Quizzes

Natural Language Processing (2025)

This document contains the quizzes for the lectures, along with correct solutions and brief explanations. The explanations were drafted using ChatGPT in January 2025 and then manually reviewed and corrected.

Unit 1

Lecture 1.1

1. What is the datatype of token ID vectors?

   - `list[str]`

     Incorrect. While token IDs may conceptually represent strings in some cases (e.g., when they represent words), the datatype of token ID vectors is not a list of strings. Token IDs are numerical representations.

   - `Tensor[int]`

     Correct. Token ID vectors are represented as integer tensors, where each integer corresponds to a unique token in the vocabulary.

   - `Tensor[float]`

     Incorrect. Token ID vectors are integers, not floating-point numbers. Floating-point tensors are typically used for embeddings, not IDs.

2. Whitespace tokenisation can suffer from undersegmentation. What is an example of oversegmentation?

   - `co–writer/director → co – writer / director`

     Correct. Oversegmentation occurs when a single meaningful token is split into multiple, less meaningful tokens. In this case, `co–writer/director` is split unnecessarily into five tokens, including the less meaningful `co`, `–`, `writer`.

   - `co–writer/director → co–writer / director`

     Incorrect. This is an example of reasonable segmentation, where tokens are split at logical boundaries without breaking meaning.

   - `co–writer/director → co–writer/director`

     Incorrect. This is an example of no segmentation at all, which may not capture subunits for downstream tasks.

3. Suppose we apply the regex-based tokeniser to the negative review. Which tokens would we get? (Separate tokens are enclosed in square brackets.)

   - `[relentless] [gaiety] [of] [the] [1920's] [,]`

     Incorrect. This segmentation keeps ”1920's” as a single token, which does not match the regex-based tokeniser behavior of splitting contractions and possessive markers.

- `[relentless] [gaiety] [of] [the] [1920] ['s] [,]`

  Correct. The regex-based tokeniser splits possessive markers (e.g., `Jackson's`) into separate tokens, which matches this output.

- `[relentless] [gaiety] [of] [the] [1920] ['] [s] [,]`

  Incorrect. This segmentation splits `'s` into two tokens, which does not match the regex shown in the slide.

4. What is an example of lemmatisation?

   - `centers → center`

     Correct. Lemmatisation reduces words to their base or dictionary form (lemma). Here, `centers` is reduced to `center`.

   - `center → centre`

     Incorrect. This is not lemmatisation but a change of spelling (e.g., American to British English).

   - `center → centers`

     Incorrect. This represents inflection (e.g., singular to plural), not lemmatisation.

5. In the graph for the Heaps' law experiment, after how many tokens, approximately, had the vocabulary reached half of its final size?

   - 250,000

     Correct. At 250,000 tokens, the vocabulary size is slightly more than 15,000 tokens, whereas the final size is slightly more than 30,000 tokens.

   - 500,000

     Incorrect. At 500,000 tokens, the vocabulary size is approximately 22,500 tokens, more than half the final size of slightly more than 30,000 tokens.

   - 1,000,000

     Incorrect. At 1,000,000 tokens, the vocabulary size is approximately 28,000 tokens, quite close to the final size of slightly more than 30,000 tokens.

Lecture 1.2

1. In BPE, how many merge operations give us a vocabulary of 1,024 tokens?

   - 256

     Incorrect. With only 256 merge operations, the resulting vocabulary size would be 256 + 256 = 512 tokens.

   - 768

     Correct. If we start with a base vocabulary of size 256 (all possible single bytes), adding 768 merges gives us a total of 1,024 tokens. Each merge adds one token to the vocabulary.

   - 1,024

     Incorrect. 1,024 merge operations would create a vocabulary with 1,280 tokens when starting from a base vocabulary of 256.

2. Unicode codepoints between 2048 and 65535 need three bytes in UTF-8. How many bytes are in the UTF-8 encoding of the Unicode string oó□, where the last letter is the letter O in Devanagari?

   - 4

     Incorrect.

   - 5

     Incorrect.

   - 6

     Correct. The UTF-8 encoding is as follows: o → 1 byte (7-bit ASCII), ó → 2 bytes (cf. the example from Icelandic), □ → 3 bytes (cf. the Unicode chart, where this letter has codepoint 0913, corresponding to 2323). Adding these gives a total of 6 bytes.

