**University of Central Missouri**

**Department of Computer Science & Cybersecurity**

**CS5720 Neural Networks and Deep Learning**

**Summer 2025**

**Home Assignment 3. (Cover Ch 7, 9)**

**Student name: ALCHURI NIKHITHA**

**Submission Requirements:**

* Total Points: 100
* 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 and video on BrightSpace.
* Comment your code appropriately ***IMPORTANT.***
* Make a simple video about 2 to 3 minutes which includes demonstration of your home assignment and explanation of code snippets.
* Any submission after provided deadline is considered as a late submission.

**Q1: Implementing an RNN for Text Generation**

**Task:** Recurrent Neural Networks (RNNs) can generate sequences of text. You will train an **LSTM-based RNN** to predict the next character in a given text dataset.

1. Load a **text dataset** (e.g., "Shakespeare Sonnets", "The Little Prince").
2. Convert text into a **sequence of characters** (one-hot encoding or embeddings).
3. Define an **RNN model** using LSTM layers to predict the next character.
4. Train the model and generate new text by **sampling characters** one at a time.
5. Explain the role of **temperature scaling** in text generation and its effect on randomness.

ANS: **Temperature** controls randomness in predictions.

* + Low (e.g., 0.2): more deterministic and repetitive.
  + High (e.g., 1.2): more random and creative outputs.

***Hint:*** *Use tensorflow.keras.layers.LSTM() for sequence modeling.*

**Q2: NLP Preprocessing Pipeline**

Write a Python function that performs basic NLP preprocessing on a sentence. The function should do the following steps:

1. **Tokenize** the sentence into individual words.
2. **Remove common English stopwords** (e.g., "the", "in", "are").
3. **Apply stemming** to reduce each word to its root form.

**Use the sentence:**

**"NLP techniques are used in virtual assistants like Alexa and Siri."**

The function should print:

* A list of all tokens
* The list after stop words are removed
* The final list after stemming

**Expected Output:**

Your program should print three outputs in order:

1. **Original Tokens** – All words and punctuation split from the sentence
2. **Tokens Without Stopwords** – Only meaningful words remain
3. **Stemmed Words** – Each word is reduced to its base/root form

**Short Answer Questions:**

1. What is the difference between stemming and lemmatization? Provide examples with the word “running.”

ANS: **Stemming vs. Lemmatization**:

* **Stemming**: crude cutting, e.g., running → run.
* **Lemmatization**: uses dictionary, e.g., running → run, better → good.

1. Why might removing stop words be useful in some NLP tasks, and when might it actually be harmful?

ANS: **Stop Words**:

**Useful**: for classification where structure isn't needed.

**Harmful**: in summarization or translation, where context matters.

**Q3: Named Entity Recognition with SpaCy**

**Task:** Use the spaCy library to extract **named entities** from a sentence. For each entity, print:

* The **entity text** (e.g., "Barack Obama")
* The **entity label** (e.g., PERSON, DATE)
* The **start and end character positions** in the string

Use the input sentence:

**"Barack Obama served as the 44th President of the United States and won the Nobel Peace Prize in 2009."**

**Expected Output:**

Each line of the output should describe one entity detected

**Short Answer Questions:**

1. How does NER differ from POS tagging in NLP?

ANS: **NER vs POS**:

* NER labels real-world entities (e.g., PERSON).
* POS tags grammatical roles (noun, verb, etc.).

1. Describe two applications that use NER in the real world (e.g., financial news, search engines).

ANS: **Applications**:

**Finance**: Extracting company names, events from news.

**Search Engines**: Enhancing queries with entity context.

**Q4: Scaled Dot-Product Attention**

**Task:** Implement the **scaled dot-product attention** mechanism. Given matrices Q (Query), K (Key), and V (Value), your function should:

* Compute the dot product of Q and Kᵀ
* Scale the result by dividing it by √d (where d is the key dimension)
* Apply softmax to get attention weights
* Multiply the weights by V to get the output

**Use the following test inputs:**

***Q = np.array([[1, 0, 1, 0], [0, 1, 0, 1]])***

***K = np.array([[1, 0, 1, 0], [0, 1, 0, 1]])***

***V = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])***

**Expected Output Description:**

Your output should display:

1. The **attention weights matrix** (after softmax)
2. The **final output matrix**

**Short Answer Questions:**

1. Why do we divide the attention score by √d in the scaled dot-product attention formula?

ANS: we divide the attention score by √d in the scaled dot-product attention formula to prevent large dot products that lead to small gradients after SoftMax (stabilizes training).

1. How does self-attention help the model understand relationships between words in a sentence?

ANS: It captures dependencies between words, regardless of position (e.g., subject–verb agreement).

**Q5: Sentiment Analysis using HuggingFace Transformers**

**Task:** Use the HuggingFace transformers library to create a **sentiment classifier**. Your program should:

* Load a pre-trained sentiment analysis pipeline
* Analyze the following input sentence:

**"Despite the high price, the performance of the new MacBook is outstanding."**

* Print:
  + **Label** (e.g., POSITIVE, NEGATIVE)
  + **Confidence score** (e.g., 0.9985)

### **Expected Output**:

Your output should clearly display:

***Sentiment: [Label]***

***Confidence Score: [Decimal between 0 and 1]***

**Short Answer Questions:**

1. What is the main architectural difference between BERT and GPT? Which uses an encoder and which uses a decoder?

ANS: **BERT vs GPT**:

* **BERT**: encoder-only (good for classification).
* **GPT**: decoder-only (good for generation).

1. Explain why using pre-trained models (like BERT or GPT) is beneficial for NLP applications instead of training from scratch.

ANS:

* + Reduce compute cost.
  + Leverage vast training on large corpora.
  + Better generalization with minimal fine-tuning.