**University of Central Missouri**

**Department of Computer Science & Cybersecurity**

**CS5760 Natural Language Processing**

**Fall 2025**

**Homework 2.**

**Student name:**

**Venkata Nanda Krishna Yaram**

**700765514**

**Submission Requirements:**

* Once finished your assignment push your source code to your repo (GitHub) and explain the work through the ReadMe file properly. Make sure you add your student info in the ReadMe file.
* Submit your GitHub link on the Bright Space.
* Comment your code appropriately ***IMPORTANT.***
* Any submission after provided deadline is considered as a late submission.

**Q1. Bayes Rule Applied to Text (based on slide: Bayes’ Rule for documents)**  
The PPT shows that classification is based on:



### **Tasks:**

1. Explain in your own words what each term means: P(c), P(d∣c) and P(c∣d).
2. Why can the denominator P(d) be ignored when comparing classes?

ANSWER 1

### **1. Explanation of terms**

* **P(c)**  
  This is the **prior probability of class c**.  
  It reflects how likely a class is before seeing the document. For example, if 70% of all emails are not spam, then for the "not spam" class, P(c)=0.7P(c) = 0.7P(c)=0.7.
* **P(d ∣ c)**  
  This is the **likelihood**: the probability of observing the document ddd given that it belongs to class ccc.  
  In text classification, this is typically estimated using word frequencies example how likely is it to see the words "discount" and "buy" if the email is spam.
* **P(c ∣ d)**  
  This is the **posterior probability of class c given the document d**.  
  It is what we really want to know given the words in the document what is the probability that it belongs to class ccc.

### **2. Why can P(d)be ignored?**

**Bayes’ Rule is:**

P(c∣d)=P(c)P(d∣c)P(d)P(c \mid d) = \frac{P(c) P(d \mid c)}{P(d)}P(c∣d)=P(d)P(c)P(d∣c)​

* P(d)P(d)P(d) is the **probability of the document** itself, regardless of class.
* When comparing classes, P(d)P(d)P(d) is the same for all classes (since it doesn’t depend on ccc).
* Therefore, it acts as a **normalizing constant** to make probabilities sum to 1, but it doesn’t affect which class has the maximum probability.
* This is why, in classification, we only maximize P(c)P(d∣c)P(c) P(d \mid c)P(c)P(d∣c), and safely ignore P(d)P(d)P(d).

EXAMPLE

**Email classification** (word = “discount”):

* P(Spam)=0.4P(\text{Spam}) = 0.4P(Spam)=0.4
* P(Not Spam)=0.6P(\text{Not Spam}) = 0.6P(Not Spam)=0.6
* P(“discount”∣Spam)=0.3P(\text{“discount”} \mid \text{Spam}) = 0.3P(“discount”∣Spam)=0.3
* P(“discount”∣Not Spam)=0.05P(\text{“discount”} \mid \text{Not Spam}) = 0.05P(“discount”∣Not Spam)=0.05

Compute scores (ignoring denominator P(d)P(d)P(d)):

* Spam: 0.4×0.3=0.120.4 \times 0.3 = 0.120.4×0.3=0.12
* Not Spam: 0.6×0.05=0.030.6 \times 0.05 = 0.030.6×0.05=0.03

Since 0.12 > 0.03, **the email is classified as Spam.**

**Q2. Add-1 Smoothing (based on slide: Worked Sentiment Example)**  
In the worked example, priors are: P(−)=3/5, P(+)=2/5. Vocabulary size = 20.

### **Tasks:**

1. For the negative class, the total token count is 14. Compute the denominator for likelihood estimation using add-1 smoothing.
2. Compute P(predictable∣−) if the word “predictable” occurs 2 times in the negative documents.
3. Compute P(fun∣−) if “fun” never appeared in any negative documents.

