COMPUTING FOR MEDICINE

MID-SEMESTER EXAMINATION

Total Marks: 50
Time: 1 Hour

Q1- Fill in the Blank:
In the attention mechanism, __________ is the right set of matrix multiplications for Query(Q), Key(K), Value(V) to compute attention scores (ignore scaling factors). (2 Marks)
a) $(QK^T)V$
b) $(VK^T)Q$
c) $K (VQ^T)$
d) $(KQ^T)V$

Q2- Which of the following is NOT true about the Transformer architecture? (2 Marks)
a) It relies on positional encoding to preserve word order.
b) It uses a sequence of recurrent layers.
c) It includes feed-forward layers after each attention mechanism.
d) It uses multi-head attention for better learning of relationships between words.

Q3- What in Transformers directly enables learning of weights? (2 Marks)
a) Skip connections
b) Feed-forward layer
c) Multi-head Attention
d) Positional Encoding

Q4- Which of the following IS a problem with the Bag of Words representation? (2 Marks)
a) Enables character level representation
b) Produces non-sparse representations
c) Loss of word order
d) Effectively manages out of vocabulary (OOV) words

Q5- In the context of interoperability which of the following is responsible for preserving meaning of the information transfered? (2 Marks)
a) Human interoperability
b) Syntactic interoperability
c) Process interoperability
d) Semantic interoperability

Q6- Define pre-coordination and post-coordination in the context of SNOMED CT. Provide an example of each and explain how post-coordination enhances the flexibility of SNOMED CT.

Pre-coordination and Post-coordination in SNOMED CT (3marks)

Pre-coordination refers to the use of a **single SNOMED CT concept** that represents a complete clinical meaning in a single, predefined term. It simplifies clinical documentation by providing one concise identifier for a particular clinical situation. In pre-coordination, the concept is fully expressed without needing to combine other concepts.

- **Example of Pre-coordination:**
  The SNOMED CT concept "Acute Myocardial Infarction" (Concept ID: 22298006) is a pre-coordinated term. It provides a complete clinical diagnosis in a single term, which can be directly used for coding a patient's condition.

Post-coordination, on the other hand, allows users to **combine multiple SNOMED CT concepts** to create a more detailed or specific clinical description. This approach enhances the flexibility of SNOMED CT, enabling clinicians to document complex scenarios that might not be covered by pre-coordinated terms alone. It involves the use of multiple SNOMED CT concepts and relationships (e.g., refinements, attributes) to build a composite clinical statement.

- **Example of Post-coordination:**
  If "Pneumonia" (Concept ID: 233604007) is combined with "Caused by Streptococcus pneumoniae" (Concept ID: 233580004), the resulting description is more specific: "Pneumonia caused by Streptococcus pneumoniae." This combination is not pre-coordinated in SNOMED CT but can be expressed using post-coordination.

How Post-coordination Enhances Flexibility (2 marks)

Post-coordination enhances the flexibility of SNOMED CT by allowing healthcare providers to document clinical information with **greater specificity and precision**. It enables the creation of unique and tailored clinical expressions for situations that may not have pre-coordinated terms available. This is especially useful in dynamic clinical environments where new diseases, symptoms, or combinations of conditions arise that need detailed representation.

Post-coordination also supports the **scalability** of SNOMED CT, as it avoids the need for the terminology to predefine every possible clinical concept. Instead, users can dynamically combine existing concepts to represent new or complex medical scenarios. For example, a clinician could document "Fracture of the femur with complications" by combining multiple concepts to detail the specific type of fracture and associated complications. This flexibility ensures that SNOMED CT can adapt to the evolving needs of healthcare data representation.

Q7- Explain the concept of look-ahead masking and positional encoding in a Transformer model. (10 Marks)

