CS 4740 (+crosslists): Natural Language Processing **(Due: February 21, 11.59 p.m.)**

Homework #1

Instructors: Tanya Goyal

Submission by (netIDs): kl2235, zl2254

Logistics: Read all the instructions carefully before you start working on the assignment, and before you make a submission.

- The assignment is due on February 21, 11.59 p.m. **Please read the submission instructions included at the end of this document carefully.**
- You can do this homework either individually or in a group of 2. We strongly recommend doing this in a group. All students in a group will receive the same grade. You will **declare your group by February 16, 11.59 p.m.** on gradescope.
- This assignment is worth 10% of your grade. Make sure to submit this assignment by the due date mentioned above. Remember that you have 5 slip days to use throughout the course. You may use a maximum of 2 slip days for each assignment. Refer to the course webpage for more information about late submission policy.
- Please typeset your assignment in LaTeX. You can use this template for your written answers. Please include the netID(s) of all members in the group at the top of this page. (Search for "FILL YOUR NETID" to find the appropriate place to enter your netIDs).
- You will submit this assignment on gradescope. We have created 3 assignments: "hw1-partnerdeclaration", "hw1-written" and "hw1-programming". You will submit the written parts of the assignment in the form of a pdf to "hw1-written". This will include your answers to Section A and the written questions in Section B. You will submit code or outputs of code to "hw1-written". Please read the submission instructions included at the end of this document for details on this.
- The University Academic Code of Conduct will be strictly enforced.

Section A: Conceptual Questions (40 points)

N-gram Language Models

(1) Consider the following small corpus of sentences:

$$< s > \textit{ the cat sat on the mat } < /s >$$
$$< s > \textit{ the dog sat } < /s >$$
$$< s > \textit{ the mat is on the floor } < /s >$$

where $< s >$ and $< /s >$ are tokens representing the start and end of a sentence respectively. Remember that in all probability computations of n-gram models, we will ignore the unigram probability of the token $< s >$.

(a) (4 points) Provide all **non-zero** bigram probabilities, i.e. $P(w_{i+1}|w_i)$ for this corpus. Your vocabulary set for this corpus is $\mathcal{V} = \{\text{the, cat, dog, sat, on, is, mat, floor}, < s >, < /s >\}$.

(Solution) We first count all bigrams in the corpus and compute:

$$P(w_{i+1} \mid w_i) = \frac{\text{count}(w_i, w_{i+1})}{\text{count}(w_i)}.$$

The non-zero bigram probabilities are:

- $P(\text{the} \mid < s >) = 1$
- $P(\text{cat} \mid \text{the}) = \frac{1}{5}$, $P(\text{dog} \mid \text{the}) = \frac{1}{5}$, $P(\text{mat} \mid \text{the}) = \frac{2}{5}$, $P(\text{floor} \mid \text{the}) = \frac{1}{5}$
- $P(\text{sat} \mid \text{cat}) = 1$
- $P(\text{sat} \mid \text{dog}) = 1$
- $P(\text{on} \mid \text{sat}) = \frac{1}{2}$, $P(< /s > \mid \text{sat}) = \frac{1}{2}$
- $P(\text{the} \mid \text{on}) = 1$
- $P(\text{on} \mid \text{is}) = 1$
- $P(< /s > \mid \text{floor}) = 1$
- $P(\text{is} \mid \text{mat}) = \frac{1}{2}$, $P(< /s > \mid \text{mat}) = \frac{1}{2}$

All other bigram probabilities are zero.

(b) (2 points) Give 2 examples of grammatically valid sentences in the English language that will be assigned a zero probability according to this bi-gram language model. Your examples should only use tokens in the above vocabulary set $\mathcal{V}$.

(Solution) Under the bigram model, if any conditional probability is zero, then the entire sentence has zero probability.

Two examples are:

- $< s >$ the cat is on the mat $< /s >$, since $P(\text{is} \mid \text{cat}) = 0$.
- $< s >$ the dog is on the floor $< /s >$, since $P(\text{is} \mid \text{dog}) = 0$.

Therefore, both sentences have zero probability.

(c) (4 points) The sentences seen in the training corpus (i.e. the 3 sentences above) will always have a strictly higher probability than any unseen grammatical sentence according to the above 2-gram model. True or False? Justify your choice with either a proof (if true) or a counterexample (if false).

(Solution) The statement is false.

