# University of Central Missouri

# Department of Computer Science & Cybersecurity

# CS5760 Natural Language Processing

# Fall 2025

# Homework 4.

# Student name:

## Submission Requirements:

- Once finished your assignment push your source code to your repo (GitHub) and explain the work through the ReadMe file properly. Make sure you add your student info in the ReadMe file.
- Submit your GitHub link on the Bright Space.
- Comment your code appropriately ***IMPORTANT.***
- Any submission after provided deadline is considered as a late submission.

**Part I. Short Answer**

**1) RNN Families & Use-Cases (Many-to-X)**
a) Map each task to the most suitable RNN I/O pattern and explain why (1–2 lines each):
• next-word prediction

• sentiment of a sentence

• NER

• machine translation
(Choose from: one-to-many, many-to-one, many-to-many aligned, many-to-many unaligned.)
b) In one sentence, explain how "unrolling" over time enables BPTT and weight sharing.
c) Give one advantage and one limitation of weight sharing across time in RNNs.

**2) Vanishing Gradients & Remedies**
a) Describe the vanishing gradient problem in RNNs and how it affects long-range dependencies.
b) List two *architectural* solutions and briefly how each helps gradient flow.
c) Give one *training* technique (not an architecture change) that can mitigate the issue and why.

**3) LSTM Gates & Cell State**
a) Explain the roles of the forget, input, and output gates. For each, name its activation and purpose.
b) Why is the LSTM cell state often described as providing a "linear path" for gradients?
c) In one or two sentences, contrast "what to remember" vs. "what to expose" in LSTMs.

**4) Self-Attention**

a) Define Query (Q), Key (K), and Value (V) in the context of self-attention.
b) Write the formula for dot-product attention.
c) Why do we divide by $\sqrt{d_k}$?

**5) Multi-Head Attention & Residual Connections**

a) Why do Transformers use multi-head attention instead of single-head attention?
b) What is the purpose of Add & Norm (Residual + LayerNorm)? Explain two benefits.
c) Describe one example of linguistic relation that different heads might capture (e.g., coreference, syntax).

**6) Encoder–Decoder with Masked Attention**

a) Why does the decoder use masked self-attention? What problem does it prevent?
b) What is the difference between encoder self-attention and encoder–decoder cross-attention?
c) During inference (no teacher forcing), how does the model generate tokens step by step?

## Part II: Programming

### Q1. Character-Level RNN Language Model ("hello" toy & beyond)

Goal: Train a tiny character-level RNN to predict the next character given previous characters.

Data (toy to start):

- Start with a small toy corpus you create (e.g., several "hello…", "help…", short words/sentences).
- Then expand to a short plain-text file of ~50–200 KB (any public-domain text of your choice).

Model:

- Embedding → RNN (Vanilla RNN or GRU or LSTM) → Linear → Softmax over characters.
- Hidden size 64–256; sequence length 50–100; batch size 64; train 5–20 epochs.

Train:

- Teacher forcing (use the true previous char as input during training).
- Cross-entropy loss; Adam optimizer.

Report:

1. Training/validation loss curves.
2. Sample 3 temperature-controlled generations (e.g., $\tau = 0.7, 1.0, 1.2$) for 200–400 chars each.
3. A 3–5 sentence reflection: what changes when you vary sequence length, hidden size, and temperature?
   (Connect to slides: embedding, sampling loop, teacher forcing, tradeoffs

### Q2. Mini Transformer Encoder for Sentences

Task: Build a mini Transformer Encoder (NOT full decoder) to process a batch of sentences.

Steps:

1. Use a small dataset (e.g., 10 short sentences of your choice).
2. Tokenize and embed the text.
3. Add sinusoidal positional encoding.
4. Implement:
   - Self-attention layer
   - Multi-head attention (2 or 4 heads)

- Feed-forward layer
- Add & Norm

5. Show:

    - Input tokens
    - Final contextual embeddings
    - Attention heatmap between words (visual or printed)

**Q3. Implement Scaled Dot-Product Attention**

Goal: Implement the attention function from your slides:

$$\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Requirements:

- Write a function in PyTorch or TensorFlow to compute attention.
- Test it using random Q, K, V inputs.
- Print:
    - Attention weight matrix
    - Output vectors
    - Softmax stability check (before and after scaling)