CS4248 Assignment 1: Regexs and Language Models

By A0184415E (Please change this as appropriate)
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 perthe Pokémon Go rule.

Signed, A0184415E

2. **References**.

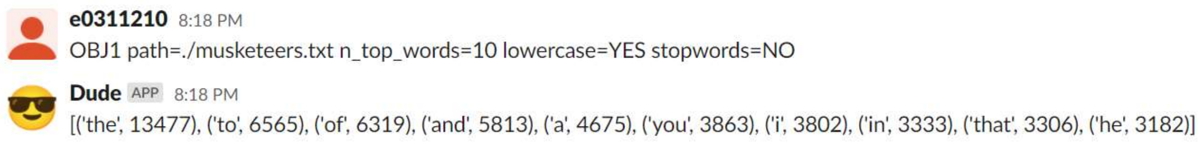
I give credit where credit is due. I acknowledge that I used the following websites or contacts to complete this assignment

- StackOverflow (stackoverflow.com) for general debugging queries
- https://www.regular-expressions.info/ for regex references
- https://regexr.com/ To test regexes
- Anyone who asked a question or provided an answer to a question on the slack channel

Part 1

Objective 1 — Tokenization, Zipf's law

A.



e0311210 8:18 PM

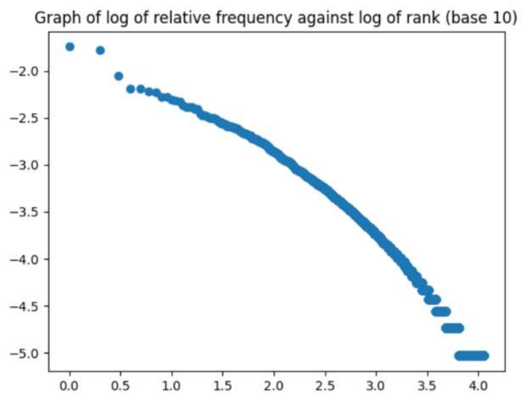
```
OBJ1 path=./musketeers.txt n_top_words=10 lowercase=YES stopwords=NO
```

Dude APP 8:18 PM

```
[('the', 13477), ('to', 6565), ('of', 6319), ('and', 5813), ('a', 4675), ('you', 3863), ('i', 3802), ('in', 3333), ('that', 3306), ('he', 3182)]
```

This is no surprise. The stopwords form the grammatical backbone of the English language, and we did not remove the stopwords in this case. Thus it makes sense that most of the common words would be stopwords.

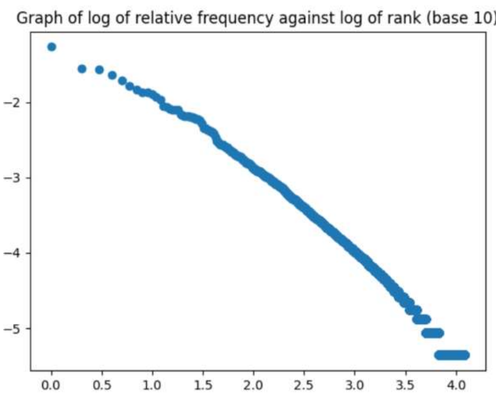
B.



lowercase=NO stopwords=YES

Stopwords are removed, but words are not converted to lowercase. The graph shows a plot that has a nearly-constant gradient, but which is nonetheless obviously not linear. The removal of the stopwords is likely the culprit for this – Zipf's Law applies to large bodies of text that follow the English language, in this case. A large body of text without the stopwords would have poor grammar, and would not follow the English language so well – Zipf's Law thus wouldn't apply as much.

