R4 = R1 \cap {\sim}R2$ thus fulfilling the requirement that $R1 = R3 \cup R4$. We also can deduce $R3 \cap R4 = \emptyset$. Therefore, the FSA of R1 is the union of R3's FSA and R4's FSA, and the FSA of R3 and R4 does not have common transitions or states since R3 and R4 have no intersection.

Objective 2 — Tokenization, Zipf’s Law

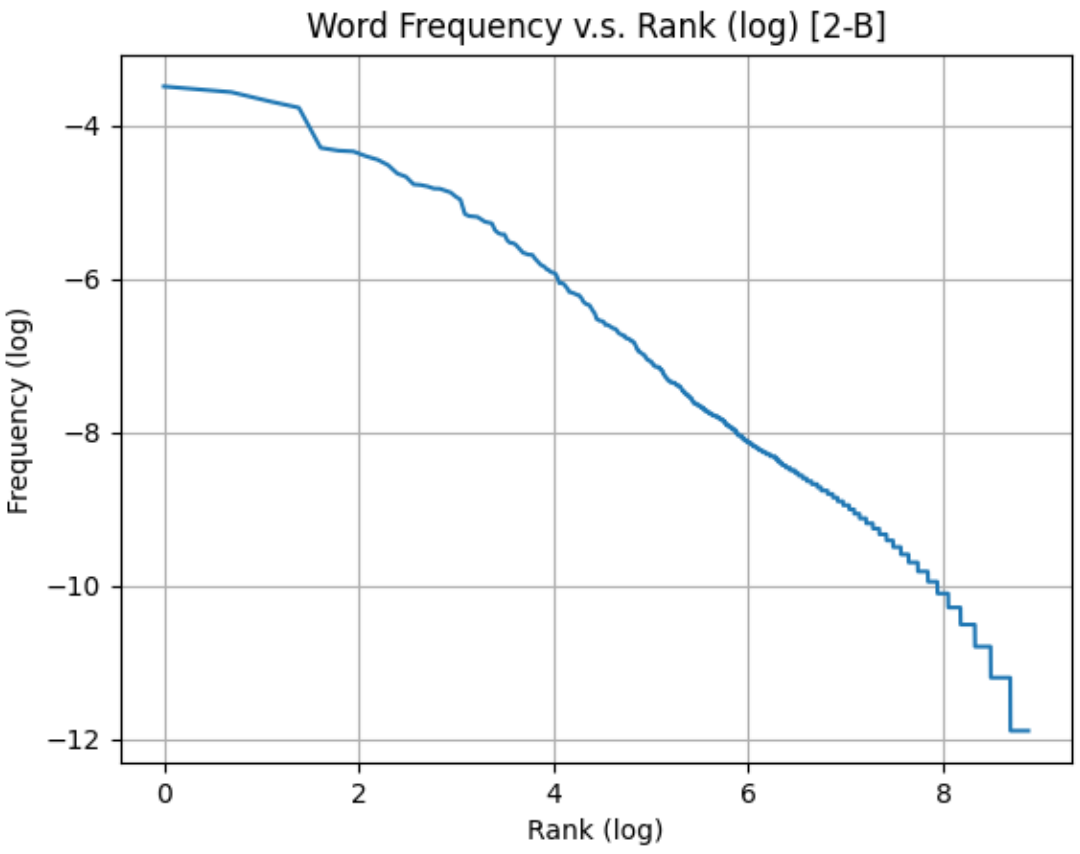
A. Plot here.



Justification here.

Yes, the figure is consistent with Zipf’s law. As we can see, the slope of the curve is approximately -1, which means frequency is approximately inversely proportional to rank. Zipf’s law shows that a few items occur very frequently, while many others occur rarely. Zipf's law is a specific example of a power-law distribution. Many natural phenomena, including language, exhibit power-law distributions, where a small number of items have a very high frequency, while the vast majority have low frequencies. Tokenization deals with language, which follows such distributions. The structure of English language itself contributes to Zipf's law in tokenization. Certain grammatical structures and semantic patterns lead to certain words or phrases occurring more frequently than others.