Consider the seen sentence:

$$< s > \text{ the cat sat } < /s >$$

Its probability is:

$$1 \times \frac{1}{5} \times 1 \times \frac{1}{2} = \frac{1}{10}.$$

Consider the seen sentence:

$$< s > \text{ the cat sat on the mat } < /s >$$

Its probability is:

$$1 \times \frac{1}{5} \times 1 \times \frac{1}{2} \times 1 \times \frac{2}{5} \times \frac{1}{2} = \frac{1}{50}.$$

We have $\frac{1}{10} > \frac{1}{50}$, even though both sentences appear in the training data.

Therefore, the statement is false.

(d) (6 points) Suppose you are given a prefix $w_0 w_1 w_2 w_3 w_4 w_5 =$ "$< s >$ *the mat is on the*". We can use the bigram model to generate possible sentences, i.e. continuations ending in $< /s >$, for this prefix. The pseudocode for this algorithm is:

1. Initialize str $= w_0 ... w_4 w_5$
2. $i = |\text{str}| - 1$ # $i = 5$ for our example

3. while $(w_i \neq < /s >)$
4. Sample $w_{i+1} \sim P(w_{i+1}|w_0...w_i)$
5. str = str + w_{i+1}
6. $i = i + 1$
7. **return** str

Essentially, in each iteration of the while loop, we sample one word according to your bigram's next word probability. We continue to do this till you encounter the $< /s >$ token. This gives us one possible sentence according to the bigram language model.

Write 2 different sentences (sequences ending in $< /s >$) that have a non-zero probability according to your bigram language model starting from the above prefix ($w_0w_1w_2w_3w_4w_5 = < s >$ *the mat is on the*). Report both the sentences and the probability of the continuation, i.e., report $P(w_6....w_N| < s >$ the mat is on the), where $w_N =< /s >$.

(Solution) From part (a), we know:

$$P(\text{mat} \mid \text{the}) = \frac{2}{5}, \quad P(\text{floor} \mid \text{the}) = \frac{1}{5}.$$

$$P(< /s > \mid \text{mat}) = \frac{1}{2}, \quad P(< /s > \mid \text{floor}) = 1.$$

Starting from the prefix:

$$< s > \text{ the mat is on the,}$$

we can generate the following two sentences.

Sentence 1:

$$< s > \text{ the mat is on the mat } < /s >$$

The continuation is "mat $< /s >$", so its probability is:

$$P(\text{mat} \mid \text{the}) \times P(< /s > \mid \text{mat}) = \frac{2}{5} \times \frac{1}{2} = \frac{1}{5}.$$

Sentence 2:

$$< s > \text{ the mat is on the floor } < /s >$$

The continuation is "floor $< /s >$", so its probability is:

$$P(\text{floor} \mid \text{the}) \times P(< /s > \mid \text{floor}) = \frac{1}{5} \times 1 = \frac{1}{5}.$$

(e) (3 points) For the prefix in part (d), what is the total number of distinct non-zero probability sentences that we can generate? Comment on the grammatically of these, would a higher order n-gram model help or hurt?

(Solution) For the prefix in part (d),

$$< s > \text{ the mat is on the,}$$

the last token is "the". From part (a), the possible next words after "the" are:

$$\text{cat, dog, mat, floor.}$$

Key observation: the bigram model contains a cycle:

$$\text{the} \rightarrow \{\text{cat, dog}\} \rightarrow \text{sat} \rightarrow \text{on} \rightarrow \text{the.}$$

Moreover, from sat we can also terminate with non-zero probability since:

$$P(< /s > \mid \text{sat}) = \frac{1}{2} > 0.$$

Also, from the we can terminate via:

$$\text{the} \rightarrow \text{mat} \rightarrow < /s >, \qquad \text{the} \rightarrow \text{floor} \rightarrow < /s >,$$

both of which have non-zero probability.

Because we can traverse the cycle *any number of times* (and choose cat or dog each time) before eventually ending with either sat $\rightarrow < /s >$ or by choosing mat/floor and then $< /s >$, there are **infinitely many** distinct sentences with non-zero probability that can be generated from the prefix.

Grammaticality: Many generated sentences can be awkward or unnatural due to repetition, e.g.,

$$< s > \text{ the mat is on the dog sat on the cat sat } < /s >,$$

even though every bigram transition is valid under the model. This happens because a bigram model only conditions on one previous word and cannot enforce longer-range syntactic/semantic constraints.

Would a higher-order n-gram help or hurt? A higher-order model (e.g., trigram) would generally *help* reduce such unrealistic repetitions by using more context and breaking many spurious transitions. However, it can also *hurt* due to data sparsity: with limited training data, many valid continuations may never appear and thus get zero probability.

