HW 1
PRELIMINARIES AND N-GRAM LANGUAGE MODELING

CS 678 ADVANCED NATURAL LANGUAGE PROCESSING (FALL 2025)
https://nlp.cs.gmu.edu/course/cs678-fall25/

OUT: Sept 02, 2025

DUE: Sept 16, 2025

TOTAL CREDITS: 100 Points
(Contribution to final grade = points earned × 10%)

Your name: RIYA ELIZABET STANLY

Your GID: G01521289

Overview: This homework includes two parts. In Part 1, you will answer questions about the logistics of this course. In Part 2, you will build your first language model in CS678. Specifically, you will implement a bigram language model (Part 2.1), evaluate its performance (Part 2.2), and sample texts from it (Part 2.3). The assignment also comes with a Jupyter notebook (https://jupyter.org/), which can be completed on your laptop/computer locally or on the cloud by using Google Colab (https://colab.research.google.com/), although using Colab requires extra effort in setting up the data access.

Submission Guideline:

- Complete the Jupyter notebook, **make sure to save all the cell outputs for grading**, and submit your notebook to Canvas.
- Complete this LaTeX assignment project and compile a PDF. Submit the PDF to Gradescope.

Important Notes: (1) For Part 1 (class logistics), you are welcome to ask for clarification about class logistics from the instructor or the TAs. For Part 2: (2) Please do NOT modify existing code/markdown cells in the notebook unless instructed. (3) While there may have been well-packed Python tools for producing and evaluating language models, you should not use them. Instead, you are expected to implement the language model from scratch, following the instructions in the notebook. The only exceptions are commonly used general-purpose Python libraries such as `numpy` and `collections`, as they do not directly produce or evaluate a language model. Please consult the grader or the instructor if you have any questions. AI tools such as ChatGPT and CoPilot are disallowed and violate the university's Academic Standards.

Instructions for Specific Problem Types: For "Select One" questions, please fill in the appropriate bubble completely:

Select One: Who is teaching this course?

● Ziyu Yao ○ Marie Curie ○ Noam Chomsky

For "Select all that apply" questions, please fill in all appropriate squares completely:

Select all that apply: Which are scientists?

■ Stephen Hawking ■ Albert Einstein ■ Isaac Newton □ None of the above

For questions requiring a short answer, please fill in the blank underlined:

Give a short answer to the question: Where is George Mason University located? Fairfax, VA

For questions requiring a long answer, please fill in the blank in the text box. Do not modify the template code; only replace the text portion with your answer:

Give a long answer to the question: What do we study in CS678?

In CS678, we will study building computing systems that can process, understand, and communicate in natural language. This field is called natural language processing, or NLP. The syllabus and other logistics information can be found at https://nlp.cs.gmu.edu/course/cs678-fall25/syllabus/.

Part 1: Class Logistics (20 points)

1. Who are the TAs? Saurabh Srivastava and Hao Yan
2. How many assignments will there be? ◯ 1 ● 3 ◯ 6 ◯ 8
3. You can find all assignment release and due dates on the course website. You are responsible for keeping track of these dates. ● True ◯ False
4. Homework assignments can be done in groups. ◯ True ● False
5. How many late days *in total* are allowed without penalty? ◯ 0 ● 5 ◯ 15 ◯ 20
6. You have already used out your permissible late days for HW 1 and HW2, and now you deliver HW 3 2 days late. Suppose your raw grade for HW 3 is 90 out of 100 points, what will be your actual grade for HW 3 after the late-day penalty? 80/100
7. In-class activities, including quizzes, project presentations, and project peer review, cannot be late or taken home. ● True ◯ False
8. If Alice received 90 (out of 100) points for HW1, 100 (out of 100) points for HW2, 80 (out of 100) points for HW3, 90 (out of 100) points for the project, and full points for class participation, her final grade will be: $90 \times 10\% + 100 \times 20\% + 80 \times 20\% + 90 \times 30\% + 20 = 92.0$. ● True ◯ False
9. Assume a difficult situation arises in the middle of the semester (e.g., illness) that might prevent you from submitting assignments on time (even with the permissible late days). What should you do?

