CS4248 Assignment 1:
Regexes and Language Models

By A0155664X

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, A0155664X

2. **References**.

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

- Regex Golf: https://alf.nu/RegexGolf for practicing regular expressions.

Part 1. Programming

Objective 1 — Regular Expressions

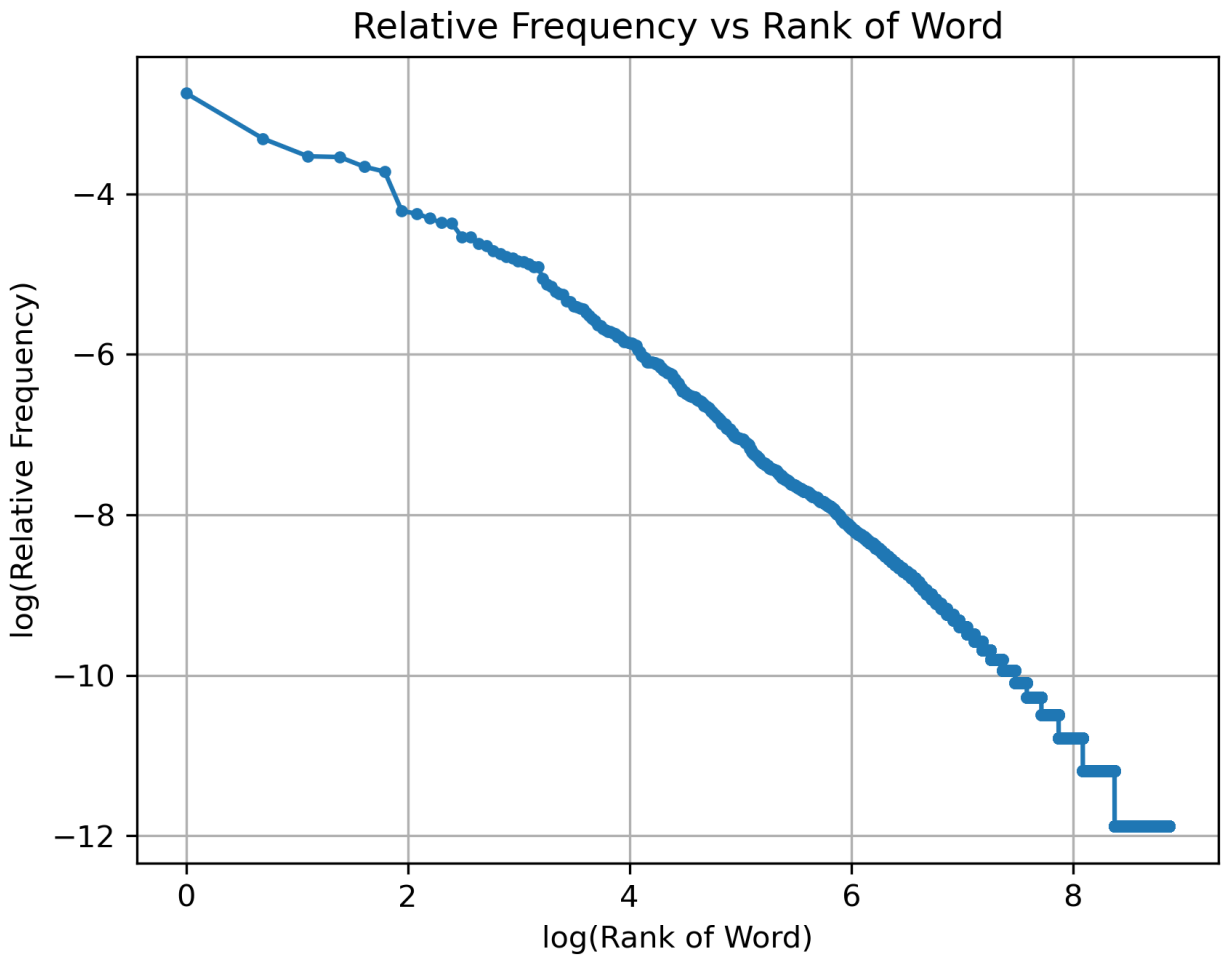
C. Descriptions of relations between R3, R4, and R1 in FSA.
R4: ^\S\S*\1$
R4 = R1 is a valid answer, since R3 is a subset of R1. And hence, R1 = R3 union R4.

