Instructions:

1. The exam is consisting of two parts, 1) two problem set with sub questions, and 2) two programming problem with some parts requiring a written answer.
2. Graduate students should answer and solve all problem, while undergraduate students can skip sub problems marked as **[Graduate only]**.
3. Each problem must be submitted in a separate file or folder, for example problem set 1 answer will be in a separate pdf file and so on. Note: you should submit 2 pdf files and 2 zip folders corresponding to each problem. The file names should be as follows:
    a. Problem set 1: "yourlastname_ps1.pdf"
    b. Problem set 2: "yourlastname_ps2.pdf"
    c. Programming Problem 1: "yourlastname_pp1.zip"
    d. Programming Problem 2: "yourlastname_pp2.zip"

    Details for Programming Problem 1 and 2 files submissions are descried below.
4. LATEX solutions are strongly encouraged, but scanned handwritten copies are also acceptable. Hard copies are not accepted.
5. Problem Set 1 and 2 each worth 15 points, Programming Problem 1 is worth 25 points, and Programming Problem 2 is worth 45 point. The total worth of this exam is 100 points.
6. 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 on the submission site.**
7. Late submissions are not allowed. The exam is due **Monday March 18 @ 11:59 pm.**
8. Note: Failure to follow these instructions may result in parts of your Exam not being graded. We will not entertain regrading requests for failure to follow instructions.

We wish you all the best, and enjoy the diving into the NLP world!

Problem Set 1: [15 points]

A collection of movie reviews (data $\mathcal{D}$) contains the following keywords and binary labels for whether each review was positive (+) or negative (−). The data is shown below: for example, the cell at the intersection of "Review 1" and "epic" indicates that the text of Review 1 contains 2 tokens of the word "epic". Answer the following questions, reporting all scores as log-probabilities, to three significant figures.

Review	great	amazing	epic	boring	terrible	disappointing	Y
1	2	2	2	1	1	0	+
2	1	4	0	0	0	1	+
3	3	2	3	1	0	0	+
4	0	1	1	2	1	2	-
5	1	0	2	1	2	2	-
6	1	0	1	2	1	2	-

Table 1: Movie reviews keywords frequency and binary labels

a) Assume that you have trained a Naive Bayes model on data $\mathcal{D}$ to detect positive vs. negative movie reviews. Compute the model's predicted scores for both positive and negative classes for the following sentence S (i.e. $P(+|S)$ and $P(-|S)$), and determine which label the model will apply to S.

   S: The film was great, the plot was simply amazing! Makes other superhero movies look terrible, this was not disappointing.

b) The counts in the original data are sparse and may lead to overfitting, e.g. a strong prior on problem the negative label to reviews that contain "terrible". What would happen if you applied smoothing? Apply add-1 smoothing and recompute the Naive Bayes model's predicted scores for S. Did the predicted label change?

c) What is an additional feature that you could extract from text to improve the classification of sentences like S (not necessarily in NB), and how would it help improve the classification?

Generate one PDF file of your answers, and uploaded to Canvas. File name should be "yourlastname_ps1"

Problem Set 2: [15 points]

You find yourself eating popcorn and watching The Office after your NLP course. You notice that Michael, Jim and Dwight use very distinctive language from one another! and get to wondering whether you could apply your text classification knowledge to build a multi-class classifier. You collect the dataset of all transcripts from The Office and utilize regex to isolate dialogues by Michael, Jim and Dwight. You build and train a simple multinomial logistic regression using bag-of-words input representation (not from scratch, you utilize Sci-Kit Learn because you'd still like to go back to streaming). Here is the confusion matrix obtained after running the classifier on your held-out test set:

		True Labels		
		Michael	Jim	Dwight
Predicted Labels	Michael	8200	300	100
	Jim	800	4200	200
	Dwight	500	500	3500

Table 2: The Office Confusion Matrix

a) Calculate the precision and recall for all three classes.
b) Calculate the macroF1 score.
c) Calculate the microF1 score. Explain why it looks different from macroF1.
d) You want to improve the classifier using a perceptron classifier. You decide to go with one hidden layer. However, you're unsure about what activation functions to use. The default option is ReLU but you are unsure. What are the advantages and disadvantages of using Sigmoid or Tanh activations? Which activation function should you use and why?
e) **[Graduate only]** What is Leaky ReLU? Would you use it over the default option (ReLU)? Why or why not?