Text Classification

(2) In this question we will consider the task of determining if a given sentence is imperative, i.e. is a command (label = 1) or not (label = 0). For example, the sentence "*Alexa, turn off the lights.*" is imperative, but the sentence "*The lights are turned off.*" is not. We want to perform binary logistic regression to complete this task. Assume we have selected the feature vector $< f_0, f_1, f_2, f_3, f_4 >$, with the features defined below:

Feature w	Description c	Numerical Value
f_0	infinitive verb at beginning of sentence	1 if true, else 0
f_1	subject absent	1 if true, else 0
f_2	object after verb	1 if true, else 0
f_3	sentence length	# of words
f_4	ends with a ! or .	1 if true, else 0

(a) (4 points) Consider the sentence "Please submit your homework by Friday, February 20th". With the above model, the extracted feature vector is $< 0, 1, 1, 8, 1 >$. Compute the probability that our binary logistic regression model will classify this sentence as imperative, given the weight vector $< 0.9, 1.1, 0.6, -0.25, 0.2 >$. With a threshold of 0.5, what does the model classify the sentence as? Show all work.

(Solution)

$$z = 0.9(0) + 1.1(1) + 0.6(1) + (-0.25)(8) + 0.2(1) = 0 + 1.1 + 0.6 - 2 + 0.2 = -0.1$$

$$P(y = 1 \mid \mathbf{x}) = \sigma(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{0.1}} \approx 0.475$$

$$0.475 < 0.5$$

not imperative

(b) (5 points) Design a feature so as to augment the current feature set with an intuitively "informative" feature. Derive the new feature vector $< 0, 1, 1, 8, 1, x >$ for the above sentence in part (a). Recompute the probability you found in part **(a)** with the additional feature now assigned a weight 1.0. If needed, what changes would you make to your additional feature to ensure that this classification is correct.

(Solution)

$$f_5 = \begin{cases} 1, & \text{if the sentence contains "please"} \\ 0, & \text{otherwise} \end{cases}$$

Then the feature vector becomes $\langle 0, 1, 1, 8, 1, 1 \rangle$ and the weight vector becomes $\langle 0.9, 1.1, 0.6, -0.25, 0.2, 1.0 \rangle$.

$$z = 0.9(0) + 1.1(1) + 0.6(1) + (-0.25)(8) + 0.2(1) + 1.0(1) = 0 + 1.1 + 0.6 - 2 + 0.2 + 1 = 0.9$$

$$P(y = 1 \mid \mathbf{x}) = \sigma(z) = \frac{1}{1 + e^{-0.9}} \approx 0.711$$

$0.711 > 0.5$, the sentence now becomes imperative.

(c) (4 points) You decide to train your model using stochastic gradient descent. You augment the feature vector with a new feature f_5 = the sentence contains "Please". After training on many examples and over many epochs, your learned weights are $< -0.3, 2.8, 1.9, -0.18, 0.4, 0.08 >$. You find it odd that the "Please" weight is so low, so you add another training example:

sentence w	feature vector	true label	learning rate
"Please close the window later."	$< 0, 1, 1, 5, 1, 1 >$	1 (is imperative)	$\alpha = 0.1$

Table 1: Training Data

Compute the gradient for the weight associated with f_5 after one step of training on the example in Table 1 and using the learning rate $\alpha = 0.1$. Recall that for logistic regression:

$$\frac{\partial L_j}{\partial w_i} = f_i^j \left[\sigma\left(\sum_k w_k f_k^j \right) - y_j \right]$$

(Solution)

$$z = (-0.3)(0) + (2.8)(1) + (1.9)(1) + (-0.18)(5) + (0.4)(1) + (0.08)(1) = 4.28$$

$$\sigma(4.28) = \frac{1}{1 + e^{-4.28}} \approx 0.9864$$

$$\frac{\partial L}{\partial w_5} = f_5\big(\sigma(z) - y\big) = 1 \cdot (0.9864 - 1) = -0.0136$$

(d) (2 point) What could be one explanation for why the weight for the contains "Please" feature remained close to 0 even with many examples? Provide an example that supports your reasoning.

(Solution) One explanation could be that the word "please" is not predictive by itself because it appears in both imperative and non-imperative sentences at similar rates. When a feature shows up in positive and negative examples, its gradient updates tend to cancel out over many SGD steps, so its weight stays near 0. Part b and c in this question have demonstrated the imperative examples, and a non-imperative example could be "Could you please close the door?"

Word Embeddings

(3) You are given the following data (Table 2) to train a **skip-gram model** with negative sampling. Let C and W be the context and word embedding matrices respectively, where row c_t and w_t denote the context and word vector for token t.