3. Suppose we apply the BPE algorithm to the string aaaa and do at most four merges. What is the second merge rule extracted by the algorithm?

   - `a + a → [aa]`

     Incorrect. This is the first merge rule extracted by the algorithm, not the second.

   - `[aa] + a → [aaa]`

     Incorrect. The second merge rule is not applied to as string with occurrences of both `[aa]` and a but to the result of the first merge rule, which is `[aa][aa]`.

- `[aa] + [aa] → [aaaa]`

  Correct. After the first merge rule, the string becomes `[aa][aa]`. The second merge rule is then `[aa] + [aa] → [aaaa]`.

4. Suppose we apply the BPE algorithm to a very long English text and do a lot of merge rules. Which token would we expect *not* to see in our final vocabulary?

   - `[eaux]`

     Correct. This should be a rare token even in very long English texts, especially compared to the other two.

   - `[tion]`

     Incorrect. This token is a very frequent subword in English (e.g., *action*, nation) and is likely to appear in the final vocabulary after many merges.

   - `[thes]`

     Incorrect. This token may not be as frequent as `[tion]` but appears in several English words (e.g., *hypothesis*, *anaesthesia*) and should be much more likely than `[eaux]`.

5. How many tokens are in the string `Linköping University`, according to GPT-4?

   - 2

     Incorrect.

   - 3

     Incorrect.

   - 4

     Correct. According to Tiktokenizer, the `o200k_base` tokeniser of GPT-4 tokenises this string as `Link`, `ö`, `ping`, and `University`.

Lecture 1.3

1. We initialise an embedding layer `e` as `torch.nn.Embedding(15000, 100)`. What is the number of trainable parameters in `e`?

   - 15,000

     Incorrect. This value corresponds to the number of unique embeddings, not the total number of parameters in the layer.

   - 15,100

     Incorrect. This does not match the formula for calculating the total number of parameters in an embedding layer.

   - 1,500,000

     Correct. The number of trainable parameters in the embedding layer is given by the formula: *number of embeddings* $\times$ *embedding size*. Here, $15000 \times 100 = 1500000$.

2. We build a continuous bag-of-words classifier using the embedding layer `e` from the previous question and a softmax classifier. Assuming five classes and no bias, what is the correct formula for the number of trainable parameters of the classifier?

   - number of parameters in `e` + 105

     Incorrect. Adding 105 parameters would only account for a softmax classifier with 5 classes and 21 inputs, which does not match the 100-dimensional embeddings.

   - number of parameters in `e` + 500

     Correct. For a softmax classifier with 100 inputs and 5 output classes, the weight matrix contains $100 \times 5 = 500$ parameters.

   - number of parameters in `e` + 7500

     Incorrect.

3. True or false? "The embeddings learned for the words university and school are similar."

   - True

     Incorrect. While embeddings for related words can be similar, their similarity depends on the specific training task.

   - False

     Incorrect. This assumption cannot always be made, as similarity depends on the task.

- Depends on the training task

  Correct. The similarity between embeddings for *university* and *school* depends on the nature of the training task and dataset. For example, in a task focusing on education, they may be similar, but in a task unrelated to education, they may not.

4. We train a continuous bag-of-word classifier on the task of predicting the category of a product as either "book" or "bike" based on a product description. Which of the following word pairs can be expected to have embeddings with low cosine similarities?

   - *pages*, *gear*

     Correct. The words *pages* and *gear* belong to very different semantic categories ("book" and "bike", respectively), so their embeddings are expected to have low cosine similarity.

   - *pages*, *chapter*

     Incorrect. Both words are strongly related to the "book" category, so their embeddings are likely to have high cosine similarity.

   - *gear*, *brake*

     Incorrect. Both words are strongly related to the "bike" category, so their embeddings are likely to have high cosine similarity.

5. Which of the following is an instance of "pre-training and fine-tuning"?

   - initialise the embedding layer with random weights + train the network on a prediction task

     Incorrect. This describes standard training without any pre-training step.

   - initialise the embedding layer with trained weights + train the full network on a prediction task

     Correct. Pre-training refers to using weights trained on a related task or large corpus, and fine-tuning involves further training on a specific task. This description matches that process.