The union of R3 and R4 will match all strings except for single character strings, where R3 matches all strings that start and end with the same character. but don't contain substring 'abba', while R4 matches all strings that start and end with the same character, regardless of the 'abba' pattern.

In FSA, R1 is represented by the union of 2 disjoint states (R3 and R4), with a common start state and end state.

Objective 2 — Tokenization, Zipf's Law

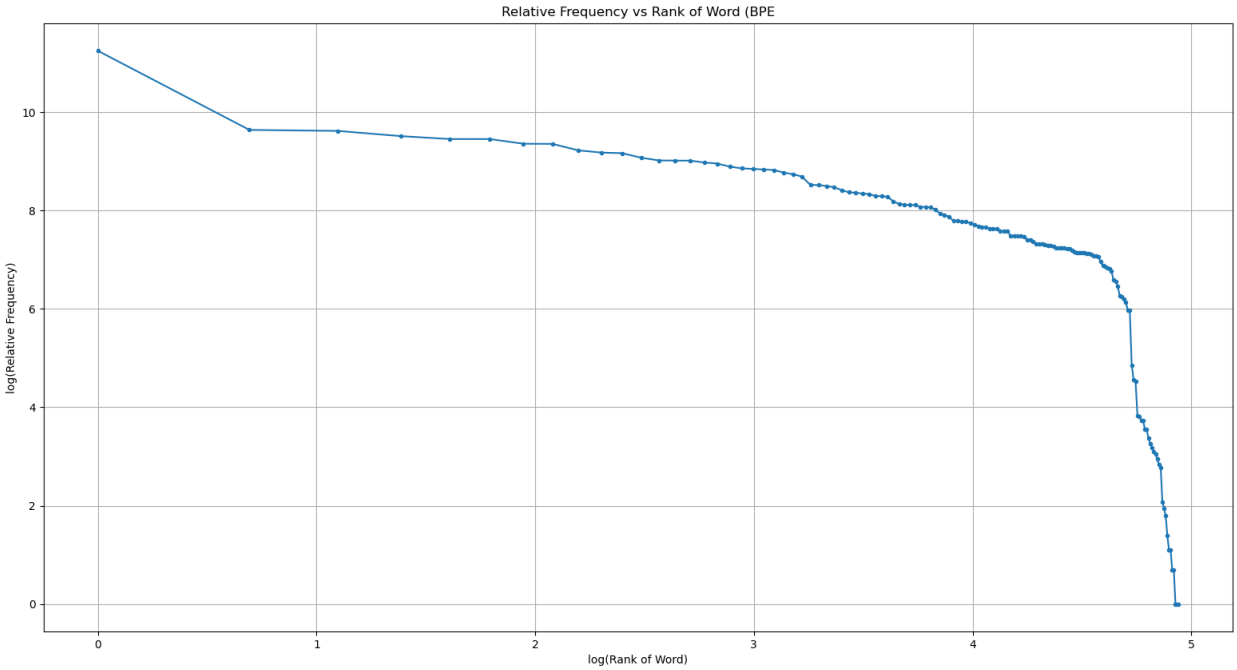
A.



Justification:

Zipf's law in tokenization refers to the observation that the word frequency is inversely proportional to its rank in the frequency table. From the plot for Q2A above, the figure is consistent, as we can see the frequency of the word decreasing as the rank of the word gets larger, reflecting an inverse relationship.

B.



Justification:

Byte-pair encoding is a data compression technique that uses a frequency-based dictionary of subword units to replace frequent word sequences with single codes. It does not necessarily follow Zipf's law. While BPE considers the frequency of subword units, it does not strictly adhere to Zipf's law as it uses a different approach to represent words. Hence, the shape of the graph does not show inverse relationship.

From the graph above, we can see that there is a much more complex relationship than an inverse as shown in the above. Initially, as rank of word increases, the frequency drops slowly. However, when the rank is much larger, the frequency of the words drops quickly, attributing to subword units being found less in our text.

