Instructions

1. This homework has two parts: Q1, Q2 and Q3 are written questions and Q4 is a programming assignment with some parts requiring a written answer. Each part needs to be submitted as follows:

   - Submit the answers to the written questions as a `pdf` file on Canvas for the assignment corresponding to Homework 3 Written. This should consist answers to Q1, Q2, Q3 and the descriptive answers from Q4. Name the pdf file as- `LastName_FirstName.pdf`. We recommend students type answers with LaTeX or word processors for this part. A scanned handwritten copy would also be accepted, try to be clear as much as possible. No credit may be given to unreadable handwriting.
   - The programming assignment requires you to work on boilerplate code. Submit the code to the programming assignment in a zip that contain `ngram.ipynb`, `rnn.ipynb` `hw3_skeleton_char.py` and `hw3_skeleton_word.py`. This submission is to be made on Canvas for the assignment corresponding to Homework 3 Programming. Name the zip file as- `LastName_FirstName.zip`.
   - You first need to join this course on Gradescope. Please use your actual name as Gradescope username. Entry Code for CS 4650 is `9P32V6`. Entry Code for CS 7650 is `97ZWYV`. You should also submit a zip file to Gradescope hw3 Language Models. This file should contain `hw3_skeleton_char.py` and `hw3_skeleton_word.py`. You can see your score for 4 (a) as soon as you submit it.

2. For the written questions, write out all steps required to find the solutions so that partial credit may be awarded.

3. We generally encourage collaboration with other students. You may discuss the questions and potential directions for solving them with another student. However, you need to write your own solutions and code separately, and not as a group activity. Please list the students you collaborated with.

4. The code files needed to complete the homework are included in a zip file on Canvas.

1. The Kneser-Ney smoothing method approximates the probability of an n-gram that has not been seen using the likelihood of the (n-1)-gram to occur in diverse contexts. If we want to finish the sentence "I want to go to the movie _____" with the word "theatre" but we have not observed "theatre" very often, then we want to make sure that we can guess "theatre" based on its likelihood of combining with other words.

   The full formula for the bigram version of Kneser-Ney smoothing follows:

$$P(bigram) = \text{discounted bigram probability} + \text{joint unigram probability} \tag{1}$$

$$P(w_i|w_{i-1}) = \frac{max(C(w_{i-1}, w_i) - d, 0)}{C(w_{i-1})} + \lambda(w_{i-1}) \frac{|v : C(v, w_i) > 0|}{\sum_{w'} |v : C(v, w') > 0|} \tag{2}$$

$$\lambda(w_{i-1}) = \frac{d}{\sum_v C(w_{i-1}, v)} * |w : C(w_{i-1}w) > 0| \tag{3}$$

   Assume that you have collected the data in the following tables, and assume that all other observed counts are 0. In the bigram table, rows represent w_{i-1}, columns represent w_i: e.g. $C(\text{computer}, \text{keyboard}) = 2$.

$C(w_{i-1}, w_i)$	computer	keyboard	monitor	store
computer	0	2	4	4
keyboard	1	0	0	1
monitor	0	1	1	1
store	2	0	0	0

Table 1: Bigram frequency. Rows = w_{i-1}, columns = w_i.

computer	10
keyboard	3
monitor	6
store	5

Table 2: Unigram frequency.

   Consider the following sentence fragment S: "I shopped at the computer _____".
   You need to determine whether the sentence is more likely to end with "computer store" or "computer monitor."

   (a) Compute the raw bigram probabilities for the candidate words {*store*, *monitor*} to complete the sentence S, i.e. $P(\text{store}|\text{computer})$ and $P(\text{monitor}|\text{computer})$. Is one word more likely than the other, and if so which one? [2 pts]

   (b) Compute the Kneser-Ney smoothed bigram probability of the candidate words {*store*, *monitor*} to complete the sentence. Use $d = 0.5$ as the discount term. Is one word more likely than the other, and if so which one? If the result has changed, why do you think it changed? [5 pts]

(c) Change the discount term to $d = 0.1$ and re-compute the Kneser-Ney smoothed bigram probability of the candidate words $\{$*store*, *monitor*$\}$ to complete the sentence. Is one word more likely than the other, and if so which one? If the result has changed, why do you think it changed? [3 pts]

Solution.
Solution goes here. ■

2. Consider we have a term-document matrix for four words in three documents shown in Table 3. The whole document set has $N = 20$ documents, and for each of the four words, the document frequency df_t is shown in Table 4.

term-document	Doc1	Doc2	Doc3
car	27	4	24
insurance	3	18	0
auto	0	33	29
best	14	0	17

Table 3: Term-document Matrix

	df
car	12
insurance	6
auto	10
best	16

Table 4: Document Frequency

(a) Compute the *tf-idf* weights for each word car, auto, insurance and best in Doc1, Doc2, and Doc3. [6 pts]

(b) Use the *tf-idf* weight you get from (a) to represent each document with a vector and calculate the cosine similarities between these three documents. [4 pts]

Solution.
Solution goes here. ■

3. The distributional hypothesis suggests that the more similarity there is in the meaning of two words, the more distributionally similar they are, where a word's distibution refers to the context in which it appears. This motivated the work by Mikolov et al. on the skip-gram model which is an efficient way of learning high quality dense vector representations of words from unstructured text. The objective of the skip-gram model is to learn the probability distribution $P(O|I)$ where given an inside word w_I, we intend to estimate the probability that an outside word w_O lies in the context window of w_I. The basic formulation of the skip-gram model defines this using the softmax function:

$$P(O = w_O|I = w_I) = \frac{\exp(\boldsymbol{u}_{\boldsymbol{w_O}}^T.\boldsymbol{v}_{\boldsymbol{w_I}})}{\sum_{w\in\text{Vocab}} \exp(\boldsymbol{u}_{\boldsymbol{w}}^T.\boldsymbol{v}_{\boldsymbol{w_I}})} \tag{4}$$