   - initialise the embedding layer with trained weights + train the other parts on a prediction task

     Incorrect. While this involves pre-trained embeddings, freezing the embedding layer and training only other parts of the network does not fully qualify as fine-tuning.

Lecture 1.4

1. Say that we have a vocabulary of half a million English words, and suppose that the word *parsnip* has the number 350,740. How long is the one-hot vector for *parsnip*?

   - 1

     Incorrect. The value 1 would be the length of the vector if there were only one word in the vocabulary, but this is not the case here.

   - 350,740

     Incorrect. This value represents the index of the word *parsnip* in the vocabulary, not the length of the one-hot vector.

   - 500,000

     Correct. In a one-hot representation, the length of the vector corresponds to the size of the vocabulary. Since there are 500,000 words, the vector is 500,000 elements long.

2. What is a typical length for a word embedding?

   - 30

     Incorrect. A length of 30 would be too short for word embeddings to effectively capture the complexity of semantic relationships.

   - 300

     Correct. A typical word embedding, such as those used in `word2vec` or GloVe, often has a length of 300 dimensions, which provides a good balance between representational power and computational efficiency.

   - 30,000

     Incorrect. This value is far too large for a word embedding, leading to unnecessary computational costs and potential overfitting.

3. What does it mean if two vectors have a cosine similarity of 1?

   - The angle between them is 0 degrees.

     Correct. A cosine similarity of 1 indicates that the vectors are perfectly aligned, meaning the angle between them is 0 degrees.

   - The angle between them is 180 degrees.

     Incorrect. An angle of 180 degrees corresponds to a cosine similarity of -1, indicating that the vectors are perfectly opposite.

- The angle between them is 90 degrees.

  Incorrect. An angle of 90 degrees corresponds to a cosine similarity of 0, indicating that the vectors are orthogonal (no similarity).

4. The Distributional Hypothesis is often summarised in the slogan "You shall know a word by the company it keeps". What does the word "company" refer to?

   - Words that co-occur with the other word

     Correct. The "company" refers to the words that frequently co-occur in similar contexts. This is the basis of the Distributional Hypothesis, which suggests that the meaning of a word can be inferred from its context.

   - Words with similar word embeddings

     Incorrect. While word embeddings can encode relationships between words, the *company* specifically refers to co-occurring words in context.

   - Words with similar meanings

     Incorrect. Words with similar meanings often have similar co-occurrences, but the *company* directly refers to the surrounding words, not their meanings.

5. Which of the following is *not* used to evaluate word embeddings?

   - cross-entropy loss

     Correct. Cross-entropy loss is used during training to optimise word embeddings, but it is not a direct evaluation metric for the quality of embeddings.

   - odd-one-out test

     Incorrect. The odd-one-out test evaluates embeddings by determining which word does not fit in a group, assessing semantic relationships.

   - analogy benchmarks

     Incorrect. Analogy benchmarks, such as "king – man + woman = queen", are commonly used to evaluate how well embeddings capture relationships between words.

Lecture 1.5

1. For the Simple English Wikipedia corpus, the vocabulary size is roughly 15,000 words. How many entries does the co-occurrence matrix for this corpus have?

   - 15,000

     Incorrect. This corresponds to the number of unique words in the vocabulary, not the number of entries in the co-occurrence matrix.

   - 30,000

     Incorrect.

   - 225,000,000

     Correct. The co-occurrence matrix is a square matrix of size $15,000 \times 15,000$. The total number of entries is $15,000 \times 15,000 = 225,000,000$.

2. Suppose we take this co-occurrence matrix and compute 100-dimensional word embeddings using a truncated SVD. How many entries does the matrix $\boldsymbol{U}$ have?

   - 10,000

     Incorrect.

   - 1,500,000

     Correct. The matrix $\boldsymbol{U}$ has $15,000$ rows (equal to the vocabulary size) and 100 columns (equal to the target dimensionality). The number of entries is $15,000 \times 100 = 1,500,000$.

   - 22,500,000,000

     Incorrect. This value corresponds to the full size of the co-occurrence matrix, not the size of the reduced matrix $\boldsymbol{U}$.

