## Week 1a – N-Gram Language Models

1. What does the abbreviation GPT stand for?
   1. Generative Pretrained Transformer
2. What is a main difference between causal and masked language models?
   1. Causal predicts the next word in a sequence, while masked predicts an arbitrary missing word. (Causal and masked language models are the two main formulations of language models.)
3. What would be the \*least\* suitable application for a language model?
   1. Identify photographs of textbooks. (Although progress is being made to join language and image models together, you’re best off using image recognition for this application.)
4. What Python code snippet correctly computes bigrams in a given sentence?
   1. bigrams = [(sentence[i], sentence[i+1]) for i in range(len(sentence)-1)]
5. What is the main difference between unigram, bigram, and trigram models?
   1. The difference lies in the number of words they consider for prediction.
6. What does the 'n' in n-gram represent?
   1. The number of words considered for prediction
7. What are the trigrams that form the sentence: "I am studying GPT models"?
   1. [("I", "am", "studying"), ("am", "studying", "GPT"), ("studying", "GPT", "models")]
8. Analyse the following statement: "The bigger the 'n' in an n-gram model, the better the model's performance."
   1. The statement is true up to a certain point, after which the probabilities are difficult to estimate. (Increasing 'n' beyond a certain point can result in n-grams that are infrequent and therefore challenging to estimate well.)

## Week 1b – Evaluating Language Models

1. You are building a language model for a specific task and you want to evaluate its performance. Which of the following strategies would you use?
   1. Measure the model's perplexity on a held-out evaluation dataset. (Measuring perplexity on an evaluation set is a standard approach for evaluating language models.)
2. What can be inferred if a language model has a high perplexity value?
   1. The model is not accurately predicting the probability distribution of the next word.
3. Why is perplexity a suitable measure for evaluating language models?
   1. It measures the average likelihood of the next word given the previous words. (Perplexity measures how well a language model predicts a sample.)
4. What is a commonly used metric for evaluating the intrinsic performance of language models?
   1. Perplexity. (Perplexity is a measure of how well a language model predicts sample text.)
5. Why do we often use a formulation of perplexity that adds the logarithms of probabilities instead of multiplying the probabilities directly?
   1. Multiplying many small numbers can lead to floating point underflow. (Since the probabilities from language models tend to be small, multiplying them together can often result in values too small to accurately represent using typical floating point values.)
6. When performing an extrinsic evaluation, you do not need to worry about training on your test data.
   1. False. (Especially now that language models consume vast amounts of data, it’s important to check whether extrinsic evaluations are valid due to test data leakage.)
7. In the context of language models, what is an intrinsic held-out evaluation used for?
   1. To evaluate the model's performance on unseen data. (The held-out set is a portion of the dataset that the model has not been trained on, used to evaluate the model's performance on unseen data.)
8. What is the primary difference between intrinsic and extrinsic evaluation methods for language models?
   1. Intrinsic evaluation measures the quality of the language model's outputs in isolation, while extrinsic evaluation measures its performance in the context of an end-task.

## Week 1c – Text Generation with Language Models

1. If you were generating text and your requirement is to generate a text sequence that is as diverse as possible, which method would you use?
   1. Sampling. (By sampling randomly from the probability distribution, the generated sequences tend to be more diverse and interesting.)
2. How does beam search differ from greedy generation and sampling in terms of sequence generation?
   1. It considers multiple probable sequences before deciding the next word. (Beam search maintains multiple “beams”, each of which is a probably sequence.)
3. What is a potential downside of using beam search for text generation?
   1. It may produce sequences are not very interesting due to lack of diversity. (Often highly-probable text sequences are not very interesting.)
4. Which of these statements related to the consistency of responses from different generation strategies is correct?
   1. Greedy generation and beam search always generate the same response to the same input. (Both of these strategies are deterministic.)
5. What is a key difference between sampling and greedy generation?
   1. Sampling chooses the next token randomly based on a given probability distribution, while greedy generation always chooses the most probable next token.
6. What best defines the greedy generation method from language models?
   1. The method that generates sequences by always choosing the most probable next token.
7. Assume that you are generating text from an n-gram language model and you want the output to be efficient and deterministic, but also generate a higher-probability sequence than greedy search. Which method would be the most suitable?
   1. Beam search. (Beam search is deterministic and will always generate a sequence at least as probably as a greedy search.)
8. When generating text using a greedy strategy, why might you want to terminate the process after a fixed number of tokens, rather than waiting for an “end-of-sequence” token?
   1. Depending on the language model, and “end-of-sequence” token may never be generated. (N-gram language models are especially likely to get stuck in repetitive generation loops that do not contain an “end-of-sequence” when using a greedy generation approach.)

