CSE 151B Deep Learning
UCSD, Spring 2025

Homework 3

Due 11:59 p.m, May 20th, 2025

Please submit your written solutions as a single PDF file using the provided LaTeX template on the course website. (https://sites.google.com/view/cse151b-251b). You can use Overleaf (https://www.overleaf.com/) to compile LaTex files online. For problems requiring code, additionally, submit all necessary code in one zip file. Code must run and produce the results reported in the PDF for full credit.

1 Sequence Processing [3 points]

Given binary sequences, implement binary addition. The sequences start with the least significant binary digit. (It is easier to start from the least significant bit, just like how you did addition in school.) The sequences will be padded with at least one zero on the end. For instance, the problem [100111 + 110010 = 1011001] will be represented as

- Input 1: 1, 1, 1, 0, 0, 1, 0
- Input 2: 0, 1, 0, 0, 1, 1, 0
- Correct output: 1, 0, 0, 1, 1, 0, 1

There are two input units corresponding to the two inputs, and one output unit. Design the weights and biases for an RNN which has two input units, three hidden units, and one output unit, which implements binary addition. All of the units use the hard threshold activation function ($f(x) = 1$ if $x > 0$ and 0 otherwise). In particular, specify weight matrices U, V, and W, bias vector b_h, and scalar bias b_y for the architecture in Figure 1.

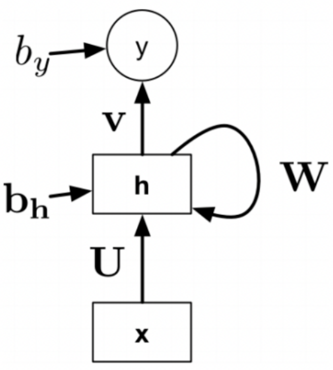

Figure 1: Architecture

2 Recurrent Neural Networks [7 Points]

William Shakespeare is perhaps the most famous poet and playwright of all time. Shakespeare is known for works such as Hamlet and his 154 sonnets, of which the most famous begins:

Shall I compare thee to a summer's day?
Thou art more lovely and more temperate:

Shakespeare's poems are nice for deep learning because they follow a specific format, known as the Shakespearean (or English) sonnet.[1] Each sonnet is 14 lines, spread into 3 quatrains (section with 4 lines) followed by a couplet (section with 2 lines). The third quatrain is known as the *volta*[2] and has a change in tone or content. Shakespearean sonnets have a particular rhyme scheme, which is *abab cdcd efef gg*.

Shakespearean sonnets also follow a specific meter called *iambic pentameter*[3]. All lines are exactly 10 syllables long, and have a pattern of unstressed stress. For example, the famous Sonnet 22 begins:

Stress	x	\	x	\	x	\	x	\	x	\
Syllable	Shall	I	com -	pare	thee	to	a	sum-	mer's	day?

Here, each x represents an unstressed syllable and every \represents a stressed syllable. Try saying it out loud!

The goal for this assignment is to generate poems that Shakespeare may have written by training a recurrent neural network on his 154 sonnets. His sonnets are available in the data file `shakespeare.txt`, provided in the data folder. Refer to the lecture/discussion material, and read the sequence modeling chapter (Chapter 10) of the Deep Learning book for more details of RNNs.

Problem A [2 points]: Pre-processing

The first step is to pre-process the `poem_data` dataset before you train on it. How you pre-process is completely up to you. Here are a couple of questions to help you decide how to do pre-processing: How will you tokenize the data set? What will consist of a singular sequence, a poem, a stanza, or a line? Do you keep some words tokenized as bigrams? Do you split hyphenated words? How will you handle punctuation? It may be helpful to get syllable counts and syllable stress information from CMU's Pronouncing Dictionary available on NLTK. You might also find the file `Syllable_dictionary.txt`, provided in the data folder, to be helpful; please see the associated file called "syllable_dict_explanation" for an explanation.

Explain your choices, as well as why you chose these choices initially. What was your final

[1] https://en.wikipedia.org/wiki/Sonnet#English_.28Shakespearean.29_sonnet
[2] https://en.wikipedia.org/wiki/Volta_%28literature%29
[3] https://en.wikipedia.org/wiki/Iambic_pentameter

pre-processing? How did you tokenize your words, and split up the data into separate sequences? What changed as you continued on your project? What did you try that didn't work? Also write about any analysis you did on the dataset to help you make these decisions.

Problem B [3 points]: Model Training

Try doing poem generation using a recurrent neural network (RNN). Start with the Pytorch Tutorial Notebook. Please follow these guidelines in this section:

- Train a **character-based LSTM** model. A single layer of 100-200 LSTM units should be sufficient. You should also have a standard fully-connected output layer with a softmax nonlinearity.
- Train your model to minimize categorical cross-entropy. Make sure that you train for a sufficient number of epochs so that your loss converges. You don't necessarily need to keep track of overfitting/keep a validation set.
- Your training data should consist of sequences of fixed length (40 characters is a good number for this task) drawn from the sonnet corpus. The densest way to do this is to take all possible subsequences of 40 consecutive characters from the dataset. To speed up training, using *semi-redundant* sequences (i.e. picking only sequences starting every n-th character) works just as well.
- To generate poems, draw softmax samples from your trained model. Try to lay around with the *temperature* parameter and generate different outputs, which controls the variance of your sampled text.

Explain in detail what model you implemented? What parameters did you tune? Comment on the poems that your model produced. Does the LSTM successfully learn sentence structure and/or sonnet structure? How about the runtime/amount of training data needed? Include generated poems using temperatures of 1.5, 0.75, and 0.25 with the following initial 40-character seed: "shall i compare thee to a summer's day?\n", and comment on their differences.

Problem C [2 points]: Improving recurrent models

List and explain two potential advantages and two potential disadvantages of using a character-level recurrent neural network, as opposed to a word-level recurrent neural network.

Additional Resources

- TED talk: Can a computer write poetry?
- Natural Language Processing Toolbox
- Markov Contraints for Generating Lyrics with Style