3. We compute the truncated SVD of a co-occurrence matrix $\boldsymbol{M}$ and target dimensionality $d = 100$. Which of the following statements is correct?

   - The matrix $\boldsymbol{U}$ has 100 columns.

     Correct. After the truncated SVD, the matrix $\boldsymbol{U}$ will have dimensions $15,000 \times 100$, where 100 is the target dimensionality.

   - The matrix $\boldsymbol{V}$ has 100 rows.

     Incorrect. After the truncated SVD, the matrix $\boldsymbol{V}$ will have 100 columns, not rows. It is the transpose of the right singular vectors.

- The matrix $\boldsymbol{\Sigma}$ has 100 zero entries.

  Incorrect. The truncated SVD only keeps the top $d = 100$ singular values, and all other (9900) entries are zeros.

4. If we double the size of the vocabulary, the runtime of the truncated SVD …
   - … will double

     Incorrect. Doubling the vocabulary size increases the size of the matrix quadratically, so the runtime will not simply double.
   - … will grow by a factor of 4

     Correct. The co-occurrence matrix is square, so doubling the size of the vocabulary increases the number of entries in the matrix by a factor of $2 \times 2 = 4$. Since SVD operates on the entire matrix, the runtime grows by a factor of 4.
   - … will grow exponentially

     Incorrect. While the growth is quadratic, it is not exponential.

5. Consider a large corpus in which a word w occurs only once, in the context of another word c (and no other word). What can we say about $\text{PPMI}(w, c)$?
   - It is high.

     Correct. Since w and c co-occur only once and no other context exists for w, the pointwise mutual information (PMI) is high because the observed co-occurrence probability $P(w, c)$ is non-negative and $P(w)$ is very small.
   - It is low.

     Incorrect. Low PMI indicates that w and c co-occur less frequently than expected, which is not the case here.
   - It is close to zero.

     Incorrect. A PMI close to zero indicates no significant relationship between w and c, which is not the case since they co-occur exclusively.

Lecture 1.6

1. The standard skip-gram model (without negative sampling) is a k-class classification problem. What would be a realistic value for k?

   - 2

     Incorrect. $k = 2$ would only apply to binary classification, not a multi-class task like the standard skip-gram model, where the model predicts context words from a large vocabulary.

   - 2,000

     Incorrect. This value is not realistic for the full vocabulary size in most corpora.

   - 20,000

     Correct. In the standard skip-gram model, k corresponds to the size of the vocabulary, which is typically tens of thousands.

2. Why is the task of predicting the other word in a skip-gram an interesting pre-training task?

   - Predicting the other word forces the model to learn co-occurrence statistics

     Correct. Predicting context words requires the model to capture co-occurrence patterns, which are key to learning meaningful word embeddings.

   - Predicting the other word can be done efficiently

     Incorrect. While efficiency is desirable, it is not the primary reason the task is interesting for pre-training.

   - Predicting the other word is a relatively simple task that can be learned with little data

     Incorrect. Skip-gram models require large amounts of data to effectively learn word representations, as they rely on co-occurrence patterns across diverse contexts.

3. In the skip-gram model, we distinguish between a target word w and a context word c. When would we want $P(c \mid w)$ to be high?

   - The word vectors for w and c have a high cosine similarity

     Correct. We want to tune the word vectors in such a way that $P(c \mid w)$ is high, and this requires them to have high cosine similarity.

   - The word vectors for w and c have a low cosine similarity

     Incorrect.

- The two words w and c are frequent

  Incorrect. Word frequency does not directly determine $P(c|w)$. Instead, the co-occurrence relationship between w and c matters.

4. What is the basic idea behind the "skip-gram with negative sampling" model?

   - approximate the probabilities $P(c \mid w)$ using a simpler prediction task

     Correct. Negative sampling approximates $P(c \mid w)$ by replacing the computationally expensive softmax with a binary classification task: distinguishing true (positive) context words from randomly sampled (negative) ones.

   - decompose the softmax into a tree-like structure of simpler computations

     Incorrect. This describes hierarchical softmax, which is another method for optimizing $P(c \mid w)$, not negative sampling.

   - replace the word embeddings with an efficient sampling process

     Incorrect. Negative sampling retains word embeddings but uses an efficient sampling-based approximation for training.

