National University of Computer and Emerging Sciences

School of Computing | Fall 2024 | Islamabad Campus

AI4001/CS4063
NLP DS(A,B)
AI(A)

Serial No:

Sessional I
Total Time: 1 Hour
Total Marks: 50

Signature of Invigilator

Tuesday, September 24, 2024
Course Instructor
Mirza Omer Beg

Student Name | Roll No | Section | Signature

DO NOT OPEN THE QUESTION BOOK OR START UNTIL INSTRUCTED.

Instructions:

1. Verify at the start of the exam that you have a total of three (3) questions printed on four (4) pages including this title page.
2. Attempt all questions on the question-book and in the given order.
3. This exam is **open book, open notes.** Mobiles, Internet and note-sharing is not allowed. Please see that the area in your threshold is free of any material classified as *useful in the paper*, i.e. mobile/internet or else there may be a charge of cheating.
4. Read the questions carefully for clarity of context and understanding of meaning and make assumptions wherever required, for neither the invigilator will address your queries, nor the teacher/examiner will come to the examination hall for any assistance.
5. Fit in all your answers in the provided space. You may use extra space on the last page if required. If you do so, clearly mark question/part number on that page to avoid confusion.
6. Use only your own stationery and calculator. If you do not have your own calculator, do manual calculations.
7. Use only permanent ink-pens. Only the questions attempted with permanent ink-pens will be considered. Any part of paper done in lead pencil cannot be claimed for checking/rechecking.

	Q1	Q2	Q3	Total
Marks Obtained	18	8	14	40
Total Marks	20	15	15	50

Q1. Byte Pair Encoding (BPE) (20 Marks) [3+5+5+7]

The pseudocode below (**Algorithm 1**) implements the Byte Pair Encoding (BPE) algorithm. which is widely used in tokenization processes for large language models (LLMs). BPE iteratively merges the most frequent pairs of symbols to build a vocabulary.

Algorithm 1 BPE TOKENIZATION

```
1: procedure BPE(C, V)
2:     merges ← {}
3:     while True do
4:         pair ← GETMOSTFREQUENTPAIR(C)
5:         newToken ← MERGEPAIR(C, pair)
6:         V ← UPDATEVOCABULARY(pair)
7:         merges ← ADD2MERGES(newToken, pair)
8:     return V, merges                    ▷ THE BPE VOCABULARY
```

(a) In practice, a maximum number of merge operations are performed in BPE. Modify the above pseudocode to include a stopping condition based on a maximum number of merges, N_{max}.

(b) Some logical mistakes may have been intentionally added to the above pseudocode. Circle any error(s) and state their correction. If there are no errors, circle None.

(c) **Algorithm 2** below shows the encoding process for BPE. The vocabulary and merge operations are stored in $V, merges$ and used to tokenize new text T.

Algorithm 2 ENCODE WITH BPE

```
1: procedure ENCODE(V, merges, T)
2:     tokens ← EXTRACTSYMBOLSFROMTEXT(T)
3:     while ∃ PAIR ∈ tokens THAT IS IN V, merges do
4:         cipher ← MERGEPAIR(tokens, pair)
5:     return cipher                       ▷ THE BPE TOKENIZED TEXT
```

Explain why this iterative process might slow down with large vocabularies and suggest an optimization.

(d) Write the *decode*() procedure for BPE.

```
def decode(cipher, V)
    text = b"".join(V[c] for c in cipher)
    text = text.decode('UTF-8', error = 'replace')
    return text
```

Q2. EDA on Urdu Audio-Text Pairs (15 Marks) [5+10]

The dataset consists of Urdu audio files paired with their respective text transcripts. Your goal is to perform EDA to understand the characteristics of this dataset.

(a) The following Python code attempts to compute the distribution of audio file durations and word counts in transcripts:

```
import pandas as pd
import librosa
import matplotlib.pyplot as plt

# Load dataset
df = pd.read_csv("urdu_audio_transcripts.csv")

# Calculate audio duration
df['duration'] = df['audio_file'].apply(lambda x: librosa.get_duration(filename=x))

# Calculate word count in transcripts
df['word_count'] = df['transcript'].apply(lambda x: len(x))

# Plotting the distribution of audio durations
plt.hist(df['duration'], bins=20)
plt.title('Audio Duration Distribution')
plt.show()

# Plotting the word count distribution
plt.hist(df['word_count'], bins=20)
plt.title('Transcript Word Count Distribution')
plt.show()
```