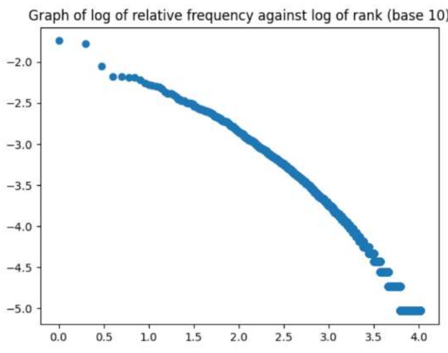
C.



lowercase=NO stopwords=NO

This looks like a straight line as predicted by Zipf's Law. There is no surprise in this, as the stopwords were not removed – the body of text was in the English language with no changes. There is nonetheless a slight downward curve in the line. This curve is less noticeable than in parts B and D. It is likely that the lack of conversion to lowercase is to blame – Zipf's Law applies to individual words, regardless of case.

D.



lowercase=YES stopwords=YES
Stopwords are removed. The graph shows a plot that has a nearly-constant gradient, but which is nonetheless obviously not linear. The removal of the stopwords is likely the culprit for this – Zipf's Law applies to large bodies of text that follow the English language, in this case. A large body of text without the stopwords would have poor grammar, and would not follow the English language so well – Zipf's Law thus wouldn't apply as much. This graph looks very similar to that of part B – this may imply that the effect of changing the case of the words is not very large. It is thus likely that many of the words in the corpus appear at the start of the sentence a proportionally equal number of times.

Objective 2 — How’s the Weather?

We define a set of words that are related to weather. These words include things like “weather”, “rain”, “temperature”. If a user’s input does not include any of these words, we can respond with the default reply, as the input does not even mention weather.

Now we check if the relevant cities are mentioned (Cairo, London, Singapore). Again, if these words are absent from the input, then we can give the default response.

Suppose both a city and a weather word are mentioned. The user could simply be making a statement like “The weather in Singapore is hot”. To check if this is the case, we need to find out if the user is asking a question about the weather in the city. We identify questions by the absence of a closing period or exclamation mark (rather than the presence of a question mark). The rationale for this is that most humans would also interpret "what's the singapore weather like" as a question, even though there's no question mark. If the user input is a question, then we check for keywords like “how” or “what” or “is” – such as “how is the weather in singapore”, “what is the weather like in singapore”, or “is it hot in singapore”. These allow us to identify questions about the weather in a city.

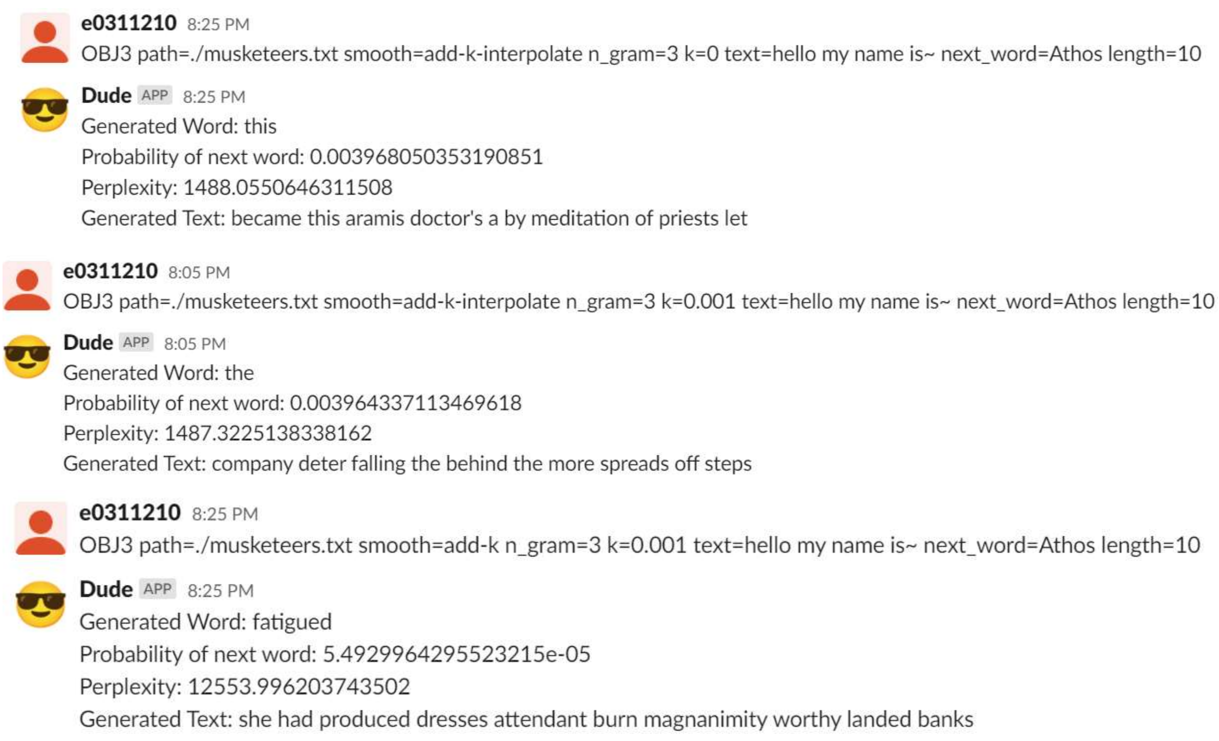
The user need not make his query in the form of a question – it could also be a request, like “tell me the weather in singapore”. To resolve this, besides only checking for questions, we also must check whether request words, like “tell me” or “please update me” are present. These allow us to identify *requests* about the weather in a city.