B. Plot here.

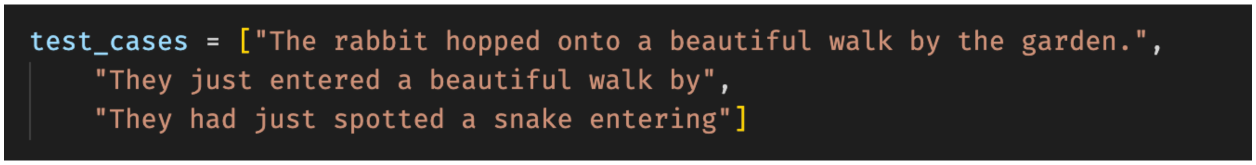


Justification here.

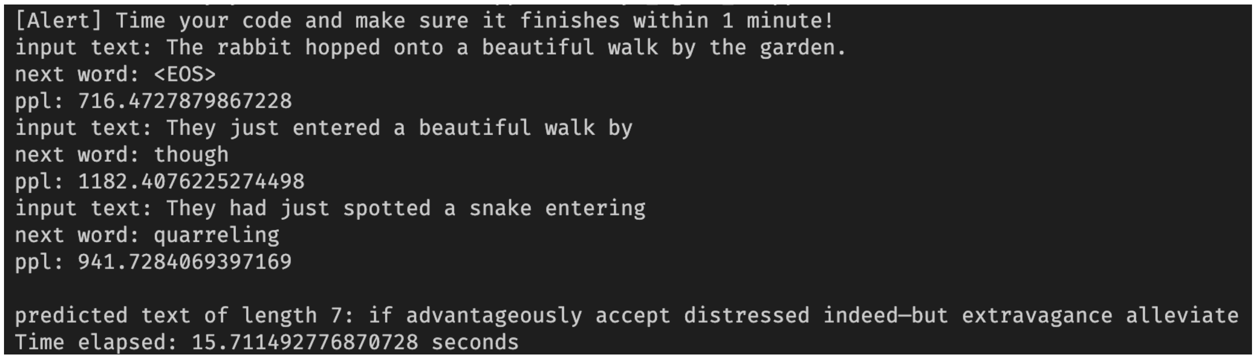
Yes, the result is still consistent with Zipf's law. One explanation is that BPE should be closer and closer to basic tokenization when number of iteration increases. This is because BPE merges tokens that are empirically common, whereas basic tokenization merges tokens based on predefined language rules. Since we set the number of iterations until the number of vocabs equals that of basic tokenization, the result should be quite similar. However, we can see the line is more concave and not exactly as straight as basic tokenization, which may happen because BPE analyses the common tokens and merge them together based on the training data so we can expect higher frequencies of medium rank tokens.

Objective 3 — Language Modeling, Regular Expressions

Here are some of the input-output pairs with respect to the generate_word, generate_text, and get_perplexity calls using the text corpus:

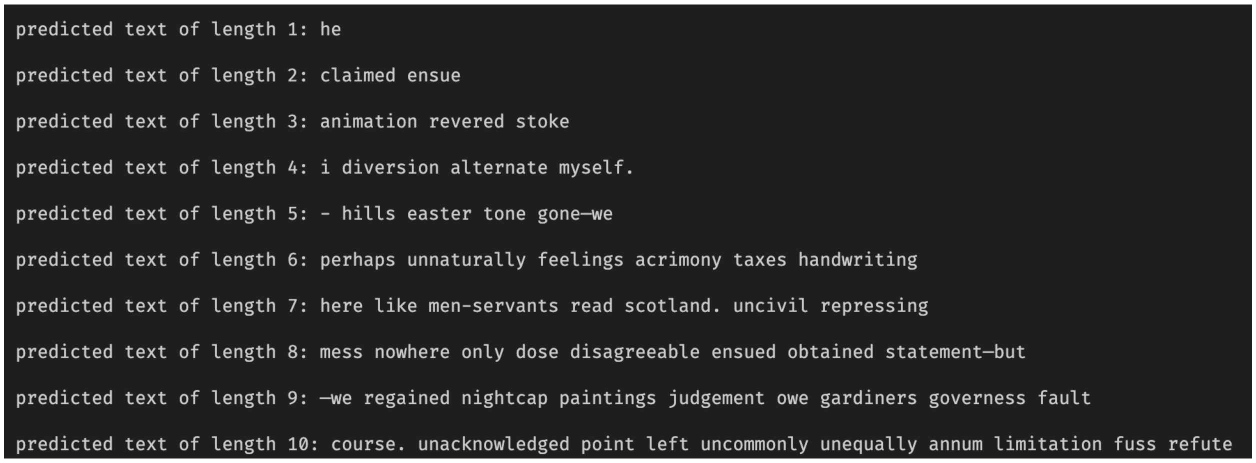


```
test_cases = ["The rabbit hopped onto a beautiful walk by the garden.",
    "They just entered a beautiful walk by",
    "They had just spotted a snake entering"]
```



```
[Alert] Time your code and make sure it finishes within 1 minute!
input text: The rabbit hopped onto a beautiful walk by the garden.
next word: <EOS>
ppl: 716.4727879867228
input text: They just entered a beautiful walk by
next word: though
ppl: 1182.4076225274498
input text: They had just spotted a snake entering
next word: quarreling
ppl: 941.7284069397169

predicted text of length 7: if advantageously accept distressed indeed—but extravagance alleviate
Time elapsed: 15.711492776870728 seconds
```



```
predicted text of length 1: he

predicted text of length 2: claimed ensue

predicted text of length 3: animation revered stoke

predicted text of length 4: i diversion alternate myself.

predicted text of length 5: - hills easter tone gone—we

predicted text of length 6: perhaps unnaturally feelings acrimony taxes handwriting

predicted text of length 7: here like men-servants read scotland. uncivil repressing

predicted text of length 8: mess nowhere only dose disagreeable ensued obtained statement—but

predicted text of length 9: —we regained nightcap paintings judgement owe gardiners governess fault

predicted text of length 10: course. unacknowledged point left uncommonly unequally annum limitation fuss refute
```

...

If you implemented any additional LM variants, you're welcomed to give high level documentation here.

High level documentation:

- I use dictionaries as a LM to store the counts of unigram tokens and dict of dicts to store bigram counts.
- Preprocess the text using nltk sent_tokenize and then append BOS and EOS. Use nltk word_tokenize to get individual word tokens.
- Count the occurrences of each token.
- To generate word, count the probability of all potential tokens (given the previous word if bigram) and use random.choices() to simulate random variable of tokens with assigned weight as their calculated probabilities.
- To generate sentence, generate word from BOS and use the generated words as "previous" words of the next generated word.

- To calculate perplexity, use the perplexity formula given in the lecture with probabilities calculated from the text data (Pride and Prejudice).

Part 2

1. **Language Models**

   a. True.
   For any sentences $P(x_1 \ldots x_n) > 0$.
   We have P(W) > 0 for all test corpus since all words are in V.
   $P(W) > 0 \Rightarrow \frac{1}{P(W)} < \infty \Rightarrow P(W)^{-1/N} < \infty \Rightarrow PP(W) < \infty$ for any test corpus.

   Perplexity of infinity means it is impossible to get the corpus. However, because we assume that the probability of all sentences is > 0 and there will not be a word outside V, the probability for us to get any corpus will be > 0 and hence finite perplexity.

   b. False.
   Proof by counterexample:
   Suppose V consists of ‘a’ and ‘b’ hence N = 2. Counts of bigrams ‘aa’ = 10, ‘ab’ = 10, ‘ba’ = 10, ‘bb’ = 1. With test corpus of ‘bb’, $P('bb') = \frac{1-0.5}{11-0.5}$.
   $$PP('bb') = P('bb')^{-1/2} = 4.58 > N + 1 = 3$$

   We can pick any bigram rare enough as our test corpus which has perplexity more than $N + 1$. Since there is no other constraints for the frequencies of the bigrams, we can simply pick rare bigrams to have high enough perplexity.

   c. True.
   For any test corpus, $PP(W) = (\prod_{i=1}^{n} q(w_i| w_{i-1}))^{-1/n} = 1/(N+1)^{-n/n} = N + 1$ by the assumptions in the problem. Thus, the perplexity of any test corpus is constant in this language model.

   The intuition for this is that given the assumption of q for any bigrams is constant, the probability of the “next” word in a sentence is the same for all words. Since perplexity is normalized for the number of words in the test corpus, constant probabilities for all words in the sentence lead to constant perplexity for any test corpus.