Introduction to Large Language Models
Assignment- 7

Number of questions: 7 **Total mark: 4 X 1 + 3 X 2 = 10**

QUESTION 1: [1 mark]

Why can a pre-trained BART model be fine-tuned directly for abstractive summarization?

a) Its encoder alone is sufficient.
b) It shares vocabulary with summarization datasets.
c) It uses a larger context window than BERT.
d) It already contains a generative decoder trained jointly during pre-training.

Correct Answer: d

Explanation:

- **Abstractive summarization** is a sequence-to-sequence (seq2seq) task that requires the model to *generate* new text (a summary) based on an input text (an article).
- **BART (Bidirectional and Auto-Regressive Transformer)** is explicitly designed as an **encoder-decoder** model. Its pre-training task involves corrupting an input (e.g., masking or deleting text) and training the model to reconstruct the original, clean text.
- This pre-training process jointly trains a **bidirectional encoder** (to understand the corrupted input) and an **autoregressive decoder** (to generate the clean output).
- Because BART already possesses this powerful, pre-trained generative decoder, it is perfectly suited for fine-tuning on other generative tasks like summarization, where it learns to map an article (via the encoder) to a summary (via the decoder).

Why not the others?

- (a) An encoder *alone* (like BERT) is not designed for text generation. It produces representations, which are typically used for classification or span-prediction tasks.
- (b) Vocabulary sharing is incidental and not the primary *architectural* reason BART is suitable for this task.
- (c) The context window size is a model hyperparameter and does not determine its ability to perform generative tasks.

QUESTION 2: [2 marks]

For pre-training of encoder-decoder models, which statement(s) is/are true?

a) The encoder attends bidirectionally to its whole input.
b) The decoder conditions on earlier decoder tokens and encoder outputs.
c) Unlabelled text is turned into a supervised task via a noising scheme.
d) Training relies on a next-sentence-prediction loss.

Correct Answer: a, b, c

Explanation:

- **(a) True:** The encoder part of an encoder-decoder model (like in BART or T5) is designed to be bidirectional. This allows it to build a rich representation of the input sequence by considering both left and right context for every token, which is crucial for understanding the "corrupted" input.
- **(b) True:** This describes the standard Transformer decoder mechanism. The decoder is **autoregressive**, so it uses a causal (masked) self-attention to look at previously generated tokens. It also uses **cross-attention** to look at the complete output from the encoder, allowing it to condition the generated sequence on the input sequence.
- **(c) True:** This is the core idea of pre-training models like BART and T5. We only have unlabelled text, so we create a "supervised" task. We apply a **noising function** (like masking, deleting, or permuting spans of text) to the input and train the model to "denoise" it, i.e., predict the original, clean text.
- **(d) False:** The Next Sentence Prediction (NSP) loss was a pre-training objective used for the original **BERT** (an encoder-only model) to help it understand sentence relationships. Encoder-decoder models like BART and T5 use denoising/span-corruption objectives, not NSP.

QUESTION 3: [2 marks]

Which attention mask(s) prevent(s) a token from looking at future positions?

a) Causal mask
b) Fully-visible mask
c) Prefix-LM mask
d) All of the above
e) None of the above

Correct Answer: a, c

Explanation:

- **(a) Causal mask:** This is the standard mask used in decoder-only models (like GPT). It's a triangular mask that ensures a token at position i can only attend to tokens at positions 1...i and *not* to any "future" tokens at positions i+1 ... N. This is essential for autoregressive generation.
- **(c) Prefix-LM mask:** This mask is a hybrid. It divides the sequence into a "prefix" (input) and a "suffix" (output). It allows **fully-visible** (bidirectional) attention over the prefix, but applies a **causal mask** to the suffix. Therefore, for any token in the suffix (the part being generated), it *is* prevented from looking at future positions within that suffix, just like a standard causal mask.
- **(b) Fully-visible mask:** This mask, used in encoders (like BERT), allows every token to attend to every other token in the sequence, including future ones. It does not prevent looking ahead.

QUESTION 4: [1 mark]

T5 experiments showed that clean and compact pre-training data can outperform a larger but noisier corpus primarily because:

A. Larger corpora overfit.
B**.** Noise forces the model to waste capacity on modelling irrelevant patterns.
C. Clean data has longer documents.
D. Compact data allows bigger batches.