ANSWER

**1. Denominator for likelihood estimation:**

Total tokens + Vocabulary size = 14 + 20 = 34

**2. P(predictable | -):**

= (2 + 1) / 34 = 3/34 ≈ 0.0882

**3. P(fun | -):**

= (0 + 1) / 34 = 1/34 ≈ 0.0294

**Q3. Worked Example Document Classification (based on slide: Test document “predictable no fun”)**  
Using the smoothed likelihoods and priors from Q2, compute the probability scores for the document *“predictable no fun”* under both the positive and negative classes.

### **Tasks:**

1. Show each step of the multiplication.
2. Which class should the system assign to this document?

ANSWER

# **Worked Example Document Classification**

Test document: **"predictable no fun"**

## **Given Information**

• Priors: P(-) = 3/5, P(+) = 2/5

• Vocabulary size = 20

• Negative class: total tokens = 14 → denominator = 14 + 20 = 34

- P(predictable | -) = (2+1)/34 = 3/34

- P(no | -) = (1+1)/34 = 2/34 (assumed)

- P(fun | -) = (0+1)/34 = 1/34

• Positive class: total tokens = 9 → denominator = 9 + 20 = 29 (assumed from worked example)

- P(predictable | +) = (0+1)/29 = 1/29

- P(no | +) = (1+1)/29 = 2/29

- P(fun | +) = (2+1)/29 = 3/29

## **Step 1: Negative Score**

Score(-) = P(-) × P(predictable | -) × P(no | -) × P(fun | -)

= (3/5) × (3/34) × (2/34) × (1/34)

= 0.6 × (6 / 39304) ≈ 9.16 × 10^(-5)

## **Step 2: Positive Score**

Score(+) = P(+) × P(predictable | +) × P(no | +) × P(fun | +)

= (2/5) × (1/29) × (2/29) × (3/29)

= 0.4 × (6 / 24389) ≈ 9.85 × 10^(-5)

## **Step 3: Classification**

Negative score ≈ 9.16 × 10^(-5)

Positive score ≈ 9.85 × 10^(-5)

Since Positive score > Negative score, **the document is classified as POSITIVE**.

**Q4. Harms of Classification (based on slide: Avoiding Harms in Classification)**

### **Tasks:**

1. Define **representational harm** and explain how the Kiritchenko & Mohammad (2018) study demonstrates this type of harm.
2. What is one risk of censorship in toxicity classification systems (based on Dixon et al. 2018, Oliva et al. 2021)?
3. Give one reason why classifiers may perform worse on African American English or Indian English, even though they are varieties of English.

ANSWER

### **Harms of Classification**

**1. Representational harm**

* **Definition:** Representational harm occurs when a system reinforces or spreads negative stereotypes or unfair associations about particular groups, even if it does not directly deny access to resources.
* **Example (Kiritchenko & Mohammad, 2018):** Their study showed that sentiment analysis systems rated phrases mentioning some social groups more negatively than equivalent phrases for others. This demonstrates representational harm because it embeds and amplifies stereotypes in the way groups are “represented” in text classification outputs.

**2. Risk of censorship in toxicity classification**

* Toxicity classifiers can mistakenly label neutral or supportive speech from marginalized groups as “toxic.”
* **Example (Dixon et al., 2018; Oliva et al., 2021):** Words reclaiming slurs or common identity terms (example, “gay,” “Black”) were often flagged as toxic leading to over-blocking of legitimate speech.
* **Risk:** This creates censorship where important conversations especially from marginalized voices are silenced.

**3. Classifier performance on African American English (AAE) or Indian English**

* These varieties differ in grammar, spelling, and vocabulary from Standard American or British English.
* If training data is dominated by Standard English the classifier sees fewer examples of AAE or Indian English.
* **Reason:** Data imbalance and lack of representation cause the system to misinterpret or misclassify these varieties lowering accuracy.

