Natural Language Processing
Assignment 1

Matthijs Prinsen (s4003365)
Rob Sligter (s5564875)
Marinus van den Ende (s5460484)
Group 34

February 22, 2025

Contents

Notes

Bigram Probability

The probability of word w_2 appearing after word w_1 is given by:

$$P(w_2|w_1) = \frac{count(w_2, w_1)}{count(w_1)} \tag{1}$$

For example, given the sentence: `"I like dogs and I like cats"`

$$P(\text{like}|I) = \frac{2}{2} = 1$$
$$P(\text{cats}|\text{like}) = \frac{1}{2} = 0.5$$

1 Problem 1

1.1 Bigram Probabilities from Corpus

The corpus consists of:

- $\langle s \rangle$ `I like blueberries` $\langle e \rangle$
- $\langle s \rangle$ `You like all types of berries` $\langle e \rangle$
- $\langle s \rangle$ `I hate bitter fruits` $\langle e \rangle$
- $\langle s \rangle$ `You like all sweet fruits` $\langle e \rangle$
- $\langle s \rangle$ `I like chocolate covered raspberries` $\langle e \rangle$

Bigram	Probability
$P(I\|\langle s\rangle)$	$\frac{3}{5} = 0.60$
$P(You\|\langle s\rangle)$	$\frac{2}{5} = 0.40$
$P(like\|I)$	$\frac{2}{3} = 0.67$
$P(hate\|I)$	$\frac{1}{3} = 0.33$
$P(blueberries\|like)$	$\frac{1}{4} = 0.25$
$P(all\|like)$	$\frac{2}{4} = 0.50$
$P(like\|you)$	$\frac{2}{2} = 1.0$
$P(types\|all)$	$\frac{1}{2} = 0.50$
$P(of\|types)$	$\frac{1}{1} = 1.0$
$P(berries\|of)$	$\frac{1}{1} = 1.0$
$P(bitter\|hate)$	$\frac{1}{1} = 1.0$
$P(fruits\|bitter)$	$\frac{1}{1} = 1.0$
$P(sweet\|all)$	$\frac{1}{1} = 1.0$
$P(fruits\|sweet)$	$\frac{1}{1} = 1.0$
$P(chocolate\|like)$	$\frac{1}{4} = 0.25$
$P(covered\|chocolate)$	$\frac{1}{1} = 1.0$
$P(raspberries\|covered)$	$\frac{1}{1} = 1.0$
$P(\langle e\rangle\|blueberries)$	$\frac{1}{1} = 1.0$
$P(\langle e\rangle\|fruits)$	$\frac{2}{2} = 1.0$
$P(\langle e\rangle\|raspberries)$	$\frac{1}{1} = 1.0$
$P(\langle e\rangle\|berries)$	$\frac{1}{1} = 1.0$

Table 1: Complete Bigram Probabilities

1.2 Sentence Probability Calculation

To find the probability of these sentences, we look at the **joint probability** of the words. For this we use the **chain rule**:

$$P(S) = P(A, B, C, D, E, \ldots) = P(A) \cdot P(B|A) \cdot P(C|A, B) \cdot P(D|A, B, C) \ldots \tag{2}$$

Generally we would have the **bigram model**:

$$P(S) = P(w_1|w_0) \cdot P(w_2|w_1) \cdot \ldots \cdot P(w_n|w_{n-1}) \tag{3}$$

1.2.1 $S_1 = \langle s\rangle$ `You like all types of berries` $\langle e\rangle$

For the sentence $\langle s\rangle$ `You like all types of berries` $\langle e\rangle$. The probability of the sentence **should not** be $\frac{1}{5}$ simply because it is one of five sentences in the corpus. This is because **we count words, not sentences**. However, it is possible that the joint probability of the sentence happens to be $\frac{1}{5}$ by chance, so let's determine if that is the case.