Correct Answer: b

Explanation:

- The T5 paper introduced the "Colossal Clean Crawled Corpus" (C4), which was meticulously filtered to remove non-natural-language content like code, menus, "Lorem ipsum" placeholder text, and other web-page "noise."
- In experiments, the clean **C4** dataset produced better results on downstream tasks than the much larger **unfiltered** C4 dataset.
- **(B)** This is the correct reason. A model trained on noisy data must use a portion of its finite capacity (parameters) to learn patterns in the noise (e.g., how to predict JavaScript code or HTML tags). This "wasted capacity" is then unavailable for learning the useful patterns of natural language, leading to worse performance on downstream NLP tasks.
- **(A)** Overfitting is typically a concern for *small* datasets, not massive corpora.
- **(C) & (D)** The cleaning process and dataset size do not guarantee longer documents or bigger batch sizes; these are not the primary reasons for the performance difference.

QUESTION 5: [1 mark]

What makes sampling from an auto-regressive language model straightforward?

A. The model is deterministic.
B. The vocabulary is small.
C. Each conditional distribution over the vocabulary is readily normalised and can be sampled token-by-token.
D. Beam search guarantees optimality.

Correct Answer: c

Explanation:

- An auto-regressive language model, the probability of a text sequence X is a product of conditional probabilities: $P(X) = \prod P(x_i | x_1, \ldots, x_{i-1})$.
- **(C)** This property is what makes sampling possible. At each step i, the model takes the context (all previous tokens x1...xi−1) and produces a vector of logits (raw scores) for every token in the vocabulary. A **softmax** function is applied to these logits to create a complete, normalized probability distribution. Sampling is then the

straightforward process of picking one token from this distribution. This new token is appended to the context, and the process repeats.

- **(A)** The neural network's forward pass is deterministic, but the *sampling* process itself is stochastic (probabilistic) by definition.
- **(B)** The vocabulary is typically very large, not small.
- **(D)** Beam search is a decoding algorithm (a heuristic search to find a high-probability sequence), not a sampling method. It also does not guarantee finding the optimal (most probable) sequence.

QUESTION 6: [1 mark]

Why does **ELMo** build its input token representations from a **character-level CNN** instead of fixed word embeddings?

A. To reduce training time by sharing parameters
B. To avoid **UNK** tokens and generate representations for any string
C. To compress embeddings to 128 dimensions
D. To ensure the same vector for a word in every context

Correct Answer: b

Explanation:

- **(B)** The primary advantage of using a character-level CNN is to handle out-of-vocabulary (OOV) words. A model with fixed word embeddings (like word2vec) has a finite vocabulary; any word not in this vocabulary is mapped to a single "UNK" (unknown) token, losing all its meaning. By building representations from characters, ELMo can compose a unique vector for *any* word, including rare words, misspelled words, or new words, as long as it's made of known characters.
- **(A)** While CNNs do share parameters, this is a general property, not the specific reason for choosing them *over* fixed word embeddings in this context.
- **(C)** The lecture slides state the projection is to 512 dimensions, not 128.
- **(D)** This is the exact *problem* ELMo was designed to *solve*. ELMo's goal is to create *context-dependent* representations, whereas fixed embeddings (the alternative) *do* have the same vector for a word in every context.

QUESTION 7: (Numerical Question) [2 marks]

The **einsum** function in numpy is used as a generalized operation for performing tensor multiplications. Now, consider two matrices: $A = \begin{bmatrix} 2 & 8 \\ 4 & 3 \end{bmatrix}$ and $B = \begin{bmatrix} -9 & 9 \\ 0 & 11 \end{bmatrix}$. Then, what is the output of the following numpy operation?

numpy.einsum('ij,ij->', A, B)

Correct Answer: 87

Explanation:

The einsum notation 'ij,ij->' defines the operation:

1. **ij,ij**: The two inputs are both 2D matrices (indexed by i and j). Because the indices are identical for both inputs, this specifies an **element-wise multiplication** (also known as the Hadamard product).
2. **->**: The right side of the arrow is empty, which means the output should be a scalar (0-dimensional). This implies that we must sum over all indices that appear in the input but not the output (in this case, both i and j).

Therefore, the operation is an element-wise multiplication of A and B, followed by a sum of all the elements in the resulting matrix.

Step 1: Element-wise Multiplication (A⊙B)

$A \odot B = \begin{bmatrix} -18 & 72 \\ 0 & 33 \end{bmatrix}$

Step 2: Sum all elements

Sum = (−18) + 72 + 0 + 33 = 54+33 = 87