5. Which of the following is *not* used to speed up the "skip-gram with negative sampling" model?

   - randomly exclude stop words and other non-content words from the training data

     Correct. Excluding stop words is a common preprocessing step that can speed up training by reducing unnecessary computations, but is not used specifically in skip-gram with negative sampling.

   - randomly discard tokens with a probability that grows with the token frequency

     Incorrect. This is the subsampling technique implemented in skip-gram with negative sampling, which helps speed up training by reducing the impact of frequent words.

   - randomly sample window sizes up to a maximum size with uniform probability

     Incorrect. Rather than considering all window sizes up the maximum size, the model samples these sizes at random to speed up the training.

Unit 2

Lecture 2.1

1. What component would you *not* typically need in a pipeline for interlingual machine translation?

   - part-of-speech tagger

     Incorrect. A part-of-speech tagger helps identify grammatical categories of words, which can be useful in an interlingual MT pipeline.

   - syntactic dependency parser

     Incorrect. A syntactic dependency parser provides structural relationships between words, which is often essential for constructing an interlingual representation.

   - sentiment classifier

     Correct. A sentiment classifier is not directly necessary for translation, as it is mainly used to analyse emotions in text rather than its syntactic or semantic structure.

2. In the Noisy Channel Model, which of the following quantities do we want to maximize when translating from Arabic (A) to Swedish (S)?

   - $P(A|S)P(S)$

     Correct. This is the decomposition of $P(S|A)$ using Bayes' Rule.

   - $P(S|A)$

     Incorrect. This expresses the probability of a target-language sentence given a source-language sentence, but the Noisy Channel Model rewrites this using Bayes' rule.

   - $P(A|S)$

     Incorrect. This represents the likelihood of the source sentence given the target sentence, but it does not consider how likely the target sentence is in the target language.

3. Which of the following statements is true about word alignments when viewed as a mathematical relation R between the set of positions on the source side and the set of positions on the target side?

- R is a function.

  Incorrect. A function requires that each input (source position) maps to exactly one output (target position), but in word alignment, a single source word can align to multiple target words.

- The inverse of R is a function.

  Incorrect. The inverse of R would mean each target position maps to exactly one source position, but this is also not true in word alignment.

- Neither of these statements is true.

  Correct. Word alignment is a relation, not necessarily a function, because it allows one-to-many and many-to-one mappings.

4. Which of the following is an advantage of neural machine translation (NMT) over statistical machine translation (SMT)?

   - NMT systems do not need complex feature engineering.

     Correct. Unlike SMT, which relies on manually designed features, NMT automatically learns representations from data.

   - NMT systems can be trained without parallel text.

     Incorrect. Like SMT, NMT requires parallel text for supervised training, although some unsupervised approaches exist.

   - NMT systems are more interpretable than SMT systems.

     Incorrect. NMT models are often criticised for their lack of interpretability compared to SMT models, which have explicit probabilistic structures.

5. Why does the BLEU evaluation measure include a brevity penalty?

   - It is easy to achieve high precision with short translations.

     Correct. Short translations can match many n-grams from the reference translation while omitting important content, inflating the BLEU score without truly reflecting translation quality.

   - It is easy to achieve high recall with short translations.

     Incorrect. Recall measures how much of the reference translation is captured, and short translations tend to have low recall.

   - Short translations should be penalised because they are typically not very informative.

     Incorrect. While short translations may be less informative, the BLEU brevity penalty is specifically designed to counteract artificially high precision scores.

Lecture 2.2

1. Which of the following tasks do *not* usually lend themselves to the use of autoregressive language models?

   - machine translation

     Incorrect. Machine translation is commonly handled by autoregressive models, where each token is generated based on previously generated tokens.

   - text summarisation

     Incorrect. Autoregressive models are widely used for text summarisation, where they generate a summary token by token.

   - document classification

     Correct. Document classification is not typically performed using autoregressive models, as it does not require sequential token generation.

2. Suppose we translate from Arabic (A) to Swedish (S). Which of the following quantities does a neural sequence-to-sequence model learn?

   - $P(S|A)$

     Correct. Neural sequence-to-sequence models learn the conditional probability $P(S|A)$, which models the probability of a target sequence given a source sequence.