Objective 3 — Language Modeling, Regular Expressions

Here are some of the input-output pairs with respect to the generate_word and get_perplexity calls using the text corpus as in the list of strings test_cases:

```
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"]
```

For the case of add-k where k = 1.0 for n_gram = 2, the following is the output for generate_word and get_perplexity for all 3 cases:

input text: The rabbit hopped onto a beautiful walk by the garden.
next word: opposite
ppl: 5814.460838857911
input text: They just entered a beautiful walk by
next word: learned
ppl: 7854.491084129707
input text: They had just spotted a snake entering
next word: checking
ppl: 11573.59548145296

generate_text was performed for the string ‘Hello, nice to meet you.’ for a length input of 10 and the result was:

predicted text of length 10: highflown impatient liberality eliza entail intrigues satin crossing biting portion

For the case of add-k where k = 0 for n_gram = 2, the following is the output for generate_word and get_perplexity for all 3 cases:

input text: The rabbit hopped onto a beautiful walk by the garden.
next word: trembling
ppl: 3647160855602.477
input text: They just entered a beautiful walk by
next word: such
ppl: 4460970391.803261
input text: They had just spotted a snake entering
next word: the
ppl: 3.3248282343350035e+17

generate_text was performed for the string ‘Hello, nice to meet you.’ for a length input of 7 and the result was:

predicted text of length 7: bent the possible consequence of her particular

For the case of add-k where k = 1.0 for n_gram = 1, the following is the output for next word and perplexity for all 3 cases:

input text: The rabbit hopped onto a beautiful walk by the garden.
next word: ,
ppl: 8269.669721208385
input text: They just entered a beautiful walk by
next word: .
ppl: 2547.2235668085837
input text: They had just spotted a snake entering
next word: troop
ppl: 7398.783330804937

generate_text was performed for the string ‘Hello, nice to meet you.’ for a length input of 8 and the result was:

predicted text of length 8: been miss foresaw to , exceedingly , therefore

For the case of add-k where k = 0 for n_gram = 1, the following is the output for next word and perplexity for all 3 cases:

input text: The rabbit hopped onto a beautiful walk by the garden.
next word: to
ppl: 207239805631.5343
input text: They just entered a beautiful walk by
next word: parted
ppl: 324259.8924394127
input text: They had just spotted a snake entering
next word: sitting
ppl: 16301496221.12131

generate_text was performed for the string ‘Hello, nice to meet you.’ for a length input of 3 and the result was:

predicted text of length 3: removal the .

Part 2

1. **Language Models**

    a. True.
    Explanation:

This is due to the assumption in question that all words in any test corpus are in vocabulary V, which means that the counts of occurrence of any bigram and unigram will never be zero during the calculation of perplexity.

Mathematical proof:
We know that perplexity = PP(W) = P(w1w2…wn)^(-1/N), where N is the number of words in the test corpus. Since P(w1w2...wn) > 0 as all sequences x1..xn are in vocabulary V, we have PP(W) < infinity.

Intuitive interpretation:
Since all words in the test corpus are in the vocabulary V, it will always be possible to form a sentence with words from V, and the probability of the sentence will be > 0. For bigram language model without Laplace smoothing, we have probabilities = count(bigram) / count(unigram).

Since counts of bigram and unigram will be non-zero due to all words being seen in the test corpus for vocab V, we can be sure that perplexity score will be finite.

b. False.

Mathematical proof:
For a bigram model, we have the following for perplexity calculations:
N + 1 comes from the fact that we have vocab size N and +1 is from xn = STOP.
PP(W) = (Πn (i = 1) q(wi|wi-1)^(-1/N)). Since q(wi|wi-1) >= 1/(N+1) as some bigrams might be more probable than others in the vocab, PP(W) >= N+1.

Intuitive interpretation:
Assume the case whereby we assign the probabilities of the bigram to be equal with q(wi|wi-1) = 1/(N+1), we will get a lower bound PP(W) = N+1. In actual fact, within the test corpus, some bigrams might occur more than other bigrams and this assumption might not hold true all the time. Thus, the perplexity score of N+1 represents only a lower bound, and perplexity score could be higher than N+1.

c. True.

Mathematical proof:
Now, we have q(wi|wi-1) = 1/(N+1). Given that PP(W) = (Πn (i = 1) q(wi|wi-1)^(-1/N)), by substitution, we get PP(W) = ((1/(N + 1))^n)^(-1/N) = (N + 1)^(n/N) = N+1 (since n = N). PP(W) = N+1 in this case, and this represents the lower bound PP(W) mentioned in part B.

Intuitive interpretation:
Now we have an additional assumption that for every bigram, $q(w|v) = 1/(N+1)$, which means that the model assigns equal probability to every bigram in the vocab and the model is not confident about its predictions and there is the assumption that the all bigrams have same probability of occurring only once in the entire vocabulary. This represents a case of language model with the least amount of information about the language, as it does not differentiate between the bigrams (same probability for each bigram).