Homework 4: Build Your Own Language Model — RNN

CPSC 1710 · Introduction to AI Applications · Fall 2025

This assignment sits at the course's transition from using *AI to* understanding *it. In previous weeks you used AI tools freely. Now we slow down and open the black box—starting with a recurrent neural network small enough to trace by hand.*

Submission

1. A **PDF** uploaded to Canvas, named `[Firstname]_[Lastname]_Homework4_main.pdf`
2. Your **GitHub Classroom repository link**

Total: max 100 points (including bonus questions).

Q1. Concepts

▷ AI Policy: No AI Assistance

Q1 must be completed **without any AI assistance**. Work through the math yourself and show your work. You may run the code to check your answers.

Q1.1 [10 pts] — Trace an RNN by hand

Consider the following RNN step function:

```
def rnn_step(x, hidden_state):
    hidden_state = Wx * x + Wh * hidden_state + b
    return hidden_state

h1 = rnn_step(x1, h0)
h2 = rnn_step(x2, h1)
h3 = rnn_step(x3, h2)
```

Given an RNN with parameters:

$$\texttt{h0 = 5}, \quad \texttt{Wx = 2}, \quad \texttt{Wh = 1}, \quad \texttt{b = -3}$$

and input sequence `[2, 3, 4]`, compute `h1`, `h2`, and `h3`.

Q1.2 [10 pts] — Expand to hidden size 2

Now expand to a hidden state of size 2:

```
import numpy as np
Wx = np.array([2, 1])
Wh = np.array([1, 1])
b = np.array([-3, -3])
```

```
hidden_state = np.array([5, 5])

def rnn_step(x, hidden_state):
    hidden_state = Wx * x + Wh * hidden_state + b
    return hidden_state

x1 = 1
h1 = rnn_step(x1, hidden_state)
print(h1)
x2 = 2
h2 = rnn_step(x2, h1)
print(h2)
```

Calculate `h1` and `h2` by hand (also type it up by hand), showing your work. Run the code to check your answers.

Q1.3 [Bonus: 5 pts] — Fibonacci via RNN

The Fibonacci sequence: $1, 1, 2, 3, 5, 8, 13, \ldots$
Can you fit a model using an architecture similar to Q1.2 but with the *matrix-multiply* version of `rnn_step`?

```
Wh = np.array([[0, 0], [0, 0]])  # initialized with zeros
def rnn_step(x, h):
    hidden_state = Wx * x + Wh @ h + b
    # '@' is matrix-vector multiply
    return hidden_state
```

▷ AI Policy: AI Assistance Permitted

For the rest of this Homework, feel free to **use AI assistance to help you learn**, but for "type" requests we ask you to refrain from directly copying AI content. The goal is understanding, not output.

Q2. A Miniature Text Generator

Q2 is one coherent arc: run the starter code, annotate it until you understand it, write a specification from your understanding, then use that specification to regenerate the code with AI and compare the result.

Q2.1 [10 pts] — Setup

Get the starter code: https://github.com/xiuyechen/1710-hw4-starter. Clone the repository. Run `tiny_rnn.py` and share a screenshot of the results in your VS Code terminal.

Q2.2 — Annotate & Specify

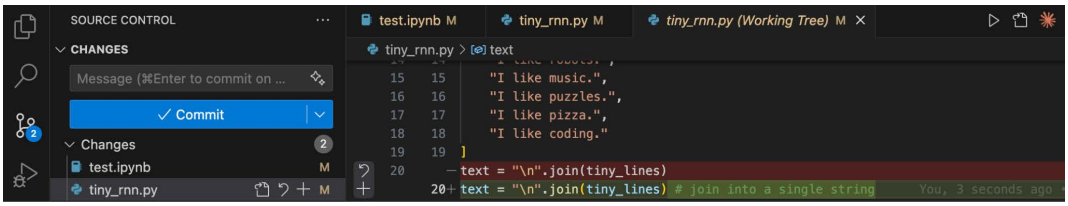
Try to understand the code by adding comments. Make a copy of `tiny_rnn.py` → `tiny_rnn_commented.py`. Stage and commit.

With AI help, go through the code piece by piece, taking notes in two ways:

Q2.2.1 [10 pts] — Inline comments

Type up notes as comments (using `#` for single-line comments) in *your own words* into `tiny_rnn_commented.py`. Work at a level of detail that feels challenging but not too challenging. Commit your changes.

Include a screenshot of the VS Code Source Control panel showing your commit.



Example: adding an inline comment and viewing the diff in Source Control.

Q2.2.2 [10 pts] — Written specification

Start a new text file `prompt.txt` in the same repository. Given your understanding (after studying with AI help):

- Type up the high-level description of what the code does.
- Type up the behavior of the demo.

Q2.3 [15 pts] — Regenerate & Compare

Use your `prompt.txt` as a detailed prompt to ask for a new AI-generated code implementation.

1. Test the new code. See if you can recreate the demo. Git commit.
2. Tweak the prompt to make it more similar to the original. Test again. Git commit again.
3. When you're done iterating, evaluate your result and discuss at least **1 notable difference** between the original and your version.

If you recreated it somewhat successfully, your prompt is now a good abstraction of the code—one that should also help your understanding of the original material. Commit often, and use meaningful, readable commit messages.

Q3. Tokenizer from Scratch

Compare three types of tokenizers: **character-level**, **word-level**, and **subword-level**. Use a single AI chat session for all of Q3. Create a folder `tokenizers` in your repository.

Use either your own example paragraph or the synthetic one below. Use the **same input text** for Q3.1–Q3.3.

Sample text for tokenization

```
In 2025, we teach "intro-to-AI" with hands-on labs-no hype. Students
ask: "Why tokens?"
Because models read pieces, not words. E.g., 'ChatGPT-5'  'Chat',
'GPT', '5' in all schemes.
We track loss/accuracy, compare char/word/BPE, and push to GitHub:
https://example.org/a/b?c=42.
Café prices rose 3.7%-blame supply-chain weirdness (and  demand).
```

Q3.1 [10 pts] — Character-level tokenizer

In `tiny_rnn.py`, there is a section that converts characters to numbers and back. Copy those lines here. Calculate and report the **length after tokenization**.

Q3.2 [15 pts] — Word-level tokenizer

Build a word-level tokenizer following this outline:

1. Tokenize by converting to lowercase, removing punctuation, splitting by whitespace.
2. Build a vocabulary.
3. Numericalize the tokens.
4. Decode back to tokens.
5. Join the decoded tokens.

- Write **pseudocode** for each step here in this document.
- Line by line, translate pseudocode to Python with AI help. Run and verify individual pieces, then put the pieces together **without AI help**. Commit at least 2 times.
- Calculate and report the **length after tokenization**.

Q3.3 [10 pts] — Subword tokens

Use OpenAI's tokenizer visualizer: https://platform.openai.com/tokenizer (GPT-4o / GPT-4o mini version).

- Calculate the **average characters per token**.
- Compare the length after tokenization across all three schemes. Describe the trade-offs.
- Research real-world tokenization and name a few reasons why it is challenging.