Now let’s suppose we have a request or question about a city’s weather. We can immediately reply with the city’s weather in most cases. However, sometimes we may not be able to reply with the weather immediately, as the input may contain multiple cities, like “Is the weather in Singapore as hot as Cairo?” In such cases, we look for keywords like “in” (e.g. “weather in Singapore”) or “’s” (e.g. “Singapore’s weather”) to determine the city of interest. We then reply with the relevant weather.

If all of the above do not match the input, then we return the default.

Objective 3 — Language Modeling

Here are some of the captured output conversations with respect to the generate_word, generate_text, an perplexity calls using the text corpus `musketeers.txt` (replace as needed).



```
e0311210 8:25 PM
OBJ3 path=./musketeers.txt smooth=add-k-interpolate n_gram=3 k=0 text=hello my name is~ next_word=Athos length=10
Dude APP 8:25 PM
Generated Word: this
Probability of next word: 0.003968050353190851
Perplexity: 1488.0550646311508
Generated Text: became this aramis doctor's a by meditation of priests let

e0311210 8:05 PM
OBJ3 path=./musketeers.txt smooth=add-k-interpolate n_gram=3 k=0.001 text=hello my name is~ next_word=Athos length=10
Dude APP 8:05 PM
Generated Word: the
Probability of next word: 0.003964337113469618
Perplexity: 1487.3225138338162
Generated Text: company deter falling the behind the more spreads off steps

e0311210 8:25 PM
OBJ3 path=./musketeers.txt smooth=add-k n_gram=3 k=0.001 text=hello my name is~ next_word=Athos length=10
Dude APP 8:25 PM
Generated Word: fatigued
Probability of next word: 5.4929964295523215e-05
Perplexity: 12553.996203743502
Generated Text: she had produced dresses attendant burn magnanimity worthy landed banks
```

In addition to add-k smoothing, I have also implemented the option of no smoothing, by passing in the value of k=0.

Of course the above is trivial.

I have also implemented a form of interpolation. This can be invoked with a smoothing value of "*add-k-interpolate*". This interpolation will take into account the current context, as well as the current context with one and two words removed. From the screenshots above, it is very clear that using the interpolation greatly improves perplexity, with or without the add-k smoothing. It does not, however, produce a significant noticeable difference in subjective sentence quality.

Part 2

1. **Subtraction Regular Expressions**

(\2\3) - (a+) = (a+)

2. **Language Models**

a. True.