$$\begin{aligned} P(S_1) &= P(\langle s\rangle, \text{You, like, all, types, of, berries}, \langle e\rangle) \\ &= P(\langle e\rangle|\langle s\rangle, \text{You, like, all, types, of, berries}) \times P(\text{berries}|\langle s\rangle, \text{You, like, all, types, of}) \\ &\quad \times P(\text{of}|\langle s\rangle, \text{You, like, all, types}) \times P(\text{types}|\langle s\rangle, \text{You, like, all}) \\ &\quad \times P(\text{all}|\langle s\rangle, \text{You, like}) \times P(\text{like}|\langle s\rangle, \text{You}) \times P(\text{You}|\langle s\rangle) \times P(\langle s\rangle) \end{aligned}$$

When applying bigram probabilities to approximate the probability of the sentence using the **Markov assumption**, we can ignore the entire history of the new words. That leads us to the following:

$$P(S_1) = P(\text{You} \mid \langle s \rangle) \times P(\text{like} \mid \text{You}) \times P(\text{all} \mid \text{like}) \times P(\text{types} \mid \text{all}) \\ \times P(\text{of} \mid \text{types}) \times P(\text{berries} \mid \text{of}) \times P(\langle e \rangle \mid \text{berries})$$

$$P(S_1) = 0.40 \times 1 \times 0.50 \times 0.50 \times 1 \times 1 \times 1 = 0.1$$

Thus, the sentence has a **probability of 0.1**.

1.2.2 $S_2 = \langle s \rangle$ `You hate bitter fruits` $\langle e \rangle$

Going through the same steps as for the previous sentence, we get the following:

$$P(S_2) = P(\text{You} \mid \langle s \rangle) \times P(\text{hate} \mid \text{You}) \times P(\text{bitter} \mid \text{hate}) \times P(\text{fruits} \mid \text{bitter}) \times P(\langle e \rangle \mid \text{fruits}) \\ = 0.40 \times 0 \times 1 \times 1 \times 1 = 0$$

As the probability for $P(\text{hate} \mid \text{You})$ is 0, the joint probability of the sentence is also 0.

1.3 Trigram Probabilities

We can calculate the **conditional probability** of a **trigram** in a similar way to what we did for the bigrams:

$$P(w_3 \mid w_1, w_2) = \frac{\text{count}(w_1, w_2, w_3)}{\text{count}(w_1, w_2)} \tag{4}$$

As we don't have the true counts, we can't calculate them directly.
Conversion from bigrams to trigrams can't be done without approximations. Using an approximation like:

$$P(w_3 \mid w_1, w_2) \approx P(w_3 \mid w_2) \tag{5}$$

but this is not in the spirit of the question.

1.4 Perplexity Calculation

Perplexity is calculated as such:

$$\text{Perplexity}(S) = P(w_1, \ldots, w_n)^{-1/n} \tag{6}$$

1.4.1 $S_3 = \langle s \rangle$ `I like all sweet fruits` $\langle e \rangle$

To find the **Perplexity** of S_3, we first use our **bigram model** (3) equation as an approximation:

$$P(S_3) = P(\text{I} \mid \langle s \rangle) \times P(\text{like} \mid \text{I}) \times P(\text{all} \mid \text{like}) \times P(\text{sweet} \mid \text{all}) \times P(\text{fruits} \mid \text{sweet}) \times P(\langle e \rangle \mid \text{fruits}) \tag{7}$$

$$= 0.60 \times 0.67 \times 0.50 \times 1 \times 1 \times 1 = 0.201$$

With this approximation, we can calculate the perplexity with an $n = 6$. This includes the end sentence symbol $\langle e \rangle$:

$$\text{Perplexity}(S_3) = 0.201^{-1/6} = 1.30657393585 \tag{8}$$

Rounded to two decimal places:

$$\text{Perplexity}(S_3) = 1.31 \tag{9}$$

1.4.2 $S_4 = \langle s \rangle$ `You like raspberries` $\langle e \rangle$

Again, to calculate the **Perplexity** of S_4, using our **bigram model** (3) approximation:

$$P(S_4) = P(\text{You} \mid \langle s \rangle) \times P(\text{like} \mid \text{You}) \times P(\text{raspberries} \mid \text{like}) \times P(\langle e \rangle \mid \text{raspberries}) \tag{10}$$

$$= 0.40 \times 1 \times 0 \times 1 = 0$$

We have a probability of 0, which does not provide useful information. To address this, we use **Backoff and Interpolation Methods**, replacing our probability of 0 with an approximation.

