Gal Cesana 318510633
Shahar Benny 211766670

Natural Language Processing – Exercise 1

Theoretical part

Q1 a.

Given a bigram model where $\forall w \quad p(Stop|w) > 0$, the complementary event where we do not get a stop token after one word is less than 1:

$$\text{i.e.} \qquad \forall w \quad 1 - p(Stop|w) < 1$$

From the bigram model definition, the probability for a sentence $w_1, w_2, \dots, w_n$:

$$p(w_1, w_2, \dots, w_n) = p(w_1) \prod_{i=2}^{n} p(w_i|w_{i-1})$$

So, the probability of never getting a stop token is:

$$p(w_1) \prod_{i=2}^{\infty} 1 - p(Stop|w_{i-1})$$

But we know that the probability of not getting a Stop token after any word in less than 1. So, we get:

$$p(infinite\ sentence) = \ p(w_1) \prod_{i=2}^{n} 1 - p(Stop|w_{i-1}) \xrightarrow[n\to\infty]{} 0$$

$$hence, \qquad p(finite\ sentence) = 1$$

Which means that the sum of probabilities for all finite sentences is 1.

Q1 b.

The property from (a) is not true for non-Markovian models.

We'll look on the case of a non-Markovian model, where the probability of 'Stop' token is determined by all previous words:

$$\forall n \in \mathbb{N}, n > 2 \quad p('Stop'|w_1, w_2, \dots, w_n) = \frac{1}{n^2}$$

Define the sum probabilities of any word from the corpus that is different than 'Stop' to be:

$$\sum_{w \neq Stop} 1 - \frac{1}{n^2} \qquad such\ that\ \forall w \ \ p(w) > 0$$

We get that the probability of never getting a 'Stop' token is:

$$p(infinite\ sentence) = \prod_{i=2}^{n\to\infty} p(w_i \neq Stop | w_1, w_2, \dots, w_{i-1}) = \prod_{i=2}^{n\to\infty} (1 - \frac{1}{i^2}) = \frac{1}{2}$$

So, we got that the probability of getting an infinite sentence is greater than 0.

$Q2$ a.

In a unigram language model, the probability of a sentence $w_1, w_2, \dots, w_n$ is calculated as the product of the probabilities of the individual words, assuming that words occur independently of one another:

$$P(w_1, w_2, \dots, w_n) = \prod_{i=1}^{n} P(w_i)$$

$$where\ P(w_i) = \frac{count(w_i)}{total\ number\ of\ words\ in\ the\ corpus}$$

For the sentence "He went where there where more opportunities":

1. First instance of "where": The corrector will predict "where" if:

   $$P(\text{"where"}) > P(\text{"were"})$$

   Which is true if the word "where" appears more times than "were" in the corpus.

2. First instance of "where": The corrector will predict "were" if:

   $$P(\text{"were"}) > P(\text{"}where\text{"})$$

   Which is true if the word appeared more times than "where" in the corpus.

3. Both instances: The unigram corrector will not be able to be correct on both instances at the same time.
   A unigram model has significant limitations because it ignores context. It relies only on the overall frequency of the words in the corpus, which in our test case makes it not possible to be correct on both instances.

Gal Cesana 318510633
Shahar Benny 211766670

$Q2$ b.

In a bigram language model, the probability of a sentence $w_1, w_2, \dots, w_n$ is calculated using the conditional probability of each word given the previous word:

$$P(w_1, w_2, \dots, w_n) = P(w_1) \prod_{i=2}^{n} P(w_i | w_{i-1})$$

$$where: \quad P(w_i | w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

For the sentence "He went where there where more opportunities":

1. First instance of "where": The corrector will predict "where" for the first instance if:
$$P('where' | 'went') > P('were' | 'went')$$
i.e. in the corpus, the word 'where' appears more times than the word 'were' right after the word 'went'.
2. Second instance of "where": The corrector will predict "were" for the second instance if:
$$P('were' | 'there') > P('where' | 'there')$$
i.e. in the corpus, the word 'were' appears more times than the word 'where' right after the word 'there'.
3. Both instances: The bigram corrector will predict correctly regarding both instances if both conditions from cases 1 and 2 will be held.

