**University of Central Missouri**

**Department of Computer Science & Cybersecurity**

**CS5720 Neural network and Deep learning**

**Spring 2025**

**Home Assignment 4. (Cover Ch 9, 10)**

**Student name: Sushma Veluru**

**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 the BB.
* 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: 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. The difference between stemming and lemmatization:

Stemming crudely chops off word endings to get a root form (often not a real word), while lemmatization uses vocabulary and morphological analysis to return the base dictionary form (lemma).

Example with "running":

Stemming: "run" (might also return "runn" with some stemmers)

Lemmatization: "run" (proper dictionary form)

1. Why might removing stop words be useful in some NLP tasks, and when might it actually be harmful?
2. Ans. Stop word removal considerations:
   * Useful when:
     + We want to focus on content words (e.g., for topic modeling, search engines)
     + Reducing dimensionality of text data
     + Stop words carry little semantic meaning in the specific task
   * Harmful when:
     + The task involves understanding sentence structure (e.g., machine translation)
     + Working with phrases or idioms where stop words are important (e.g., "to be or not to be")
     + In sentiment analysis where negation words (like "not") are technically stop words
     + When context depends on function words (e.g., "The cake" vs "A cake")

**Q2: 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. Difference between NER and POS tagging:

* + NER (Named Entity Recognition) identifies and classifies named entities (people, organizations, locations, dates, etc.) in text.
  + POS (Part-of-Speech) tagging identifies the grammatical category of each word (noun, verb, adjective, etc.).
  + Example: In "Apple released a new product":
    - NER would tag "Apple" as an ORGANIZATION
    - POS tagging would tag "Apple" as a PROPN (proper noun) and "released" as a VERB.

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

Ans. Two real-world applications of NER:

* + **News Aggregation**: Automatically extracting people, organizations, and locations from news articles to categorize stories or build knowledge graphs.
  + **Customer Support Automation**: Identifying product names, problem types, and dates in customer queries to route tickets to the appropriate department or suggest solutions.
  + Other examples include:
    - Resume parsing to extract skills, education, and experience
    - Medical records processing to identify drugs, diseases, and treatments
    - Financial document analysis to extract company names and monetary figures

**Q3: 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.

* Why do we divide the attention score by √d in scaled dot-product attention?The scaling factor √d (where d is the key dimension) prevents the dot product from becoming too large when the dimensionality is high.
* Without scaling, large dot products lead to vanishing gradients in softmax (extremely sharp distributions where one value dominates).
* Scaling ensures stable gradients and better learning in deep networks.

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

Ans.

* How does self-attention help the model understand relationships between words in a sentence?Self-attention computes weighted relationships between all words in a sequence.
* Each word’s representation is updated based on its relevance (attention weights) to other words.
* For example:

1. In "The cat sat on the mat", the word "sat" would attend strongly to "cat" (subject) and "mat" (location).
2. This allows the model to capture long-range dependencies (e.g., subject-verb agreement) better than RNNs/CNNs.
3. Unlike traditional methods (e.g., RNNs), self-attention directly models interactions between all words, making it more parallelizable and efficient.

**Q4: 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 (Bidirectional Encoder Representations from Transformers) uses only the encoder part of the Transformer architecture.
* It reads text bidirectionally (left and right context simultaneously).
* Optimized for understanding tasks like classification, QA, NER.
* GPT (Generative Pre-trained Transformer) uses only the decoder part.
* It reads text unidirectionally (left-to-right).
* Optimized for generative tasks like text completion, summarization, and generation.

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

Ans.

* Saves time and resources: Training from scratch requires vast datasets, compute power, and time.
* High accuracy: Pre-trained models are trained on massive corpora and capture general language understanding.
* Transfer learning: You can fine-tune them on your specific task with minimal labeled data.
* Reduced complexity: Easy to plug-and-play with tools like HuggingFace for tasks like sentiment analysis, translation, and more.