CSE 151B/251B Deep Learning
UC San Diego, Spring 2025

Homework 3

Name: Ouyang Yingxuan UID: 10486415

1 Problem 1 Sequence Processing

Solution :

We design an RNN with two input units, three hidden units, and one output unit. All units use a hard threshold activation function:

$$f(x) = \begin{cases} 1 & \textit{if } x > 0 \\ 0 & \textit{otherwise} \end{cases}$$

The RNN architecture follows the diagram in Figure 1. We specify:

Input: $x_t = [x_{1t}, x_{2t}]^T \in \{0,1\}^2$

Hidden state: $h_t \in \{0,1\}^3$

Output: $y_t \in \{0,1\}$

The update equations are:

$$h_t = f(Ux_t + Wh_{t-1} + b_h), \quad y_t = f(Vh_t + b_y)$$

Weight Matrices and Biases are:

$$U = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}, \quad W = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad b_h = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

$$V = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}, \quad b_y = 0$$

In the RNN network, the first two hidden units compute a partial sum and carry logic. The third hidden unit is used to store the carry from the previous step. The output uses the second and third hidden units to compute the final bit.

2 Problem 2 Recurrent Neural Networks

Problem A: Data Preprocessing for Poem Modeling

Solution A: *We preprocess the poem corpus to train a language model capable of generating coherent poetic lines. Initially, we explored several tokenization and cleaning strategies, evaluated their impact, and finally adopted the following approach.*

Tokenization Strategy: *We chose* ***word-level tokenization****, where each word is treated as a token. This helps the model capture semantic and grammatical structure better than character-level tokenization, while avoiding data sparsity of higher-level structures like bigrams.*

Handling Hyphenated Words: *Words with hyphens (e.g.,* `dead-doing`*) were treated as single tokens. This is consistent with the format in* `Syllable_dictionary.txt`*, which considers hyphenated forms as one word. This preserves syllabic rhythm and stress patterns.*

Punctuation Removal: *We removed most punctuation characters (e.g., comma, period, semicolon), but preserved apostrophes within words (e.g.,* `th'` *or* `ne'er`*) to maintain consistency with the CMU dictionary and poetic style. This helps reduce vocabulary size while retaining poetic contractions.*

Syllable Dictionary Integration: *We used the provided* `Syllable_dictionary.txt` *to extract syllable counts for each token. If a word had multiple valid syllable counts (e.g.,* `being 1 2`*), we stored all options for flexible future use. For out-of-vocabulary (OOV) words, we attempted to infer syllables using heuristics (e.g., vowel cluster counts).*

Failed Attempts: *Initially, we attempted bigram tokenization, but it caused data sparsity and reduced generalization. We also tried lowercasing all text, but this degraded poetic style and rhyming fidelity, so we retained capitalization.*

Final Preprocessing Pipeline: *1.Load text from* `shakespeare.txt` *and* `spenser.txt`*.*

2.Normalize hyphenated words (preserve as one token).

3.Remove extraneous punctuation while preserving in-word apostrophes.

4.Tokenize by whitespace to form word-level sequences.

5.Lookup or estimate syllable counts from `Syllable_dictionary.txt`.

6.Segment data into lines and sequences for training (e.g., line-by-line, stanza-by-stanza).

This approach balances rhythm, structure, and vocabulary efficiency. It provides a good basis for syllabic or rhyming constraints in future training.

Problem B: Model Training

Solution B:

The model implemented is a character-level Long Short-Term Memory (LSTM) neural network designed for text generation. Specifically, the model attempts to learn the structure and style of Shakespearean sonnets. The LSTM architecture is well-suited for sequential data due to its ability to retain long-term dependencies, making it a popular choice for tasks involving natural language processing.

Several parameters were tuned in this experiment:

Number of epochs: *40*

Training time: *Approximately 15 minutes*

Temperature values during generation: *1.5, 0.75, and 0.25*

Seed text: `"SHALL I COMPARE THEE TO A SUMMER'S DAY?\N"`

3. Poem Generation and Structure Evaluation

The model was evaluated by generating text with three different temperature values using the same initial seed. The generated outputs are compared as follows:

Temperature = 1.5

At a high temperature, the output is highly random and lacks coherence. The characters produced are mostly nonsensical or non-alphabetic:

```
shall i compare thee to a summer's day?  u8)EC(94lWWOTW:ky9;gqTk
MG7nY0R43q ...
```

This result demonstrates that the model outputs diverse but unreadable sequences at high temperatures, suggesting that the structure of the sonnet or language has not been preserved.

Temperature = 0.75

At a moderate temperature, the output shows more structure and contains a few valid English words, although it is still largely incoherent:

```
shall i compare thee to a summer's day?  x0yOf9POJbj;AdeEp.:q.7:Dn1RgA
...)
```

This indicates partial success in learning sentence-level patterns, but not enough to generate poetically meaningful text.

Temperature = 0.25

At a low temperature, the output becomes repetitive and rigid, with some sequences resembling English but overall still lacking sense:

```
shall i compare thee to a summer's day?  5 ...  y!  (hB2'1y
U40 3daHn cEstAny m1 ...
```

This shows that the model is overly conservative and produces less diverse outputs, but it still fails to capture the actual structure of a sonnet.

4. Evaluation of Learning

*Based on the generated samples, the LSTM model did **not successfully learn** the sentence structure or the specific rules of Shakespearean sonnets (e.g., 14 lines, iambic pentameter, or rhyme scheme). The output lacks grammatical structure, semantic coherence, and poetic form. The relatively small training time (15 minutes) and limited number of epochs (40) are likely insufficient for the model to learn the complex dependencies and structure present in Shakespearean poetry.*

Conclusion

The experiment shows the effect of temperature on text generation: higher values produce more diverse but incoherent text, while lower values reduce diversity and may cause repetition. For the model to generate proper sonnets, more training data, longer training time, and architectural improvements (such as attention mechanisms or pre-trained embeddings) may be necessary.

Problem C: Improving recurrent models

Solution C:

Advantages of Character-level RNNs:

1. *Character-level RNNs can naturally handle out-of-vocabulary (OOV) words by generating words character by character, without requiring the word to be in a predefined dictionary.*
2. *Since characters are the most basic unit, the model has more flexibility in producing stylistic features such as alliteration, rhyme, and rhythm—particularly useful in poetry generation.*

Disadvantages of Character-level RNNs:

1. *Character sequences are inherently longer than word sequences, which increases computational burden and makes learning long-term dependencies more difficult.*
2. *Because character-level models must learn both spelling and grammar, they require more training data and time to converge compared to word-level models which operate on higher-level language units.*