This bigram model is better than the unigram model in (a) because the bigram model can capture the dependency between consecutive words, enabling it to make predictions based on context. For instance, "where there" is more likely than "were there", while "there were" is more likely than "there where". This context sensitivity allows the corrector to better disambiguate between "were" and "where".

Zero-probability sentences:

A sentence in this model might get a zero probability if any bigram $P(w_i | w_{i-1})$ is zero, which occurs if the pair (w_i, w_{i-1}) does not appear in the corpus.

A zero-probability sentence in the bigram model is problematic because it prevents the model from ranking or comparing candidate sentences, overfits to seen data by deeming unseen patterns impossible, and undermines generalization to new contexts.

Gal Cesana 318510633
Shahar Benny 211766670

$Q3$ a.

Notice that:

$$(1) \quad N = \sum_{c=1}^{c_{max}} c \cdot N_c$$

Given Good-Turing estimate of a frequency:

$$The\ frequency\ of\ a\ word\ that\ appears\ c\ times\ in\ the\ corpus = \frac{(c+1)N_{c+1}}{N_c \cdot N}$$

To sum the frequencies of all words in the training corpus we need to calculate the following expression:

$$\sum_{c=1}^{c_{max}} \frac{(c+1)N_{c+1}}{N_c \cdot N} \cdot N_c$$

Because if N_c is the number of words that appears c times in the training corpus times the frequency given by the Good-Turing estimation.

We get:

$$\sum_{c=1}^{c_{max}} \frac{(c+1)N_{c+1}}{N_c \cdot N} \cdot N_c \underset{(1)}{\Rightarrow} \frac{1}{N} \sum_{c=1}^{c_{max}} (c+1)N_{c+1} = \sum_{i=1}^{c_{max}} \frac{1}{i \cdot N_i} \sum_{c=1}^{c_{max}} (c+1)N_{c+1} = \frac{N - N_1}{N}$$

$$= 1 - \frac{N_1}{N} = \mathbf{1} - \boldsymbol{p_{unseen}}$$

$Q3$ b.

The Add-One frequency estimation for a word that appears c times in the training corpus:

$$\frac{c+1}{N + c_{max}}$$

The MLE for a word's frequency is:

$$MLE = \frac{c}{N}$$

Case 1 – MLE > Add-One:

$$\frac{c}{N} > \frac{c+1}{N+c_{max}} \Rightarrow \quad c \cdot N + c \cdot c_{max} > c \cdot N + N \qquad because\ N, c_{max} \geq 0$$

$$\Rightarrow c \cdot c_{max} > N \Rightarrow \quad \boldsymbol{c > \frac{N}{c_{max}}}$$

Case 2 – Add-One > MLE:

$$\Rightarrow \boldsymbol{c < \frac{N}{c_{max}}}$$

Hence, the threshold (μ) is:

$$\boldsymbol{\mu = \frac{N}{c_{max}}}$$

All words with frequency greater than μ hold Add-One > MLE and all words with frequency smaller than μ hold that MLE > Add-One.

$Q3$ c.

In Good-Turing smoothing, the estimate depends on the observed counts c, N_c and N_{c+1}

$$p_c = \frac{(c+1)N_{c+1}}{N_c \cdot N}$$

Comparing Good-Turing smoothing with MLE, the relationship between the two estimates depends on the relative values of N_c and N_{c+1} which can vary irregularly.

For instance:

1. If $N_{c+1} > N_c$, the Good-Turing estimate may increase more rapidly than the MLE, leading to $p_c > p_{MLE}$.
2. If $N_{c+1} < N_c$, the Good-Turing estimate may decrease faster than the MLE, leading to $p_c < p_{MLE}$.

This variability means there is no consistent threshold μ such that p_c is always higher than p_{MLE} for $c < \mu$ and always lower for $c > \mu$. The irregularity of N_c and N_{c+1} violates the monotonicity required for a consistent threshold.

Gal Cesana 318510633
Shahar Benny 211766670

$Q4$ a.