Look-ahead Masking and Positional Encoding in a Transformer Model

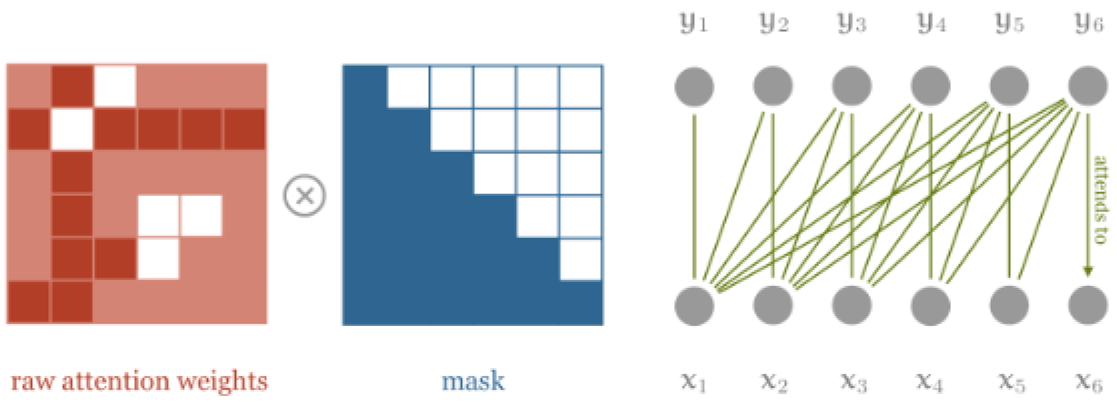
Look-ahead Masking (5 marks)

Look-ahead masking is a technique used in the **decoder** of a Transformer model during tasks like text generation (e.g., machine translation). Its primary purpose is to ensure that, at each step of prediction, the model can only consider previous tokens and not future ones. This is essential in **auto-regressive** models where the output at each time step depends on previous outputs but should not depend on the future ones.

In simpler terms, look-ahead masking prevents the model from "cheating" by seeing future tokens during the generation process. It forces the decoder to focus only on the tokens that have been generated so far, ensuring that predictions are made one token at a time.

- **Mechanics of Look-ahead Masking:**
  A **mask** is applied to the attention scores so that future positions (i.e., tokens that have not been predicted yet) are masked out, and their attention weights are set to a very low value (often negative infinity), making them effectively invisible to the model. This means that for a given position `i`, the model only attends to positions `1` to `i`, and not any beyond `i`.

Look ahead mask


raw attention weights
mask
y_1 y_2 y_3 y_4 y_5 y_6
attends to
x_1 x_2 x_3 x_4 x_5 x_6

- **Example in Action:**
  If the input sequence is ["The", "dog", "barked"], and the model is generating this sequence step by step, at step 1 ("The"), the model should not know that "dog" or "barked" will follow. The look-ahead mask ensures this by blocking access to those tokens during that time step.

Look-ahead masking is critical in tasks where we need to generate sequences, such as text generation or translation, ensuring the autoregressive property of the model is maintained.

Positional Encoding (5 marks)

Transformers, unlike recurrent or convolutional models, do not have any inherent way of capturing the **sequential order** of tokens, as the model processes all tokens in parallel. To address this, **positional encoding** is introduced to provide information about the position of each token in the sequence.

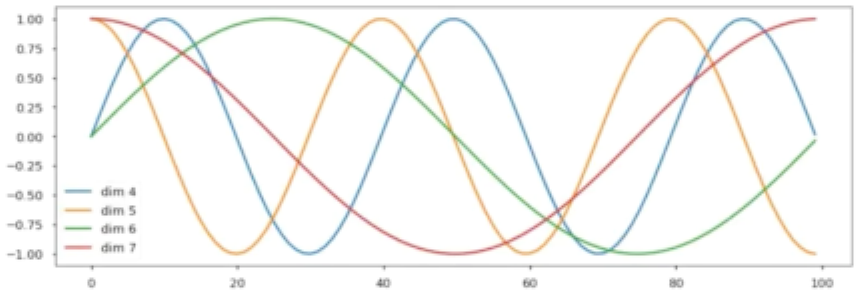
- **Concept:**
  In Transformers, input tokens are transformed into embeddings (continuous vector representations). However, these embeddings alone do not contain any information about the position of the tokens in the sequence. Positional encoding adds this missing information by incorporating unique positional values into each token embedding, allowing the model to differentiate between tokens based on their position in the sequence.
  Positional encodings are added to the input embeddings before they are passed into the self-attention layers. The idea is that these encodings represent positions using **sinusoidal functions** of varying frequencies, which ensures that tokens in the sequence

are ordered and that the model can infer positional relationships between words. The formula for positional encoding is:

Positional Encodings

No convolution, no recurrence: Let's add a positional encoding!

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/dmodel})$$
$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/dmodel})$$



where `pos` is the position and `i` is the dimension. This creates a unique encoding for each position, ensuring that the model understands relative positions in the sequence.