Using the **unigram probability** instead of the bigram probability:

$$P(\text{raspberries}) = \frac{\text{count}(\text{raspberries})}{N} \tag{11}$$

$$\text{count}(\text{raspberries}) = 2, \quad N = 28 \tag{12}$$

$$P(\text{raspberries}) = \frac{2}{28} = 0.071 \tag{13}$$

With this, we calculate the **perplexity**:

$$\text{Perplexity}(S_4) = 0.071^{-1/4} = 1.93724885474 \tag{14}$$

Rounded to two decimal places:

$$\text{Perplexity}(S_4) = 1.94 \tag{15}$$

2 Problem 2

2.1 Tokenization and Top Word Frequencies

The differences between the tokenizers are minor. The basic tokenizer simply splits at all the spaces, treating punctuation like commas as part of words. In contrast, the nltkTokenize function treats punctuation marks like commas and periods as separate tokens, which leads to slight differences in word frequencies.

```
Most common words nltk: [('the', 24657), (',', 22913),
('.', 22440), ('and', 10629), ('of', 10150), ('to', 9789),
('‘‘', 9682), ('a', 8989), ('in', 6844), ('he', 6170)]

Most common words basic: [('the', 24657), (',', 22913),
('.', 22238), ('and', 10629), ('of', 10150), ('to', 9789),
('a', 8982), ('in', 6844), ('he', 5957), ('was', 5149)]
```

2.2 Word Frequency Plot

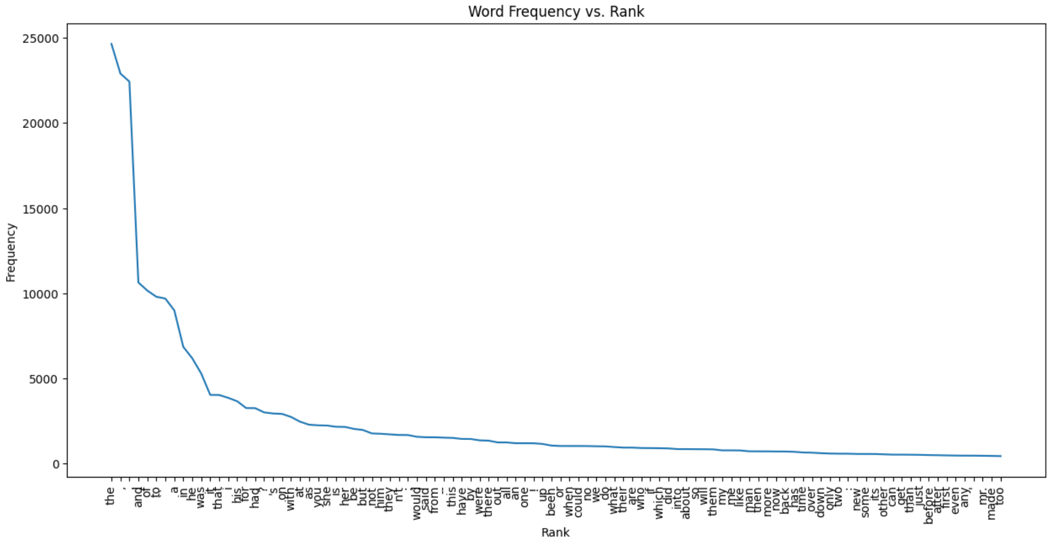

Figure 1: Plot Showing the top 100 most common words in train_corpus_nltk.

The frequency of a word is roughly **inversely proportional** to its rank:

$$f(r) \approx \frac{1}{r}$$

This means the most common word appears the most, the second most common word appears $\frac{1}{2}$ as often, the third appears $\frac{1}{3}$ as often, and so on.