**Q5: Evaluation Metrics from a Multi-Class Confusion Matrix**

The system classified 90 animals into Cat, Dog, or Rabbit. The results are shown below:

| **System \ Gold** | **Cat** | **Dog** | **Rabbit** |
| --- | --- | --- | --- |
| **Cat** | 5 | 10 | 5 |
| **Dog** | 15 | 20 | 10 |
| **Rabbit** | 0 | 15 | 10 |

Tasks:

1. Per-Class Metrics
   * Compute precision and recall for each class (Cat, Dog, Rabbit).
2. Macro vs. Micro Averaging
   * Compute the macro-averaged precision and recall.
   * Compute the micro-averaged precision and recall.
   * Briefly explain the difference in interpretation between macro and micro averaging.
3. Programming Implementation  
   Write Python code that:
   * Accepts the confusion matrix above as input.
   * Computes per-class precision and recall.
   * Computes macro-averaged and micro-averaged precision and recall.
   * Prints all results clearly.

ANSWER

# **Evaluation Metrics from a Multi-Class Confusion Matrix**

## **Confusion Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| System ↓ / Gold → | Cat | Dog | Rabbit |
| Cat | 5 | 10 | 5 |
| Dog | 15 | 20 | 10 |
| Rabbit | 0 | 15 | 10 |

## **1. Per-Class Precision & Recall**

Cat: Precision = 0.25, Recall = 0.25

Dog: Precision = 0.40, Recall ≈ 0.444

Rabbit: Precision = 0.40, Recall = 0.40

## **2. Macro vs. Micro Averaging**

Macro Precision = (0.25 + 0.40 + 0.40) / 3 ≈ 0.35

Macro Recall = (0.25 + 0.444 + 0.40) / 3 ≈ 0.365

Micro Precision = 35 / 95 ≈ 0.368

Micro Recall = 35 / 90 ≈ 0.389

Interpretation: Macro averaging treats each class equally, while Micro averaging weights larger classes more. Macro is useful for balanced evaluation; Micro is better when classes are imbalanced.

OUTPUT

**3.Programming Implementation**

A screenshot of a computer program

AI-generated content may be incorrect.

# **Q6. Bigram Probabilities and the Zero-Probability Problem**

You are given the following bigram counts from a small training corpus:

| **Previous word** | **Next words (with counts)** |
| --- | --- |
| <s> | I: 2 , deep: 1 |
| I | love: 2 |
| love | NLP: 1 , deep: 1 |
| deep | learning: 2 |
| learning | </s>: 1 , is: 1 |
| NLP | </s>: 1 |
| is | fun: 1 |
| fun | </s>: 1 |
| ate | lunch: 6 , dinner: 3 , a: 2 , the: 1 |

### **Tasks:**

1. **Bigram Sentence Probabilities**  
   Using maximum likelihood estimation (MLE):

A black and white math equation

AI-generated content may be incorrect.

* + Compute the probability of sentence **S1:** <s> I love NLP </s>.
  + Compute the probability of sentence **S2:** <s> I love deep learning </s>.
  + Which sentence is more probable under the bigram model?

1. **Zero-Probability Problem**  
   Using the same table, compute:
   * P(noodle∣ate) with MLE.
   * Explain why this probability creates problems when computing sentence probabilities or perplexity.
   * Apply **Laplace smoothing (Add-1)** to recompute P(noodle∣ate). Assume the vocabulary size is 10 and total count after “ate” is 12.

ANSWER

# **Bigram Probabilities and the Zero-Probability Problem**

## **1. Bigram Sentence Probabilities (MLE)**

**Sentence S1:** <s> I love NLP </s>

P(I | <s>) = 2/3  
P(love | I) = 2/2 = 1  
P(NLP | love) = 1/2  
P(</s> | NLP) = 1/1 = 1  
=> P(S1) = (2/3) × 1 × (1/2) × 1 = 1/3 ≈ 0.333

**Sentence S2:** <s> I love deep learning </s>

P(I | <s>) = 2/3  
P(love | I) = 1  
P(deep | love) = 1/2  
P(learning | deep) = 2/2 = 1  
P(</s> | learning) = 1/2  
=> P(S2) = (2/3) × 1 × (1/2) × 1 × (1/2) = 1/6 ≈ 0.167

Conclusion: P(S1) = 0.333 > P(S2) = 0.167, **so sentence S1 is more probable under the bigram model.**

## **2. Zero-Probability Problem**

**a) MLE Estimate:**

P(noodle | ate) = C(ate, noodle) / C(ate) = 0/12 = 0

**b) Problem:**

This is problematic because any unseen bigram leads to a sentence probability of 0, which breaks probability calculations and perplexity evaluation.