Target word w	Context Word c	label y
ball	beach	+
ball	dog	+
cafe	cat	+
park	cafe	-
ball	cat	-
cat	dog	+
cat	ball	-

Table 2: Training Data

(a) (4 points) After training on multiple epochs on the above training data, which row vectors in C and W will have changed the most from their initial values? Explain your reasoning

(Solution) From Table 2, the positive and negative training pairs are:

Positive pairs:

- (ball, beach), (ball, dog), (cafe, cat), (cat, dog)

Negative pairs:

- (park, cafe), (ball, cat), (cat, ball)

We count how many times each word appears as a target and as a context.

Target words:

- ball: 3, cat: 2, cafe: 1, park: 1

Context words:

- beach: 1, dog: 2, cat: 2, ball: 1, cafe: 1

Words that appear more frequently are updated more often and thus change more during training. In addition, words that appear in both positive and negative pairs receive conflicting updates.

For example:

- ball appears in both positive and negative pairs.
- cat also appears in both positive and negative pairs.

Therefore, the vectors for **ball** and **cat** in both W and C are expected to change the most.

(b) (2 points) In the training data, notice that 'cat' appears as a context word in multiple examples with different labels. What is the model expected to learn about the relationships between these words?

(Solution) In the training data, "cat" appears as a context word in both positive and negative examples.

For example:

- (cafe, cat, +) encourages w_{cafe} and c_{cat} to be similar.
- (ball, cat, -) encourages w_{ball} and c_{cat} to be dissimilar.

As a result, the model is expected to learn that "cat" is related to "cafe" but unrelated to "ball". The context vector for "cat" will be positioned close to the target vector of "cafe" and far from that of "ball".

Section B: Programming (100 points)

You will essentially follow the `hw1_studentfacing_sp26_cs4740.ipynb` file in this google drive folder to complete this coding assignment. Please read the description below to understand the task setup.

In this programming assignment, you will implement Binary Logistic Regression from scratch for the natural language entailment task. Given a sentence pair (P, H) where P is a premise and H is a hypothesis, we are aiming to classify the relation between them. For this assignment, we will only consider two labels, **entailment**, meaning the hypothesis must be true if the premise is true, and **contradiction**, meaning the hypothesis cannot be true if the premise is true.

(Solution)