Here, $\boldsymbol{u}_{\boldsymbol{w_O}}$ is the word vector representing the outside word o and $\boldsymbol{v}_{\boldsymbol{w_I}}$ is the word vector representing the inside word i. To update these parameters continually during training, we store these in two matrices $\mathbf{U}$ and $\mathbf{V}$. The columns of $\mathbf{V}$ are all of the inside word vectors $\boldsymbol{v}_{\boldsymbol{w_I}}$ while the columns of $\mathbf{U}$ are all the outside word vectors $\boldsymbol{u}_{\boldsymbol{w_O}}$ and both these matrices contain a vector for each word in the vocabulary.

(a) The cross entropy loss between two probability distributions p and q, is expressed as:

$$CE(p, q) = -\sum_i p_i \log(q_i) \tag{5}$$

For, a given inside word $w_I = w_k$, if we consider the ground truth distribution $\boldsymbol{y}$ to be a one-hot vector (of length same as the size of vocabulary) with a 1 only for the true outside word w_O and 0 everywhere else. The predicted distribution $\hat{\boldsymbol{y}}$ (of length same as the size of vocabulary) is the probability distribution $P(w_O|w_I = w_k)$. The i^{th} entry in these vectors is the probability of the i^{th} word being an outside word. Write down and simplify the expression for the cross entropy loss, $CE(\boldsymbol{y}, \hat{\boldsymbol{y}})$, for the skip-gram model described above for a single pair of words w_O and w_I. (Note: your answer should be in terms of $P(O = w_O|I = w_I)$.) [2 pts]

(b) Find the partial derivative of the cross entropy loss calculated in part (a) with respect to the inside word vector $\boldsymbol{v}_{\boldsymbol{w_I}}$. (Note: your answer should be in terms of $\boldsymbol{y}$, $\hat{\boldsymbol{y}}$ and $\mathbf{U}$.) [5 pts]

(c) Find the partial derivative of the cross entropy loss calculated in part (a) with respect to each of the outside word vectors $\boldsymbol{u}_{\boldsymbol{w_O}}$. (Note: Do this for both cases $w_O = O$ (true outside word) and $w_O \neq O$ (all other words). Your answer should be in terms of $\boldsymbol{y}$, $\hat{\boldsymbol{y}}$ and $\boldsymbol{v}_{\boldsymbol{w_I}}$.) [5 pts]

(d) Explain the idea of negative sampling and the use of the parameter K. Write down the loss function for this case. (Note: your answer should be in terms of $\boldsymbol{u}_{\boldsymbol{w_O}}$, $\boldsymbol{v}_{\boldsymbol{w_I}}$ and the parameter K.) [3 pts]

Solution.
Solution goes here. ■

4. In the textbook, language modeling was defined as the task of predicting the next word in a sequence given the previous words. In this assignment, you will implement character-level, word-level N-gram and character-level RNN language models. You need to both answer the questions and submit your code. You need to submit a zip file to Canvas Homework 3 Programming. This file should contain `ngram.ipynb`, `rnn.ipynb` `hw3_skeleton_char.py` and `hw3_skeleton_word.py`.

   You should also submit a zip file to the Gradescope assignment HW3 Language Models. This file should contain `hw3_skeleton_char.py` and `hw3_skeleton_word.py`.

   (a) For N-gram language models, You should complete two scripts `hw3_skeleton_char.py` and `hw3_skeleton_word.py`. Detailed instructions can be found in `ngram.ipynb`. You should also use test cases in `ngram.ipynb` and use `ngram.ipynb` to get development results for (c).

   You need to submit a zip file to Gradescope HW3 Language Models. This file should contain `hw3_skeleton_char.py` and `hw3_skeleton_word.py`. You can see the scores for your code there. Character-level N-gram language models accounts for 20 points. Word-level N-gram language models accounts for 10 points, which are bonus for CS 4650. [30pts for CS 7650, 20 pts + bonus 10pts for CS 4650]

   (b) See the generation results of your character-level and word-level N-gram language models respectively ($n \geq 1$). The paragraphs which character-level N-gram language models generate all start with *F*. The paragraphs which word-level N-gram language models generate all start with *In*. Did you get such results? Explain what is going on. (CS 4650 can only focus on character-level N-gram language model.) [2 pts]

   (c) (Bonus for CS 4650) Compare the generation results of character-level and word-level N-gram language models. Which do you think is better? Compare the perplexity of `nytimes_article.txt` when using character-level and word-level N-gram language models. Explain what you found. [2pts, bonus for CS 4650]

   (d) When you compute perplexity, you can play with different sets of hyper-parameters in both character-level and word-level N-gram language models. You can tune n , k and λ. Please report here the best results and the corresponding hyper-parameters in development sets. For character-level N-gram language models, the development set is `shakespeare_sonnets.txt`. For word-level N-gram language models, the development sets are `shakespeare_sonnets.txt` and `val_e.txt`. (CS 4650 should only focus on character-level N-gram language model.) [6 pts for CS 7650, 2 pts + bonus 4 pts for CS 4650]

   (e) For RNN language models, You should complete the forward method of Class RNN in `rnn.ipynb`. You need to figure out the code and tune the hyperparameters. You should also copy a paragraph generated by your model and report the perplexity on the development set `shakespeare_sonnets.txt`. Compare the results of character-level RNN language model and character-level N-gram language model. [10 pts]

Solution.
Solution goes here. ■