**c) Laplace (Add-1) Smoothing:**

Formula: P(w | ate) = (C(ate, w) + 1) / (C(ate) + V)  
Vocabulary size V = 10, total count after 'ate' = 12, C(ate, noodle) = 0  
=> P(noodle | ate) = (0+1) / (12+10) = 1/22 ≈ 0.0455

**Final Answers :**

1. P(S1)=0.333P(S1) = 0.333P(S1)=0.333, P(S2)=0.167P(S2) = 0.167P(S2)=0.167 → **S1 more probable**
2. P(noodle∣ate)=0P(noodle \mid ate) = 0P(noodle∣ate)=0 with MLE → Problem = zero sentence probability With Laplace smoothing: P(noodle∣ate)=0.0455P(noodle \mid ate) = 0.0455P(noodle∣ate)=0.0455

### **Q7. Backoff Model (based on “Activity: <s> I like cats … You like dogs” slide)**

Training corpus:

<s> I like cats </s>

<s> I like dogs </s>

<s> You like cats </s>

Counts:

* I like = 2
* You like = 1
* like cats = 2
* like dogs = 1
* cats </s> = 2
* dogs </s> = 1

### **Tasks:**

1. Compute P(cats∣I,like).
2. Compute P(dogs∣You,like) using trigram → bigram backoff.
3. Explain why backoff is necessary in this example.

ANSWER

# **Backoff Model**

## **Training Corpus and Counts**

<s> I like cats </s>  
<s> I like dogs </s>  
<s> You like cats </s>

**Counts:**• I like = 2  
• You like = 1  
• like cats = 2  
• like dogs = 1  
• cats </s> = 2  
• dogs </s> = 1

## **1. Compute P (cats | I, like)**

Formula: P(cats | I, like) = C (I, like, cats) / C (I, like)  
C (I, like, cats) = 1, C (I, like) = 2  
=> P(cats | I, like) = 1/2 = 0.5

## **2. Compute P (dogs | You, like) using backoff**

Trigram model: P(dogs | You, like) = C(You, like, dogs) / C(You, like)  
C(You, like, dogs) = 0, C(You, like) = 1 → Trigram = 0  
Backoff to bigram: P(dogs | like) = C(like, dogs) / (C(like, cats)+C(like, dogs))  
= 1 / (2+1) = 1/3 ≈ 0.333  
=> P(dogs | You, like) = 0.333

## **3. Why Backoff is Necessary?**

Backoff is needed because unseen trigrams like ('You like dogs') get probability 0. This would make any sentence containing them impossible under the model. By backing off to bigrams or unigrams, the model assigns non-zero probability, allowing generalization and avoiding the zero probability problem.

### **Q8. Programming: Bigram Language Model Implementation (based on “Activity: I love NLP corpus” slide)**

### **Tasks:**

Write a Python program to:

1. Read the training corpus:
2. <s> I love NLP </s>
3. <s> I love deep learning </s>
4. <s> deep learning is fun </s>
5. Compute unigram and bigram counts.
6. Estimate bigram probabilities using MLE.
7. Implement a function that calculates the probability of any given sentence.
8. Test your function on both sentences:
   * <s> I love NLP </s>
   * <s> I love deep learning </s>
9. Print which sentence the model prefers and why.

OUTPUT

A screenshot of a computer program

AI-generated content may be incorrect.