DEPARTMENT OF CSE

23CSX502 - Natural Language Processing

# IFET COLLEGE OF ENGINEERING
# DEPARTMENT OF CSE
# 23CSX502 - Natural Language Processing
# Answers
## UNIT II WORD LEVEL ANALYSIS

## PART A

1. Total sentences = 3
Total tokens = 15
Vocabulary $V = 7$ {<s>, I, like, love, NLP, ML, </s>}

a) Counting Words (Frequency Counts)
Unigram Counts

| Word | Count |
|---|---|
| <s> | 3 |
| I | 3 |
| like | 2 |
| love | 1 |
| NLP | 2 |
| ML | 1 |
| </s> | 3 |

b) Bigram Counts
From sentences

1. <s> I, I like, like NLP, NLP </s>
2. <s> I, I like, like ML, ML </s>
3. <s> I, I love, love NLP, NLP </s>

| Bigram | Count |
|---|---|
| <s> I | 3 |
| I like | 2 |
| I love | 1 |
| like NLP | 1 |
| like ML | 1 |
| love NLP | 1 |
| NLP </s> | 2 |
| ML </s> | 1 |

Trigram Counts

| Trigram | Count |
|---|---|
| <s> I like | 2 |
| <s> I love | 1 |
| I like NLP | 1 |
| I like ML | 1 |
| I love NLP | 1 |
| like NLP </s>` | 1 |

| | |
|---|---|
| like ML </s>` | 1 |
| love NLP </s>` | 1 |

b) Unsmoothed Probabilities (MLE)
Unigram Probability
Formula

$$P(w) = \frac{Count(w)}{15}$$

| Word | Probability |
|---|---|
| <s> | 3/15 = 0.20 |
| I | 3/15 = 0.20 |
| like | 2/15 = 0.13 |
| love | 1/15 = 0.07 |
| NLP | 2/15 = 0.13 |
| ML | 1/15 = 0.07 |
| </s> | 3/15 = 0.20 |

Bigram Probability
Formula

$$P(w_2 \mid w_1) = \frac{Count(w_1 w_2)}{Count(w_1)}$$

| Bigram | Calculation | Probability |
|---|---|---|
| <s> I | 3/3 | 1.00 |
| I like | 2/3 | 0.67 |
| I love | 1/3 | 0.33 |
| like NLP | 1/2 | 0.50 |
| like ML | 1/2 | 0.50 |
| love NLP | 1/1 | 1.00 |
| NLP </s> | 2/2 | 1.00 |
| ML </s> | 1/1 | 1.00 |

Trigram Probability
Formula

$$P(w_3 \mid w_1, w_2) = \frac{Count(w_1 w_2 w_3)}{Count(w_1 w_2)}$$

| Trigram | Calculation | Probability |
|---|---|---|
| <s> I like | 2/3 | 0.67 |
| <s> I love | 1/3 | 0.33 |
| I like NLP | 1/2 | 0.50 |
| I like ML | 1/2 | 0.50 |
| I love NLP | 1/1 | 1.00 |
| like NLP </s>` | 1/1 | 1.00 |
| like ML </s>` | 1/1 | 1.00 |
| love NLP </s>` | 1/1 | 1.00 |

c) Evaluating N-grams
Find the probability of the sentence
<s> I like NLP </s>
Using Bigram model

$$P = P(I \mid < s >) \times P(like \mid I) \times P(NLP \mid like) \times P(</s> \mid NLP)$$

Substitute values

- $P(I \mid < s >) = 1$
- $P(like \mid I) = 2/3 = 0.67$
- $P(NLP \mid like) = 1/2 = 0.50$
- $P(</s> \mid NLP) = 1$

$$P = 1 \times 0.67 \times 0.50 \times 1$$
$$P = 0.335$$

d) Add-1 (Laplace) Smoothing
Find the probability of a bigram
love ML
Formula

$$P(w_2 \mid w_1) = \frac{Count(w_1 w_2) + 1}{Count(w_1) + V}$$

Where

- Count(love ML) = 0
- Count(love) = 1
- Vocabulary size $V = 7$

$$P(ML \mid love) = \frac{0 + 1}{1 + 7}$$
$$= \frac{1}{8}$$
$$= 0.125$$

5. Given the following word counts from a corpus

| Bigram | Count |
|---|---|
| (the, cat) | 40 |
| (the, dog) | 20 |
| the | 100 |