The above code contains several errors. Identify and correct them. Explain your corrections and how they affect the output. No error in the above code

(b) Suggest a method to identify and visualize cases where the audio duration and the length of the corresponding transcript (measured by word count) are mismatched. Write a code snippet for this task.

```
for i in range(len(word_count)):
    if(df[duration][i] == df['word_count'][i]):
        continue
    else:
        x = df['duration'][i]
        y = df['word_count'][i]
plt.hist(x, y, bins = 20)
plt.title('Mismatched length')
```

National University of Computer and Emerging Sciences

School of Computing Islamabad Campus

Q3. Parts-of-Speech (15 Marks) [3+12]

Consider the following POS annotated story.

```
Little/JJ Red/NNP Riding/NNP Hood/NNP lived/VBD in/IN the/DT woods/NNS with/IN her/PRP mother/NN .
One/CD day/NN Little/NNP Red/NNP Riding/NNP Hood/NNP went/VBD to/TO visit/VB her/PRP granny/NN .
She/PRP had/VBD a/DT nice/JJ cake/NN in/IN her/PRP basket/NN .
On/IN her/PRP way/NN Little/JJ Red/NNP Riding/NNP Hood/NNP met/VBD a/DT wolf/NN .
Hello/UH ! the/DT wolf/NN said/VBD . Where/WRB are/VBP you/PRP going/VBG ?/.
I/PRP m/RB going/VBG to/TO see/VB my/PRP grandmother/NN . She/PRP lives/VBZ in/IN a/DT house/NN
behind/IN those/DT trees./NNS
The/DT wolf/NN ran/VBD to/TO Granny/NNP '/NNP s/NN house/NN and/CC ate/NN Granny/NNP up/RB . He/PRP
got/VBD into/IN Granny/NNP '/NNP s/NN bed/NN . A/DT little/JJ later/RB , Little/NNP Red/NNP Riding/NNP
Hood/NNP reached/VBD the/DT house/NN . She/PRP looked/VBD at/IN the/DT wolf/NN .
Granny/NNP , what/WP big/JJ eyes/NNS you/PRP have/VBP !
All/PDT the/DT better/JJR to/TO see/VB you/PRP with/IN ! said/VBD the/DT wolf/NN .
Granny/NNP , what/WP big/JJ ears/NNS you/PRP have/VBP !
All/PDT the/DT better/JJR to/TO hear/VB you/PRP with/IN ! said/VBD the/DT wolf/NN .
Granny/NNP , what/WP a/DT big/JJ nose/NN you/PRP have/VBP !
All/PDT the/DT better/JJR to/TO smell/VB you/PRP with/IN ! said/VBD the/DT wolf/NN .
Granny/NNP , what/WP big/JJ teeth/NNS you/PRP have/VBP !
All/PDT the/DT better/JJR to/TO eat/VB you/PRP with/IN ! shouted/VBD the/DT wolf/NN .
A/DT woodcutter/NN was/VBD in/IN the/DT wood/NN . He/PRP heard/VBD a/DT loud/JJ scream/NN and/CC
ran/NN to/TO the/DT house/NN . The/DT woodcutter/NN hit/VBD the/DT wolf/NN over/IN the/DT head/NN
. The/DT wolf/NN opened/VBD his/PRP mouth/NN wide/JJ , gave/VBD a/DT cry/NN and/CC Granny/NNP
jumped/VBD out/RP . The/DT wolf/NN ran/VBD away/RB and/CC Little/JJ Red/NNP Riding/NNP Hood/NNP
never/RB saw/VBD the/DT wolf/NN again/RB .
```

(a) Assume that the `similarity()` method can be used for unsupervised training. It uses the context of surrounding word annotations (one word before and one word after) to find similar words i.e. words used in the same context. Highlight or underline the words in the above text that are used in a similar context as the word 'eyes'.

eyes, ears, nose, teeth, are the words in the above text that are used in a similar context as 'eyes' 3

(b) Write an algorithm that, given a word w, can extract similar words from an annotated corpus T, such as the above.

```
Similarity(w, T):

1-  for i in range(len(T)):
2-      if (T[i] == w):          # or not(
3-          sim_word1 = T[i-1]
4-          sim_word2 = T[i+1]
        return sim_word1, sim_word2

    for i in range(len(T))
        if (T[i] == sim_word1 && T(i+2) == sim_word2)
        { return T[i+1] # Similar word as w
        }
```