Part 1: Data Exploration

Q1.1

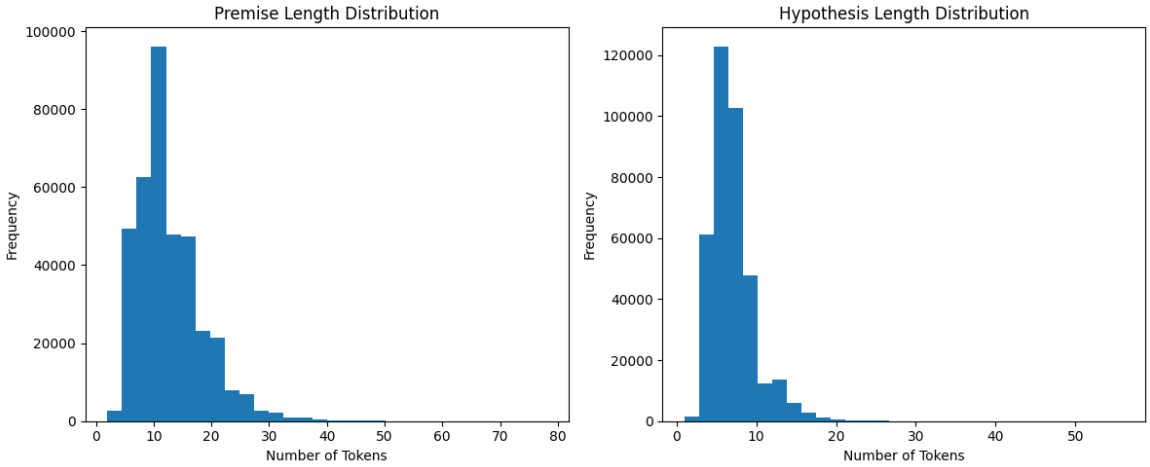

Figure 1: Premise and Hypothesis Length Distribution

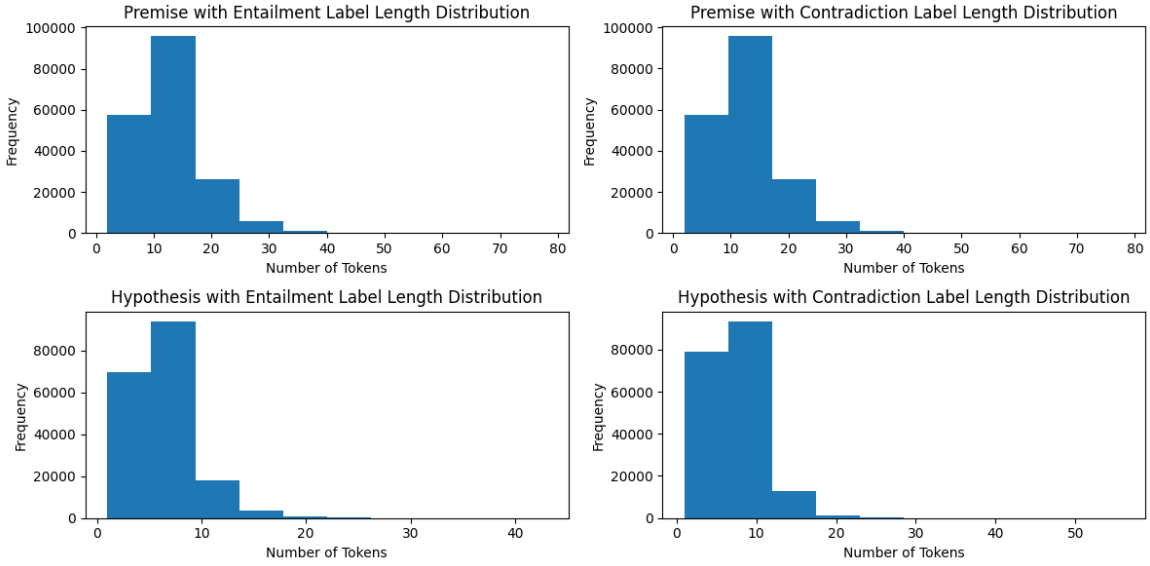

Figure 2: Premise and Hypothesis Length Distribution with Different Labels

Observations.
From the histograms, we observe that:

- Most premises contain around 5–15 tokens, while most hypotheses contain around 5–10 tokens.
- When comparing entailment and contradiction labels for premises, their length distributions are very similar. This suggests that the length of the premise does not strongly correlate with the labels.
- In contrast, for hypotheses, the length distributions show more noticeable differences between entailment and contradiction. This suggests that hypothesis length may be more correlated with the labels.

Q1.2
From the frequency analysis, we observe that:

- The most frequent unigrams are mainly function words, such as “the”, “a”, “is”, “and”, and “in”.
- The most frequent bigrams are common syntactic patterns, such as “a man”, “in a”, “on a”, “in the”, and “a woman”.
- High-frequency tokens are dominated by stop words and grammatical structures, which carry limited semantic information.

- Relying solely on raw frequency features may not be sufficient for capturing meaningful semantic relationships.

Q2.1

- Many high-frequency function words such as "a", "the", and "is", as well as the `<SEP>` token, appear frequently in unigram features. For example, in Example 0, the word "a" appears 5 times and "person" appears twice. This shows that the representation is dominated by common words, which may reduce the impact of content-specific terms.
- Bigram features capture short phrases such as "a person", "the boy", and "is jumping", which provide more contextual information than unigrams. For instance, in Example 4, bigrams like "the boy" and "is jumping" appear in the feature vector. However, bigrams are much sparser, and many of them occur only once, making the feature space highly sparse.

Submission

We will create 3 different Gradescope assignments for this homework. Below we outline submission details for each.

2. Assignment `hw1-partnerdeclaration`

Due date: February 16, 2026. 11.59 p.m.
Ungraded but mandatory!! Please declare if you plan to submit this assignment with a partner or by yourself. If former, declare your partner's NetID.

2. Assignment `hw1-written`

Due date: February 21, 2026. 11.59 p.m.
Max points: 40

This assignment is for you to submit the written parts of this homework. This includes: (1) Solutions to Section A, (2) Solutions to the written questions in Section B. Specifically, this means questions Q1 (Q1.1 - Q1.2), Q2 (Q2.1), Q3 (Q3.1, Q3.2, Q3.3 and Q3.4) and Q4 (Q4.1).

3. Assignment `hw1-programming`

Due date: February 21, 2026. 11.59 p.m.
Max points: 100

You will follow the instructions in the `hw1_studentfacing_sp25_cs4740.ipynb` in the google drive. It will ask you to implement functions in 2 different python files (also included in the google drive): *features.py* and *models.py*. You will submit these 2 Python files to `hw1-programming`.

Please double check to make sure that all the functions that you have implemented al functions that need to be implemented in these files.