Vocabulary size = 5

a) Compute the unsmoothed bigram probabilities.
b)Apply Laplace smoothing and recompute the probabilities.
c) Compute the probability of the bigram “the mouse” using Laplace smoothing
d)Perplexity of the bigram model for the sentence “the cat.”

a) Unsmoothed bigram probabilities (3 Marks)

The unsmoothed bigram probability is computed using Maximum Likelihood Estimation (MLE)

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

$$P(cat \mid the) = \frac{40}{100} = 0.4$$

$$P(dog \mid the) = \frac{20}{100} = 0.2$$

Thus, the unsmoothed model assigns probabilities only to observed bigrams.

b) Laplace smoothed bigram probabilities (4 Marks)

Laplace (add-one) smoothing formula

$$P(w_i \mid w_{i-1}) = \frac{Count(w_{i-1}, w_i) + 1}{Count(w_{i-1}) + V}$$

$$P(cat \mid the) = \frac{40 + 1}{100 + 5} = \frac{41}{105} \approx 0.390$$

$$P(dog \mid the) = \frac{20 + 1}{100 + 5} = \frac{21}{105} = 0.200$$

Smoothing slightly redistributes probability mass across all words in vocabulary.

c) **Probability of the bigram "the mouse" using Laplace smoothing**

Since "the mouse" does not occur in the corpus:

- $C(\text{the, mouse}) = 0$
- $C(\text{the}) = 100$
- $V = 5$

$$P(\text{mouse} \mid \text{the}) = \frac{0 + 1}{100 + 5} = \frac{1}{105}$$

$$= 0.0095$$

$$P(\text{mouse} \mid \text{the}) = 0.0095$$

The smoothed probability is slightly lower due to the redistribution of probability.

**d) Perplexity of the bigram model for the sentence "the cat."**

using the unsmoothed bigram model.

Bigram probability

$$P(\text{cat} \mid \text{the}) = \frac{40}{100} = 0.40$$

First compute log

$$\log_2(0.40) \approx -1.32$$

Now substitute:

$$PP = 2^{-(-1.32)}$$

$$PP = 2^{1.32}$$

$$PP \approx 2.5$$

6. **A trigram language model is trained on a corpus with the following statistics**
   - Count(the) = 500
   - Count(the, students) = 120
   - Count(the, students, study) = 60
   - Count(students) = 300
   - Count(students, study) = 90
   - Vocabulary size $V = 8$

Test sentence "The students study".

a. **Unsmoothed Trigram Probability using MLE (4 Marks)**

In a trigram language model, the probability of a word depends on the previous two words.
The Maximum Likelihood Estimation (MLE) for a trigram is

$$P(w_i \mid w_{i-2}, w_{i-1}) = \frac{Count(w_{i-2}, w_{i-1}, w_i)}{Count(w_{i-2}, w_{i-1})}$$

For the given sentence, the trigram probability is

$$P(\text{study} \mid \text{the, students}) = \frac{Count(\text{the, students, study})}{Count(\text{the, students})}$$

$$P(\text{study} \mid \text{the, students}) = \frac{60}{120} = 0.5$$

Thus, the unsmoothed trigram probability of the sentence is

$$P(\text{the students study}) = 0.5$$

b. **Trigram Probability using Laplace Smoothing (4 Marks)**

Laplace (add-one) smoothing is applied to avoid zero probabilities

$$P(w_i \mid w_{i-2}, w_{i-1}) = \frac{Count(w_{i-2}, w_{i-1}, w_i) + 1}{Count(w_{i-2}, w_{i-1}) + V}$$

Substituting the given values

$$P(\text{study} \mid \text{the, students}) = \frac{60 + 1}{120 + 8}$$

$$P(\text{study} \mid \text{the, students}) = \frac{61}{128} \approx 0.477$$

After smoothing, the trigram probability is slightly reduced due to the redistribution of probabilities.

c. **Effect of Data Sparsity in Higher-Order N-gram Models (4 Marks)**

Data sparsity is a major problem in higher-order N-gram models, such as trigrams, because
- The number of possible N-grams increases exponentially with $N$
- Many valid word sequences do not appear in the training corpus
- Unseen N-grams receive zero probability in unsmoothed models

As a result,
- Sentence probabilities become zero
- Model generalization becomes poor
- Performance degrades on unseen data

This problem becomes more severe as we move from bigrams to trigrams and higher-order models.