- **Why Positional Encoding is Necessary:**
  Unlike models like Recurrent Neural Networks (RNNs) or Long Short-Term Memory networks (LSTMs), Transformers lack a built-in mechanism to process sequences in order. Without positional encoding, the Transformer would treat each token independently, making it impossible for the model to understand the context based on the token's position in the sentence. For example, without knowing the order, the model would have no way to differentiate between "The dog chased the cat" and "The cat chased the dog."
- **Example in Action:**
  In the sentence "I went to the store yesterday," the positional encoding ensures that "yesterday" is understood as happening after "went," even though all tokens are processed in parallel. The positional encodings give the Transformer the ability to process the input sequence in a meaningful order.

Q8- Describe how the Skip-Gram and CBoW variants of Word2Vec work. (10 Marks)

Skip-Gram and Continuous Bag of Words (CBOW) Variants of Word2Vec

Word2Vec is a popular technique used to generate word embeddings, representing words as dense vectors in a continuous vector space. It has two main training architectures: **Skip-Gram** and **Continuous Bag of Words (CBOW)**. Both models are based on the distributional hypothesis, which states that words appearing in similar contexts tend to have similar meanings.

1. Skip-Gram Model

The **Skip-Gram** model's goal is to predict the **context words** (surrounding words) given a **target word**. In other words, given a word in the middle of a sentence, the model tries to predict which words are likely to appear in the surrounding window of that word.

Mechanics of Skip-Gram:(3 marks)

- **Input:** The target word (central word in a window).
- **Output:** The model predicts the context words (words surrounding the target word within a certain window size).
- **Objective:** Maximize the likelihood of predicting the correct context words for a given target word.

The Skip-Gram model generates a probability distribution over the vocabulary for each surrounding word. For example, in the sentence:
"The dog chased the cat",
if "dog" is the target word, the model will try to predict "The" and "chased" as context words based on the window size.

Example and diagram:(2 marks)

Consider the sentence:
"I love playing football".

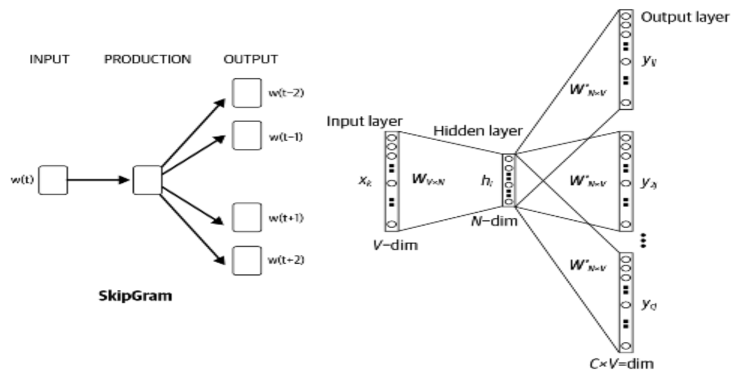
If the window size is 2 and the target word is "love," the context words are "I" and "playing." The Skip-Gram model will try to predict "I" and "playing" given the input word "love."

- Target: "love"
- Context words: "I," "playing"

Advantages of Skip-Gram:

- Effective for smaller datasets.
- Performs well with infrequent words since it focuses on predicting all surrounding words for each target word.

Continuous SkipGram Model



2. Continuous Bag of Words (CBOW) Model

The **CBOW** model works in the reverse direction compared to Skip-Gram. Instead of predicting context words, the CBOW model aims to predict the **target word** given its **context words** (the surrounding words). Essentially, CBOW aggregates information from the context and uses it to infer the most likely target word.

Mechanics of CBOW:(3 marks)

- **Input:** The context words (surrounding words within a window).
- **Output:** The model predicts the target word (the central word in the window).
- **Objective:** Maximize the likelihood of predicting the correct target word based on its context.

In CBOW, the context words are averaged or summed together to generate a single vector representation, which is then used to predict the target word.

Example and diagram:(2 marks)

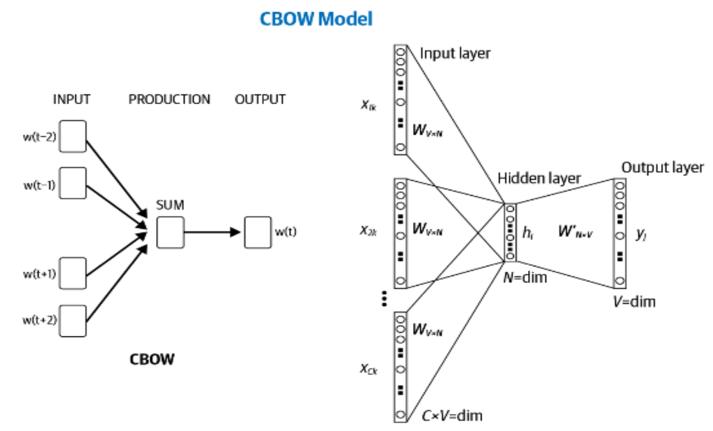
Consider the sentence:
"I love playing football".