   - $P(A|S)$

     Incorrect. This represents the probability of the source sentence given the target sentence, which is not what sequence-to-sequence models learn.

   - $P(A, S)$

     Incorrect. This represents the joint probability of both sentences occurring together, which is not explicitly modelled in neural translation systems.

3. Which of the following resources do we need in order to train sequence-to-sequence translation models?

   - parallel texts for the source language–target language pair

     Correct. Training a sequence-to-sequence model requires parallel text corpora to learn correspondences between the source and target language.

   - word alignments between the words in the source language and target language

     Incorrect. Word alignments are useful for phrase-based statistical translation models but are not directly required for training modern neural translation models.

- a language model for the target language

  Incorrect. While a separate target-language model can help in some hybrid approaches, sequence-to-sequence models inherently learn a target language model during training.

4. Which of the following parameters does *not* impact the space complexity of beam search?

   - number of possible translations for the source sentence

     Correct. Beam search only keeps track of a fixed number of hypotheses, independent of the total number of possible translations.

   - width of the beam

     Incorrect. A wider beam requires storing more hypotheses, increasing space complexity.

   - lengths of the generated target sentences

     Incorrect. Longer sentences require more memory storage, impacting space complexity.

5. Why do we use length normalisation together with beam search in decoding?

   - We do not want to penalise long translations.

     Correct. Without length normalisation, longer translations can be disproportionately penalised because the probability of a sequence decreases as more tokens are added. Length normalisation mitigates this bias.

   - We do not want to penalise short translations.

     Incorrect. Standard beam search often *favours* shorter translations because the probability of a sequence decreases as more tokens are added.

   - We want to avoid numerical overflow.

     Incorrect. Numerical overflow is not a major concern in beam search; the primary issue is the tendency of shorter sequences to have higher probability scores.

Lecture 2.3

1. Suppose we encode the sentence "Gold is heavier than silver" using a bi-directional recurrent neural network. What issue does the recency bias cause?

   - The final hidden state contains more information about *Gold* than about *heavier*.

     Correct. Recency bias means that more recent words influence the hidden state more strongly. Because we are working with a bi-directional recurrent neural network, the words *Gold* and *silver* are the most recent words.

   - The final hidden state contains more information about *Gold* than about *silver*.

     Incorrect. *Gold* and *silver* occur at the periphery of the sentence and are thus more likely to be overrepresented in the final hidden state of the bi-directional RNN due to recency bias.

   - The final hidden state contains more information about *silver* than about *Gold*.

     Incorrect for the same reason.

2. Which of the following statements about the use of attention for translation is true in the sequence-to-sequence model of Sutskever et al.?

   - The attention score of a source-language word is dependent on the previous hidden state of the decoder.

     Correct. In sequence-to-sequence models with attention, the attention score is computed based on the decoder's previous hidden state and the encoder's hidden states.

   - Attention quantifies how much we should attend to each target word when generating the next source word.

     Incorrect. Attention is used to determine how much to focus on each *source* word when generating the next *target* word.

   - The attention score will be high if the source-language word is similar to the target-language word.

     Incorrect. While similarity can play a role, attention scores depend on contextual relevance rather than direct similarity between words.

3. Consider the following values for the attention example in the lecture:

$$\begin{aligned} s &= [0.4685, 0.9785] \\ h_1 &= [0.5539, 0.7239] \\ h_2 &= [0.4111, 0.3878] \\ h_3 &= [0.2376, 0.1264] \end{aligned}$$

   Assuming that the attention score is computed using the dot product, what is v?

   - $[0.4387, 0.4855]$

     Correct. In self-attention, the context vector v must have the same dimensionality as each hidden state h_i.

   - $[0.9678, 0.5721, 0.2350]$

     Incorrect.

   - $[0.4643, 0.3126, 0.2231]$

     Incorrect.

4. Which of the following statements about the more general characterisation of attention in terms of queries, keys, and values is true?

   - The output has the same length as each value.

     Correct. In attention mechanisms, the output is a weighted sum of values, meaning it retains the same dimensionality as the values.

   - The query has the same length as each value.

     Incorrect. The query and value dimensions can differ depending on the model architecture.