Generate one PDF file of your answers, and uploaded to Canvas. File name should be "yourlastname_ps2"

Programming Problem 1: [25 points]

In this problem, you will implement some introductory DataLoaders and Models using PyTorch, a popular deep-learning library. Specifically, you'll be writing code to classify if headlines come from The Onion (a fake satirical newspaper) or are actual headlines (binary classification). We'll be using this dataset, included in the pp1.zip file.

Sometimes, telling the differences can be tricky. Here are two examples:

a) **From The Onion:**
   Entire Facebook Staff Laughs As Man Tightens Privacy Settings
b) **Real Headline:**
   Cyclist's Bike Stolen at Police Station while Reporting iPhone Theft

The notebook: 'pp1.ipynb' will guide you through the overall workflow for the problem, from loading and reshaping data to implementing a model and executing a training/testing loop. We strongly recommend using Google Colab to import the pp1.ipynb notebook, and following the instructions from there!

Finally, you should respond to the written questions in the analysis section of the notebook, and you should attach a PDF of your notebook output to your submission.

a) Run the notebook 'pp1.ipynb' and follow-along, implementing the necessary functions in neighboring files as the script calls for them. To run the notebook locally (only if you don't want to use Colab), navigate to the 'pp1' directory in your terminal and run 'jupyter notebook'.
b) Complete the analysis questions at the end of the notebook.
   i. What happens to the vocab size as you limit words by their frequency? Can you explain this in the context of Zipf's Law?
   ii. Can you qualitatively describe (1 paragraph) what cases the model is getting wrong in the witheld test-set (see notebook for details)?

Zip all your files 'src' folder with all subfiles (dataset.py, eval_utils.py, models.py, and preprocess.py), 'pp1.ipynb' and PDF output of you notebook, and uploaded to Canvas. File name should be "yourlastname_pp1"

Programming Problem 2: [45 points]

In this problem, we will be exploring word embeddings and language modeling. Start by downloading pp2.zip file. For all of these notebooks, you will need to export the PDF outputs and concatenate them to your writing portion.

a) word embeddings.ipynb:

We'll start by looking at word2vec, a technique to generate word vectors. You will not be training your own word embeddings: instead you would be using pre-trained word embeddings from GenSim. This problem is designed to provide you a better understanding of the vector space the word embeddings lie in. Specifically, you will be looking at similarities, semantics, analogies, biases and visualization. Download and complete the notebook, following the instructions provided therein.

b) ngram.ipynb:

i) Complete 'pp2 skeleton char.py.' Detailed instructions can be found in the 'ngram.ipynb' file in pp2.zip. You should also use test cases in 'ngram.ipynb' to get development results for parts (iii) and (iv) of this subproblem.

ii) **[Graduate only],** complete 'pp2 skeleton word.py,' along with the corresponding cells for word-based LMs in 'ngram.ipynb'

iii) Observe the generation results of your character-level n-gram language models ($n \geq 1$). The paragraphs which character-level n-gram language models generate all start with F. Did you get such results? Explain what is going on.

iv) **[Graduate only]** Compare the generation results of character-level and word level n-gram language models. Which do you think is better? Compare the perplexity of 'shakespeare sonnets.txt' when using character-level and word level n-gram language models. Explain what you found.

v) 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'.

vi) For word-level n-gram language models, the development sets are 'shakespeare sonnets.txt' and 'val e.txt'.

c) rnn.ipynb:

i) For RNN language models, you should complete the forward method of the RNN class in rnn.ipynb.

- Figure out the forward pass and tune hyperparameters,
- Copy a paragraph generated by your model here,
- Report hyperparameters and perplexity on the development set 'shakespeare sonnets.txt' here, and
- Compare the results of character-level RNN language model and character-level n-gram language model here.
- **[Graduate only]** Imagine you designed an RNN trained on word-level tokens from a corpus similar to Gensim's. How would the RNN's hidden state for a word x be similar to Gensim's word vectors? How would it be different?

Zip your file 'word embedding.ipynb,' 'ngram.ipynb', 'rnn.ipynb', and skeleton files, with PDF output of you notebook, and uploaded to Canvas. File name should be "yourlastname_pp2"