1. Suppose you are using sequence to sequence RNN model with attention. Your hidden layer is of size 2 (both encoder and decoder). Suppose the encoder has 4 hidden states (2-dimensional) [1,-1], [-1,1], [0,-1], [-1,0] and the decoder hidden state at a given time point is [-1,-1]. Using attention (dot product) over the encoder states, what will be the context vector to the next hidden state of the decoder?

2. Suppose you are pretraining a BERT model with 8 layers, 768-dim hidden states, 8 attention heads, and a sub-word vocabulary of size 40k. Also, your feed-forward hidden layer is of dimension 3072. What will be the number of parameters of the model? Include the embedding and positional embedding parameters as well. You can ignore the bias terms, and other parameters used corresponding to the final loss computation from the final encoder representation. The BERT model can take at most 512 tokens in the input.

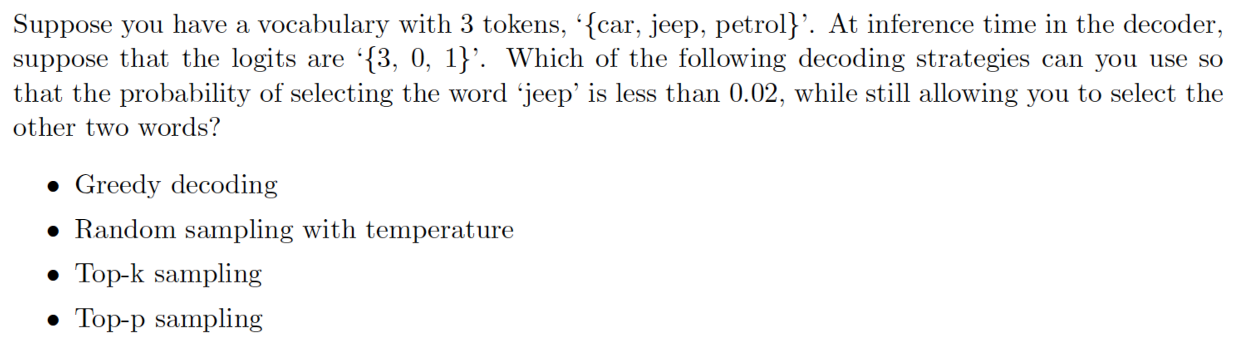
3. Consider below your dictionary containing 4 words, along with their frequencies in the corpus. Suppose you are using BPE to create a vocabulary. (a) What will be the initial vocabulary? (b) What will be the 3 tokens that you will add to your vocabulary next, in that order?

Word	**Frequency**
enter	3
entry	5
tenth	6
cant	2

4. Answer the following questions

Suppose you are training a transformer encoder-decoder for the task of machine translation from source ‘English’ to target ‘Bengali’. Assume that the source vocabulary is 30k, while the target vocabulary is 40k. Across encoder and decoder, you are using model-dimensionality of 1,024 and the feed-forward hidden layer has 2,048 units. Assume that there are 16 attention heads for all kinds-of-attention, as per the default setup. Also, assume that both encoder and decoder use the (same) trainable positional embeddings, and the maximum sequence length is 512.

However, while the encoder uses 24 layers, you only use 12 layers for the decoder. How many parameters does the model has? Assume that there is weight tying for ‘unembedding’.



Suppose you have a vocabulary with 3 tokens, '{car, jeep, petrol}'. At inference time in the decoder, suppose that the logits are '{3, 0, 1}'. Which of the following decoding strategies can you use so that the probability of selecting the word 'jeep' is less than 0.02, while still allowing you to select the other two words?

- Greedy decoding
- Random sampling with temperature
- Top-k sampling
- Top-p sampling

Also inform appropriate hyper-parameters for any of the sampling algorithms that would be applicable.