   Select all that apply:

   - ■ Talk to the course staff ASAP so they can point you to the available resources on campus and make necessary arrangements
   - □ Do not speak to the course staff, reach out to the course staff only at the end of the semester to explain your special situation
   - □ To meet the assignment deadline, use AI tools to generate solutions and submit the assignment on time

10. Assume you are confused about the class content or the assignments. Where and how could you seek help?

    Select all that apply:

    - ■ A. When you have confusion in class, you are encouraged to ask about it during or after the class;
    - ■ B. Ask questions on Piazza after the class;
    - ■ C. On Piazza, see if other students have raised the same question, and follow the discussion;
    - ■ D. Discuss with your classmates with the purpose of facilitating learning;
    - ■ E. Visit the TA office hour;
    - ■ F. Schedule an appointment with the instructor;
    - □ G. You should not seek help at all.

Part 2: N-gram Language Modeling (80 points)

Part 2.1: Language Model Construction (40 points)

Your first task is to implement the bigram language model.

Step 1 – Construct a vocabulary (15 points): To get started, you will first need to construct a vocabulary based on the `sents_train` corpus. Also, don't forget the special start-of-sentence (<s>) and end-of-sentence (</s>) tokens – your LM should eventually be able to model $p(w|\texttt{<s>})$ (i.e., how to start a sentence) and $p(\texttt{</s>}|w)$ (i.e., when to stop a sentence).

Now, show the size of your vocabulary (i.e., how many distinct words in your vocab, including UNK):

Your Code Output

Vocabulary size (including UNK): 7113

If we use this vocabulary to index sentences in `sents_train`, what are the most frequent words (including UNK)? Show the top 10 word types with their counts, one pair in a row.

Provide the top 10 word types and their counts below:

Your Code Output

```
, 9129
<s> 6173
</s> 6173
. 5513
to 4115
the 3805
and 3748
of 3386
UNK 2931
I 2518
```

Step 2 – Build the bigram LM (25 points): Next, implement the bigram LM $p(w_t|w_{t-1})$ (with add-one smoothing) based on the `sents_train` corpus.

First, accumulate the bigrams in `sents_train` and create a bigram counter.

For how many times in `sents_train` does a sentence start with the word "I"?

Your Code Output

Sentences in sent_train starts with the word 'I' for times: 502

Next, create the bigram language model (with add-one smoothing). **Copy your code below.**

Your Code

```
import time

start_time = time.time()

# TODO: create your bigram LM
model_p = dict() # a dict of {(w_t-1, w_t): probability}

# Calculate unigram counts including <s> and </s> for denominators in add-one smoothing
unigram_counts = Counter()
for sentence in sents_train:
    processed_sentence = ["<s>"] + [word if word in vocabulary else "UNK" for word in sentence] + ["</s>"]
    unigram_counts.update(processed_sentence)


vocab_size = len(vocabulary)

for (w_tm1, w_t), bigram_count in bigram_counts.items():
    # The denominator is count(w_t-1) + |V|
    # We use the unigram count of w_t-1 as the count(w_t-1)
    denominator = unigram_counts[w_tm1] + vocab_size
    model_p[(w_tm1, w_t)] = (bigram_count + 1) / denominator

# We can iterate through all possible (w_tm1, w_t) pairs where w_tm1 has a count > 0
for w_tm1 in unigram_counts:
    if w_tm1 == "</s>": # No bigrams start with </s>
        continue
    for w_t in vocabulary:
        if (w_tm1, w_t) not in model_p:
            # This bigram was not seen in the training data, so its count is 0
            bigram_count = 0
            denominator = unigram_counts[w_tm1] + vocab_size
            model_p[(w_tm1, w_t)] = (bigram_count + 1) / denominator

# Print the bigram probabilities
for key, value in model_p.items():
    print(f"{key} {value}")

# Do not modify code after this line
end_time = time.time()
print("Spent %s for the bigram LM construction." % time.strftime("%Hh%Mm%Ss", time.gmtime(end_time-start_time)))
```

Now, can you show the most frequent 10 starting words (i.e., words following <s>) with their probabilities to three decimal places (e.g., 0.123)?