If the window size is 2 and the target word is "love," the input context words would be "I" and "playing." The CBOW model tries to predict the word "love" based on these context words.

- Context words: "I," "playing"
- Target: "love"

Advantages of CBOW:

- Faster to train compared to Skip-Gram, especially on large datasets.
- Smooths over the prediction of rare words by aggregating context information.



Q9- Describe the purpose of the multi-head attention mechanism in Transformers. How does it differ from single-head attention, and what are the benefits? Illustrate with an example. (15 Marks)

Criteria	Marks	Description
Understanding of Multi-Head Attention	5	- Clearly explains the purpose of multi-head attention. - Demonstrates knowledge of how attention works in Transformers (query, key, value). - Correctly describes the role of multi-head attention in capturing diverse relationships between words
Comparison with Single-Head Attention	4	- Explains how multi-head attention differs from single-head attention.
Benefits of Multi-Head Attention	3	- Clearly outlines the advantages of multi-head attention, including better representation of different aspects of word relationships, improved model performance, and richer embeddings. - Emphasizes specific benefits such as enhanced parallelization, capturing long-range dependencies, and better expressiveness of learned embeddings.
Example and Illustration	3	- Provides a clear, concise example demonstrating multi-head attention in action (e.g., an NLP task like machine translation or text summarization). - Correctly illustrates how multiple heads focus on different relationships (e.g., syntactic vs. semantic)

		in the input.

Multi-Head Attention Mechanism in Transformers

The **multi-head attention** mechanism is a critical component of the Transformer architecture. It enables the model to focus on different parts of the input sequence simultaneously, allowing it to capture various types of relationships between words or tokens. Multi-head attention enhances the model's ability to process contextual information efficiently and is one of the primary reasons for the Transformer's superior performance in natural language processing tasks like translation, summarization, and text generation.

Purpose of Multi-Head Attention

The purpose of **multi-head attention** is to allow the Transformer model to focus on different aspects of the input sequence from multiple perspectives. This is achieved by applying multiple attention "heads," each of which computes attention using a different learned representation. The outputs from each attention head are then concatenated and transformed through a linear layer.

In simpler terms, instead of focusing on just one type of relationship between words (like a single-head attention mechanism would), multi-head attention allows the model to capture multiple, diverse relationships. For example, one attention head might focus on the direct relationship between a subject and a verb, while another head might focus on the relationship between an adjective and a noun, all within the same sentence.