2.3 Perplexity of the Bigram Model

If we train the model to examine the perplexities of the training and test sets, we obtain the following values:

```
Train perplexity: 59.399107, Val Perplexity: inf
```

The training perplexity is within a reasonable range. However, the validation perplexity is reported as infinite, which is incorrect. We get this issue due to a division error when evaluating the model. The probability of a word given the previous word is computed as:

$$P(w_i \mid w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

If $\text{count}(w_{i-1}) = 0$, this results in a division by zero, leading to an infinite perplexity. To address this issue, we apply smoothing techniques, which will be discussed in the next question.

2.4 Additive Smoothing

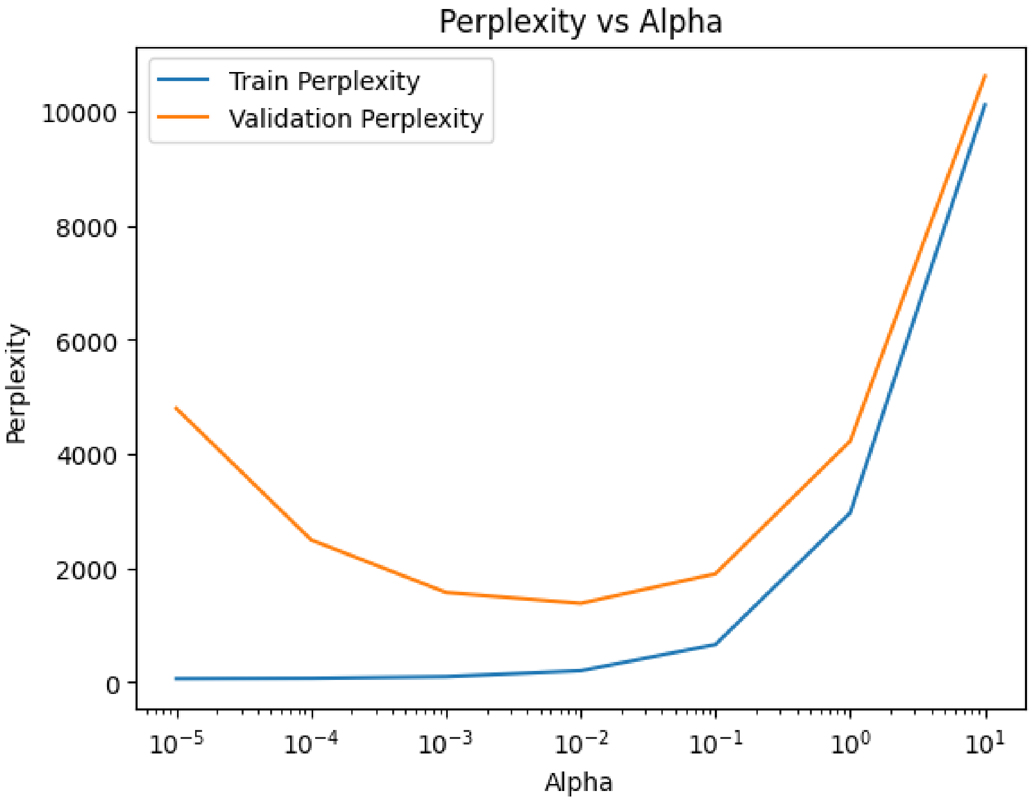

Figure 2: Plot Showing the Perplexity values compared to alpha values

After implementing Laplace (add-α) smoothing, the validation perplexity is no longer infinite. From the plot, we can see that the lowest validation perplexity occurs at $\alpha = 10^{-2}$. As α increases, the perplexity initially decreases but then starts to rise. This happens because very small values of α prevent zero probabilities for unseen bigrams, while larger values of α cause the model to over-smooth, leading to higher perplexity. $\alpha = 10^{-2}$ seems to be the best value for minimizing perplexity.