University of Central Missouri

Department of Computer Science & Cybersecurity

CS5760 Natural Language Processing

Fall 2025

Homework 2.

Student name: SANTHOSH REDDY KISTIPATI

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:

$$c_{MAP} = \arg\max_{c \in C} P(c)\, P(d \mid c)$$

Tasks:

1. **Explain in your own words what each term means: P(c), P(d|c) and P(c|d).**

ANS :Bayes Rule Applied to Text

What the terms mean

P(c): the **prior** probability of class ccc (e.g., how common "positive" vs "negative" is before seeing the document).

P(d∣c): the **likelihood** of the document ddd if it came from class ccc (with Naïve Bayes this becomes a product of word probabilities given the class).

P(c∣d): the **posterior**—what we actually want: the probability the document belongs to class ccc after seeing its words.

2) Why can the denominator P(d) be ignored when comparing classes?

ANS: Why we can ignore P(d)P(d)P(d) when comparing classes

- Bayes: P(c∣d)=P(d∣c) P(c) ⁄ P(d)
- For the *same* document ddd, the denominator P(d)P(d)P(d) is the same for every class.
- So when choosing the argmax class, compare P(d∣c) P(c)P(d\mid c)\,P(c)P(d∣c)P(c) only; P(d)P(d)P(d) cancels out.

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.

   ANS: N_+V=14+20=34.

2. Compute P(predictable|−) if the word "predictable" occurs 2 times in the negative documents.

   Ans : **(predictable∣−) when count−(predictable)=2**

   = 2+1/[14+20] = 3/34 =0.08824

3. Compute P(fun|−) if "fun" never appeared in any negative documents.

Ans: **P(fun∣−) when never seen in negative**

==0+1/[14+20] =1/34 = 0.02941

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.**

   Ans: Use Naïve Bayes with your smoothed word likelihoods and the priors from Q2.

Document likelihood under class ccc

$$= \text{score(c)} \propto \text{P(c)} \prod\nolimits_{w \in d} P(w \mid c)$$

Negative class (fully numeric with Q2 values)

We already have (from Q2):

$P(\text{predictable}|-) = 3/34$,

$P(\text{fun}|-)=1/34$

We also need $P(\text{no}|-)$. Two common cases:

- If "no" unseen in negative: $P(\text{no}|-) =[0+1]/ 34=1/34$

- If "no" appears once: $P(\text{no}|-)= [1+1]/34=2/34$

Then

$\text{score}(-) = P(-) \cdot 3/34 \cdot P(\text{no}|-) \cdot 1/34 = 3/5 \cdot 3/34 \cdot P(\text{no}|-) \cdot 1/34$

- Unseen "no" case: $\text{score}(-)=3/5 \cdot 3/34 \cdot 1/34=0.0001553$

- Seen-once "no" case: $\text{score}(-)=3/5 \cdot 3/34 \cdot 2/34 = 0.0003106$

2. Which class should the system assign to this document?

Ans: Positive class , Because the prompt didn't include the positive token totals or the counts of predictable/no/fun in +, plug those in from your slide to get the final numeric comparison. In many textbook examples, "fun" is more associated with + and "no/predictable" with −; together with $P(-) > P(+)$, the negative score often wins—but compute with your actual counts to be precise.

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.**

   Ans: Representational harm: When a system's outputs reinforce negative stereotypes or mischaracterize how a group speaks/behaves. Kiritchenko & Mohammad (2018) showed that sentiment/emotion systems can systematically rate language about certain groups (e.g., identity terms) as more negative, even when neutral, thus misrepresenting those groups.

2. **What is one risk of censorship in toxicity classification systems (based on Dixon et al. 2018, Oliva et al. 2021)?**

   Ans: Censorship risk in toxicity moderation: If models over-flag reclaiming/benign uses of identity terms, they can silence marginalized voices (Dixon et al., 2018; Oliva et al., 2021), removing non-toxic content and skewing discourse.

3. **Give one reason why classifiers may perform worse on African American English or Indian English, even though they are varieties of English.**

   Ans: Why worse on AAE or Indian English: Domain shift—training data underrepresents these varieties, so models learn mainstream spelling/grammar/lexicon. Misspellings to the model, different pragmatics, and code-switching reduce 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).