**d.Role of Interpolation and Backoff in Improving Robustness (4 Marks)**

In NLP language models, one of the main challenges is **data sparsity** — many valid word sequences may not appear in the training data. This makes higher-order models, such as trigrams, unreliable because some probabilities may be zero or based on very few observations. To address this, Backoff and Interpolation are used to improve robustness and improve probability estimation.

**Interpolation**

Interpolation combines probabilities from trigram, bigram, and unigram models instead of choosing only one level.

$$P = \lambda_3 P_{tri} + \lambda_2 P_{bi} + \lambda_1 P_{uni}$$

Where:

- $P_{tri}$= Trigram probability
- $P_{bi}$= Bigram probability
- $P_{uni}$= Unigram probability
- $\lambda_3, \lambda_2, \lambda_1$are weights such that

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

Even if the trigram exists, interpolation still considers bigram and unigram information. This makes the probability estimate more stable and less dependent on a single data source.

**Backoff**

Backoff is a technique in which the model uses a lower-order model when a higher-order probability is unavailable or unreliable.

- In a trigram model, we try to compute

  Trigram:$P(w_i \mid w_{i-2}, w_{i-1})$
- If this trigram is not found in the training data, the model "backs off" to:

  Bigram: $P(w_i \mid w_{i-1})$
- If the bigram is also missing, it further backs off to:

  Unigram: $P(w_i)$

Thus, backoff ensures that the model never assigns zero probability just because a specific trigram was not seen. It uses simpler models as a fallback, making the system more reliable when data is sparse.

**Role in Improving Robustness**

- Prevents zero probability problems due to missing word sequences.
- Handles sparse data effectively.
- Backoff provides a fallback mechanism to simpler models.
- Interpolation combines multiple levels of context for more balanced and reliable predictions.
- Improves overall language model performance in tasks like speech recognition, text prediction, and machine translation.

7.**a) Probability of tag sequence N → V**

"Sentence: **fish sleep**"

a**)Tag path: N for "fish", V for "sleep"**

$$P = P(N \mid Start) \times P(fish \mid N) \times P(V \mid N) \times P(sleep \mid V)$$

Substitute values:

$$= 0.6 \times 0.4 \times 0.7 \times 0.5$$

Stepwise:

**Answer:**

$$P(N \rightarrow V) = 0.084$$

**b) Probability of tag sequence V → N**

Tag path: V for "fish", N for "sleep"

$$P = P(V \mid Start) \times P(fish \mid V) \times P(N \mid V) \times P(sleep \mid N)$$
$$= 0.4 \times 0.6 \times 0.8 \times 0.5$$

**Answer:**

$$P(V \rightarrow N) = 0.096$$

**c) Viterbi Algorithm (Step-by-Step)**

We compute the best tag at each word.

**Step 1: Word = "fish"**

For tag N:

$$0.6 \times 0.4 = 0.24$$

For tag V:

$$0.4 \times 0.6 = 0.24$$

| Tag | Probability |
|---|---|
| N | 0.24 |
| V | 0.24 |

**Step 2: Word = "sleep"**

**Case 1: End in N**

From previous N:

$$0.24 \times 0.3 \times 0.5 = 0.036$$

From previous V:

$$0.24 \times 0.8 \times 0.5 = 0.096$$

Take maximum → **0.096 from V**

**Case 2: End in V**

From previous N:

$$0.24 \times 0.7 \times 0.5 = 0.084$$

From previous V:

$$0.24 \times 0.2 \times 0.5 = 0.024$$

Take maximum → **0.084 from N**

**d) Final Best Tag Sequence**

Final probabilities at second word:

| Final Tag | Probability | Came From |
|---|---|---|
| N | 0.096 | V |
| V | 0.084 | N |

Maximum = **0.096**
So best final tag = **N**
Backtracking:

- N came from V
- So first tag = V

**Best sequence:**

$$\boxed{V \rightarrow N}$$

**e) Viterbi Trellis Table**

| Word | Tag N | Tag V |
|---|---|---|
| fish | 0.24 | 0.24 |
| sleep | 0.096 (from V) | 0.084 (from N) |

**f) Maximum Probability Value**
Compare final values:

- Path N → V = 0.084
- Path V → N = **0.096**

**Maximum probability:**

$$\boxed{0.096}$$