How Multi-Head Attention Works

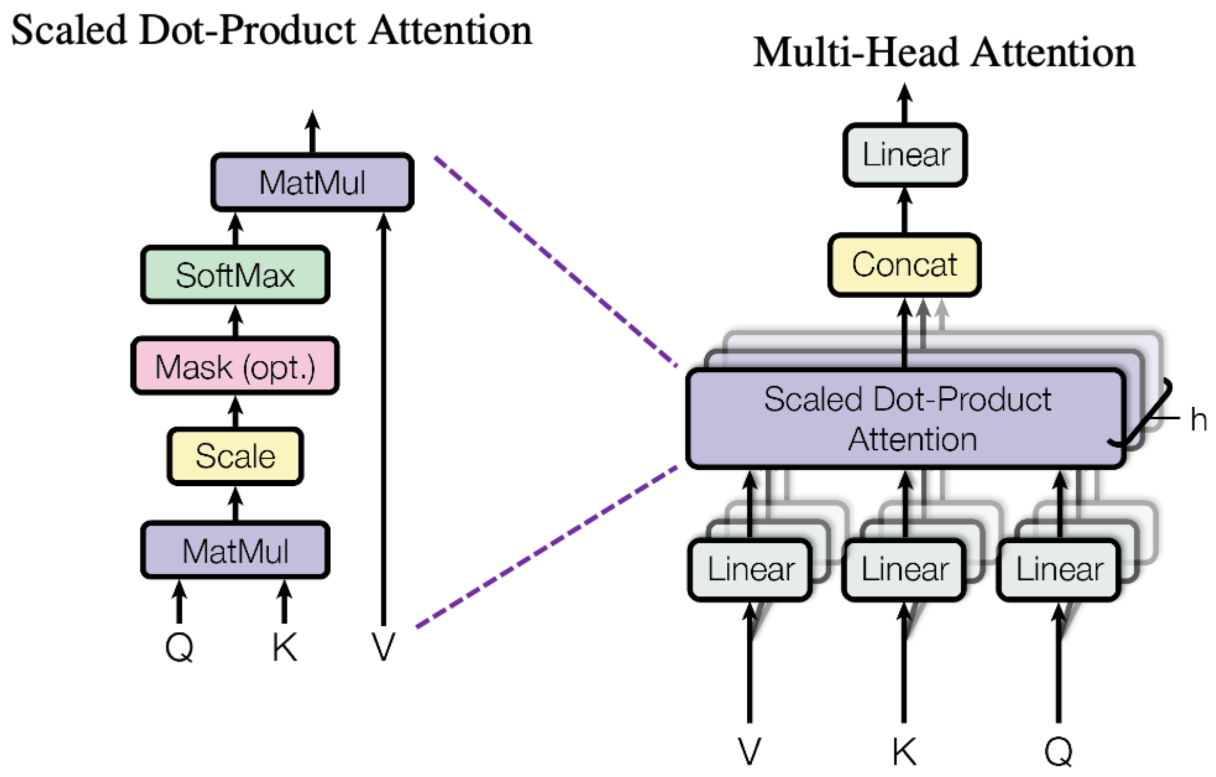
In a multi-head attention mechanism, the input sequence is first projected into multiple sets of **Query (Q), Key (K), and Value (V)** vectors (one set for each attention head). Each attention head performs scaled dot-product attention independently, as follows:

- The **query** is multiplied with the **key** to compute attention scores.
- The attention scores are normalized and used to weight the **value** vectors.
- This process is repeated for each attention head, where each head learns different projections for Q, K, and V.

The results from each head are then concatenated and passed through a final linear layer to produce the output. Mathematically, for each attention head `i`, the output is computed as:

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

$$\begin{aligned}\text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O \\ \text{where head}_\text{i} &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V\end{aligned}$$



Difference Between Single-Head and Multi-Head Attention

In **single-head attention**, the model computes attention once, using a single set of Q, K, and V vectors for the entire sequence. This means that the model can only attend to one aspect or relationship within the sequence at a time. While single-head attention can still capture meaningful relationships, it is limited to a single perspective on the data.

In contrast, **multi-head attention** uses multiple heads, each with its own set of learned Q, K, and V vectors. This allows the model to attend to multiple aspects of the sequence simultaneously, giving it a richer understanding of the relationships within the data.

Key Differences:

- **Single-head attention**: Only one attention head focuses on the entire input sequence, capturing one type of relationship.
- **Multi-head attention**: Multiple attention heads focus on different parts of the input, capturing multiple relationships simultaneously. Each head has its own set of learned weights and operates independently, giving a broader and more nuanced understanding of the input.

Benefits of Multi-Head Attention

1. Capture Diverse Information:
With multiple attention heads, the Transformer can capture different types of relationships between tokens. For example, one head might focus on long-range dependencies (like the relationship between the subject and verb in a long sentence), while another might focus on short-range dependencies (such as between an adjective and noun).

2. Better Representation Learning:
Each attention head learns to focus on different aspects of the sequence, producing richer and more comprehensive word representations. This allows the model to understand the same word in different contexts (polysemy) more effectively.

3. Improved Parallelization:
Since multi-head attention computes attention in parallel across different heads, it is computationally efficient. This is in contrast to recurrent models (like RNNs or LSTMs), which have inherent sequential dependencies that limit parallelization.

4. Robust to Noisy Data:
With multiple attention heads, the model is less prone to overfitting to specific noisy patterns in the data. Each head learns a different projection, making the model more robust and generalizable.

Example of Multi-Head Attention

Consider the sentence:
"The cat chased the dog in the garden."

- **Head 1**: Might focus on the relationship between the subject ("The cat") and the verb ("chased"). It helps the model understand that "The cat" is the one performing the action of chasing.
- **Head 2**: Could focus on the prepositional phrase ("in the garden") and its relationship with the verb. This helps the model understand where the action is taking place.
- **Head 3**: Might focus on the relationship between the direct object ("the dog") and the verb ("chased"), helping the model understand that "the dog" is the entity being chased.

Each attention head captures a different aspect of the sentence, enabling the model to better understand the overall meaning.