The equation for a trigram language model is:

$$p(w_1, w_2, \dots, w_n) = p(w_1) \cdot p(w_2|w_1) \prod_{i=3}^{n} p(w_i|w_{i-1}, w_{i-2})$$

Where $w_1, w_2, \dots, w_n$ are the words in the sentence.

The model assumes that the probability of a word depends only on the previous two words, mostly ignoring any longer-range dependencies or any other dependencies.

$Q4$ b.

English example:

- Sentence: "The dog barks".
- The subject "dog" is singular, and the trigram model can predict the singular verb form "barks" based on the previous two words "The dog".

Hebrew example:

- המשפט: "הכלבים שלי נובחים".
- בעברית, הפועל "נובחים" מתאים לשם העצם הזכרי בצורת רבים "הכלבים" והיחס "שלי" ולכן על סמך 2 המילים האחרונות "הכלבים שלי" המודל יכול לחזות את הפועל "נובחים".

$Q4$ c.

English example:

- Sentence: "The dogs that live on the farm bark loudly."
- The verb "bark" agrees with the plural subject "dogs", but the subject and verb are separated by the clause "that live on the farm", a trigram model would struggle because the dependency spans more than two words.
- An **8-gram** would have been sufficient to ensure the subject and verb are both included in the model's context.

Hebrew example:

- המשפט: "הבחורה שתמיד עוזרת לכל הסטודנטים יודעת"
- באופן דומה לדוגמה באנגלית, מודל שמתחשב בשתי המילים האחרונות לא יוכל לקחת בחשבון שהמילה "יודעת" היא בנקבה ומדברת על "הבחורה".
- מודל מרקובי מסדר 6 (n=6) יכול היה להיות מספיק על מנת להבין את ההקשר במשפט הנוכחי מכיוון שהוא כולל בתוכו את המילה "הבחורה".

Gal Cesana 318510633
Shahar Benny 211766670

$Q5$

זוג:

"כשאני מדבר איתה, היא לא יודע כמה אני מתרגשת."

ניתן לראות כי המשפט לא תקין בשל מעבר בדיבור בין זכר לנקבה אך כל זוג מילים במשפט הינו תקין ("כשאני מדבר", "מדבר איתה", "איתה, היא", "היא לא", "לא יודע", "יודע כמה", "כמה אני", "אני מתרגשת.").

שלשה:

"הנסיך הקטן מפלוגה ב', לא יראו עוד כבשה שאוכלת פרח."

במשפט הנ"ל כל שלשת מילים תקינה אבל המשפט כולו אינו תקין, גם בשל אי התאמה בין "הנסיך" לבין "יראו".

רביעייה:

"כשהייתי רואה אותה באוניברסיטה תגיד לו שהיא תפוסה."

במשפט כל רביעייה תקינה אבל כלל המשפט אינו תקין דקדוקית.

במהלך ביצוע המטלה ניתן לשים לב כי ככל ש-n גדל נעשה יותר קשה לחשוב על משפט שאינו תקין דקדוקית אך כל n-יה בו תקין דקדוקית.

זה יכול להצביע על כך שכבל שנשתמש במודל מרקובי מסדר גבוה יותר כך נשיג מודל מרקובי מדויק יותר מבחינה דקדוקית אך הדבר דורש נתונים רבים יותר.

Gal Cesana 318510633
Shahar Benny 211766670

Practical part

ex1.py file with the complete code is submitted.

Results:

Q2:

Using the bigram model, the following sentence with the most probable word predicted by the model: "I have a house in ..." is '**the**'.

Q3 + Q4:

Sentence 1 " Brad Pitt was born in Oklahoma":

- (Bigram Model) probability (log) = -inf
- (Interpolated Model) probability (log) = -36.260885436340715

Sentence 2 " Brad Pitt was born in Oklahoma":

- (Bigram Model) probability (log): -30.200694958953708
- (Interpolated Model) probability (log): -31.388261638270976

Perplexity for both sentences (Bigram Model): inf

Perplexity for both sentences (Interpolated Model): 280.7399856199719