1. Suppose not. Let perplexity $= \infty$
2. Then $P(x_1 x_2 ... x_n)^{-1/n} = \infty$
3. Then $P(x_1 x_2 ... x_n) = 0$
4. Then $P(x_2|x_1)P(x_3|x_2)...P(x_n|x_{n-1})) = 0$
5. Then $\exists i \in [1, n] : P(x_i|x_{i-1}) = 0$
6. We know that $P(x_i|x_{i-1})$ is estimated to be $\frac{c(v,w)-\beta}{c(v)-|V|\beta}$
7. $\beta = 0.5$
8. $c(v, w) \in Z$ due to the nature of counting.
9. Thus $(c(v, w) - \beta) \notin Z$
10. Thus $(c(v, w) - \beta) \neq 0$
11. Thus $\forall i \in [2, n] \cap Z, P(x_i|x_{i-1}) \neq 0$
12. There is a contradiction between points 5 and 11.
13. Thus we have proven the statement False by contradiction. QED

b. False.

1. Suppose it is true. $PP(x_1x_2...x_n) \leq N+1$
2. Then $P(x_1x_2...x_n)^{-\frac{1}{n}} \leq N+1$
3. Then $P(x_1x_2...x_n) \geq (N+1)^{-n}$
4. Then $\prod_{i\in[1,n]\cap Z} P(x_i|x_{i-1}) \geq (N+1)^{-n}$
5. $N \geq 0$, so $N+1 > 0$, so $(N+1)^{-n} > 0$
6. Then $\prod_{i\in[1,n]\cap Z} P(x_i|x_{i-1}) > 0$
7. Now consider some w, v such that $c(v,w) = 0$. Suppose that $\forall x \in V, c(x) > 0.5N$, and that $\forall x, y \in V, (x \neq v \wedge y \neq w) \rightarrow c(x,y) \geq 1$
8. Then $c^*(v,w) = -0.5$
9. $P(w|v) = \frac{-0.5}{c(v)-0.5N} < 0$
10. Also, $\forall x, y \in V, (x \neq v \wedge y \neq w) \rightarrow c^*(x,y) > 0$
11. $\forall x, y \in V, (x \neq v \wedge y \neq w) \rightarrow P(x|y) > 0$
12. Then for any test corpus where w immediately follows v, $\prod_{i\in[1,n]\cap Z} P(x_i|x_{i-1}) < 0$
13. This is a contradiction with point 5. Therefore, we have proven that the statement is False. The perplexity may not be at most $N+1$. QED

c. True

1. $\forall w, x, y, z \in V, q(w|x) = q(y|z)$
2. As such, let w be any arbitrary element of V. This proof will extend to any other elements by symmetry.
3. Let $t_0 = ""$, i.e. $t_0 = STOP$
4. $\forall i \in Z^+$, let $t_i = w + t_{i-1}$
5. Let $f(x)$ be true if and only if $P(t_x) = (\frac{1}{N+1})^{x+1}$
6. (a) Let $START$ denote the start of a sentence (not the starting character, but the start, i.e. the 0th character)
   (b) $P(t_0) = P(STOP) = P(STOP|START) = \frac{1}{N+1} = \frac{1}{N+1}^{0+1}$
   (c) So $f(0) = TRUE$
7. (a) Suppose we have $f(i) = TRUE$.
   (b) Consider t_{i+1}
   (c) $P(t_{i+1}) = P(t_i)(\frac{1}{N+1}) = (\frac{1}{N+1})^{i+2}$
   (d) So $f(i+1) = TRUE$
   (e) So $f(i) \rightarrow f(i+1)$
8. So $f(i) = TRUE \ \forall i \in Z \cap [0, \inf)$
9. So $\forall i \in Z \cap [0, \inf), P(t_i) = (\frac{1}{N+1})^{i+1}$
10. $PP(t_i) = P(t_i)^{-\frac{1}{i+1}}$
11. So $\forall i \in Z \cap [0, \inf), PP(t_i) = N + 1$
12. So for any test corpus, the perplexity under the language model is $N + 1$. QED.