## Week 2a – Tokenisation

1. You can try swapping different tokenizers and keep the same Transformer model
   1. False. (The matching tokenizer must be used alongside the corresponding Transformer model. Transformer models expect a certain vocabulary size and mapping from text to token IDs.)
2. Why is it necessary to add context to word vectors?
   1. Some words have different meanings in different contexts. (Words like “match” can have different meanings in different contexts which static word vectors without context cannot account for.)
3. What do the GPT models use the Ġ character for?
   1. Indicate that the token is at the start of a word. (Language models that use subword tokenization need to distinguish between tokens that start words and those inside words. GPT models use the Ġ character to denote tokens that have whitespace before and start a word. Other language models may use other conventions.)
4. Why do some language models use a special padding token?
   1. To deal with multiple inputs of different lengths. (Language models often process multiple sequences together which may be of different lengths. To enable this and indicate that some of the sequence is not important, padding special tokens may be added to the end of shorter sequence to make them the same length as the longest sequence.)
5. The similarity of letters between two words is the best signal of word similarity.
   1. False. (Two words can have very similar letters (e.g. match and watch) but have very different meanings.)
6. Given these tokens and this vocabulary, what is the output of the tokenizer (ignoring any special characters for denoting subword tokens)?

Tokens = ‘the dog ate quickly’

Vocab = {‘a’:0, ‘ate’:1, ‘cat’:2 ,‘dog’:3, ‘quickly’:4, ‘slowly’:5, ‘the’:6, ‘took’:7}

* 1. [6, 3, 1, 4] (The four tokens are mapped to the four token identifiers.)

1. Context vectors can be used directly for other tasks such as sentiment prediction.
   1. True. (Context vectors can encode enough information about text to be used for classification problems such as sentiment prediction.)
2. Tokenizers always remove punctuation
   1. False. (Punctuation may be integral for the meaning of a sequence of text. Tokenizers for Transformers keep the punctuation and may process it into separate tokens or include it with common words.)
3. Why are new tokens a challenge for a language system?
   1. No word vectors have been calculated for the token. (Language systems store word vectors (also known as word embeddings) for all tokens and a new token would not have one precalculated.)
   2. No prior knowledge so don’t know how to use them. (A language model that has not seen a token before would not have information about the contexts in which it appears or the function that it plays. This would inhibit processing it well.)

## Week 2b – Neural Language Models and Transformers.

1. Why are probabilities often calculated with log transformations?
   1. Computers suffer from numerical problems when dealing with very small numbers
2. What function should be used to get probabilities from a set of scores?
   1. Softmax
   2. Sigmoid
3. For a given sequence of text with 3 tokens and language model, the perplexity is calculated as 4. What is the probability calculated for the whole sequence?
   1. 1/4
   2. 1/3
   3. 1/16
   4. 1/64
4. Given these three vectors, which are the most similar (using Euclidean distance or cosine similarity)? A=[1,1] B=[1,2] C= [9,1]
   1. A and B are the most similar
   2. A and C are the most similar
   3. B and C are the most similar
   4. They are all equally spaced
5. Perplexity for text A is lower than text B. What does this mean?
   1. The language model was more likely to have generated text A than B
6. The probability of a language model generating a sequence of 2 tokens is 1/36. What is the perplexity?
   1. 2
   2. 3
   3. 6
   4. 12
7. A language model has a vocabulary of 10 tokens and generates each token with equal probability, ignoring any context. What is the probability of a specific three token sequence?
   1. 0.1
   2. 0.01
   3. 0.001
   4. 0.0001
8. A language model only ever predicts the next token to be “fish”. What perplexity would it get for a sequence “fish fish fish”.
   1. 0
   2. 0.333
   3. 1
   4. 3