Ans:

Cat:

Precision =5/20=0.25

Recall =5/20=0.25

Dog:

- Precision =20/45=4/9=0.4444
- Recall =20/45=4/9=0.4444

Rabbit:

- Precision =10/25=0.4
- Recall =10/25=0.4

2. **Macro vs. Micro Averaging**
    - **Compute the macro-averaged precision and recall.**

      ANS : Pmacro=[0.25+0.4444+0.4]/3 = 0.3648,
      Rmacro=0.3648.

    - **Compute the micro-averaged precision and recall.**

      ANS : **Micro-averaged Precision and Recall**

From the confusion matrix (Q5):

Total true positives (sum of diagonal) = 5 + 20 + 10 = **35**

Total number of predictions = total number of examples = 90

Micro-Precision=Micro-Recall= Total True Positives/[Total Instances]

=35/90=0.3889so,

Micro-averaged precision = **0.3889**

Micro-averaged recall = **0.3889**

Briefly explain the difference in interpretation between macro and micro averaging.

Ans: **Macro averaging:**
Compute precision/recall separately for each class, then average them.
Each class has equal weight, **regardless of how many examples it has**. This highlights performance on minority class

Micro averaging:
Pool all true positives, false positives, and false negatives across classes, then compute precision/recall.
Each instance has equal weight, so **larger classes dominate** the score. This reflects overall accuracy across the dataset.

3. Programming Implementation
   Write Python code that:

   0. Accepts the confusion matrix above as input.
   1. Computes per-class precision and recall.
   2. Computes macro-averaged and micro-averaged precision and recall.
   3. Prints all results clearly.

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

Previous word	Next words (with counts)
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):

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

- **Compute the probability of sentence S1: <s> I love NLP </s>.**

Ans: **Sentence 1 Probability**

Sentence S1 = `<s> I love NLP </s>`

P(I | <s>)=⅔

P(love | I)=2/2=1

P(NLP | love)=½

P(</s> | NLP)=1/1=1

P(S1)=2/3×1×1/2×1=⅓

=0.3333

- **Compute the probability of sentence S2: <s> I love deep learning </s>.**

Ans: **Sentence 2 Probability**

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=⅙

=0.1667

Which sentence is more probable under the bigram model?

Ans: P(S1)=0.3333
P(S2)=0.1667

Conclusion: S1 is more probable than S2 under the bigram model

2. **Zero-Probability Problem**
   Using the same table, compute:
   - **P(noodle|ate) with MLE.**

Ans: From the table:
After "ate" → lunch:6, dinner:3, a:2, the:1 → total = 12.

Count(ate, noodle) = 0 (never occurs).

MLE formula:

$$P(w \mid h)=\text{count}(h,w)/[\sum x \text{count}(h,x)]$$

Therefore:

$$P(\text{noodle} \mid \text{ate})=0/12=0$$

Answer: 0

- **Explain why this probability creates problems when computing sentence probabilities or perplexity.**

Ans:If a bigram has probability 0, then any sentence containing that bigram gets probability 0 (since we multiply probabilities).
This causes:

Sentence probability = 0 for unseen sequences.
Perplexity = ∞ (infinite), which makes evaluation meaningless.

- **Apply Laplace smoothing (Add-1) to recompute P(noodle|ate). Assume the vocabulary size is 10 and total count after "ate" is 12.**

  Ans; **Formula:**

  $$P(w|h)=count(h,w)+1/[\sum xCount(h,x)+|V|]$$

Given:

Count(ate, noodle) = 0

Total after "ate" = 12

Vocabulary size $|V|=10|V| = 10|V|=10$

Compute:

$P(noodle|ate)=0+1/12+10=122$

$=0.0455$

Answer: 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).**

   Ans: In the corpus: "I like cats" appears 1 time; after "I like …" we have cats and dogs once each → total 2 continuations.

   $$P(\text{cats} \mid \text{I,like}) = \text{count}(\text{I like cats}) / [\text{count}(\text{I like} \cdot)]$$

   $$= 1/2 = 0.5$$

2. **Compute P(dogs|You,like) using trigram → bigram backoff.**

   Ans: Trigram "You like dogs" is unseen ⇒ back off to bigram:
   After "like": we have cats:2, dogs:1 → total 2+1=32+1=32+1=3.

   $$P(\text{dogs} \mid \text{like}) = 1/3$$

   $$= 0.3333$$

3. **Explain why backoff is necessary in this example.**

   Ans: trigrams are sparse; backing off uses broader context (bigrams) instead of assigning zero probability.

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.