Provide the most frequent 10 starting words and their probabilities below:

Your Code Output

```
", 0.123
I, 0.049
She, 0.036
He, 0.028
It, 0.023
The, 0.022
Emma, 0.017
But, 0.015
Mr, 0.014
You, 0.013
```

Part 2.2: Evaluation of a Language Model (20 points)

Your second task is to evaluate the bigram language model that you constructed in Part 2.1. Please follow the instructions and implement the instructed functions.

Step 1 – Calculate the log2 probability of a test sentence (10 points): Provide your `log2_P` output for the first sentence in `sents_test`.

Your Code Output

log2_P returns: -155.8266713670138

Step 2 – Calculate the per-word cross entropy (10 point): Provide your `H` function's output below:

Your Code Output

H returns: 10.38844475780092

Step 3 – Implement the perplexity metric: Provide your `ppl` output below:

Your Code Output

Perplexity: 527.0427483740344

Part 2.3: Sampling from a Language Model (20 points)

Your third task is to sample texts from the bigram language model you built in Parts 2.1 and 2.2. You will need to implement two sampling methods, greedy decoding and sampling by LM next-word distribution, each worth 1 point.

Step 1 – Greedy decoding (10 point): Provide your code below:

Your Code

```
sent = [] # used to store the generated words in your sentence
current_word = "<s>"
while current_word != "</s>" and len(sent) < max_len:
    # Find the next word with the highest probability given the current word
    next_word = None
    max_prob = -1.0
    for (w_tm1, w_t), prob in model_p.items():
        if w_tm1 == current_word:
            if prob > max_prob:
                max_prob = prob
                next_word = w_t
    if next_word is not None:
        if next_word != "</s>":
            sent.append(next_word)
        current_word = next_word
    else:
        break
print(sent)
```

Provide the greedily decoded sentence by your implementation below:

Your Code Output

['"', 'I', 'am', 'sure', ',', 'and', 'the', 'UNK', ',', 'and', 'the', 'UNK', ',', 'and', 'the', 'UNK', ',', 'and', 'the', 'UNK', ',', 'and', 'the', 'UNK', ',', 'and', 'the', 'UNK', ',', 'and', 'the', 'UNK', ',', 'and', 'the', 'UNK', ',', 'and', 'the', 'UNK', ',', 'and']

Step 2 – Sampling by distribution (10 point): Provide your code below:

Your Code

```
np.random.seed(1234) # for grading purpose, do not change the random seed
sent = [] # used to store your sentence
current_word = "<s>"
while current_word != "</s>" and len(sent) < max_len:
    next_word_probs = {w_t: prob for (w_tm1, w_t), prob in model_p.items() if
    w_tm1 == current_word}

    words = list(next_word_probs.keys())
    probabilities = list(next_word_probs.values())
    if words:
        next_word = np.random.choice(words, p=probabilities)

        if next_word != "</s>": # Don't add </s> to the sentence list
            sent.append(next_word)
        current_word = next_word
    else:
        break
print(sent)
```

Provide your sampled sentence below:

Your Code Output

```
['She', 'keep', 'reflected', 'liberal', 'Lord', 'dwelt', 'gladly', '
    _There_',
'exert', 'disappointed', 'entering', 'basin', 'fresh', 'lips', '
    governess',
'oblige', 'form', 'acquaintance', 'sensation', 'alluded', 'examine', '
    similar',
'ideas', 'get', 'neighbour', 'thin', 'latest', 'partner', 'appear', '
    meals',
'suspected', '.--"', 'intelligence', 'thought', 'believe', 'Farm', '
    accidental',
'staircase', 'rapidity', 'contrived', 'station', 'had', 'been', '
    increase',
'Wallis', 'conversation', 'described', 'another', 'Papa', '
    compassionate']
```