   - Each key has the same length as each value.

     Incorrect. Keys and values do not necessarily have the same dimensionality, though they often do in standard self-attention.

5. Consider an instance of multi-head attention with 8 heads where the queries and keys have size 256. What would be the typical key length in each block's attention mechanism?

   - 256

     Incorrect. The total query/key size is 256, but it is divided across the 8 attention heads.

- 32

  Correct. The key length per head is $\frac{256}{8} = 32$ because the attention mechanism splits the dimensions evenly across multiple heads.

- 8

  Incorrect.

Lecture 2.4

1. Which of the following is the main advantage of the Transformer architecture over recurrent neural networks?

   - direct access to all elements in the input sequence

     Correct. Unlike RNNs, which process sequences sequentially, Transformers use self-attention, allowing each token to access all tokens in the input at once, leading to more efficient parallel computation.

   - significantly reduced need for training data

     Incorrect. While Transformers can leverage large datasets effectively, they typically require *more* data than RNNs to achieve good performance.

   - supports significantly more compact models

     Incorrect. Transformers often have a higher number of parameters than RNNs due to their attention mechanisms and feedforward layers.

2. Consider the example translation used to illustrate the Transformer architecture. Which of the following statements is *false*?

   - The final encoder representation of *drink* depends on the token embedding of *Kaffee*.

     Correct. The Transformer encodes the English sentence and uses the representations computed in this process to generate the German sentence – not the other way around.

   - The final encoder representation of *coffee* depends on the token embedding of *drink*.

     Incorrect. Self-attention ensures that each token's representation is influenced by other tokens in the sequence.

   - The final decoder representation of *Kaffee* depends on the final encoder representation of *coffee*.

Incorrect. In an encoder–decoder architecture, all representations computed in the encoder can influence the representations computed in the encoder through cross-attention.

3. The Transformer architecture uses three different variants of multi-head attention. Which one is used in the encoder?

   - self-attention

     Correct. The Transformer encoder uses self-attention to allow each token to attend to all other tokens in the input sequence.

   - masked self-attention

     Incorrect. Masked self-attention is used in the decoder to prevent attending to future tokens during training.

   - cross-attention

     Incorrect. Cross-attention is used in the decoder to attend to the encoder's output.

4. What is the purpose of layer normalisation?

   - centering and scaling the layer's input values

     Correct. Layer normalisation standardises the inputs to a layer by centering and scaling them to stabilizs.

   - squeezing the layer's input values into the interval $[0, 1]$

     Incorrect. Layer normalisation does not necessarily map values into the $[0, 1]$ range; it normalises them based on mean and variance.

   - down-scaling the layer's output values

     Incorrect. The normalisation occurs on the inputs, not the outputs, and it involves both centering and scaling rather than just down-scaling.

5. True or false: Permuting the input tokens to a Transformer encoder does not change the final token representations.

   - True

     Incorrect. The Transformer architecture relies on positional encodings to capture word order, so permuting tokens changes their representations.

   - False

     Correct. Transformers process input tokens in parallel, but they use positional encodings to maintain word order, meaning changing the order alters final representations.

- Depends on the input tokens

  Incorrect. While some word orders may yield similar representations, in general, changing the order affects the token representations due to positional encodings.

Lecture 2.5

1. What does the term generative pre-training refer to?
   - pre-training on a language modelling task

     Correct. Generative pre-training refers to training a model on a language modelling task where it learns to predict the next token in a sequence before fine-tuning it on specific downstream tasks.
   - pre-training with a generative probabilistic model

     Incorrect. While generative pre-training involves a generative model, it specifically refers to pre-training using autoregressive language modeling, not just any generative probabilistic approach.
   - pre-training on automatically generated text

     Incorrect. Pre-training is typically done on large corpora of natural text rather than automatically generated text.
2. Looking at the original GPT model architecture (Radford et al., 2018), what is the approximate number of trainable parameters in the FNN?
   - 4,718,592

     Correct. The feedforward neural network (FNN) layer in GPT-1 contains approximately 4,718,592 trainable parameters based on its architecture.
   - 589,824

     Incorrect.
   - 9,216

     Incorrect.
3. Suppose you want to fine-tune a GPT model on the Stanford Natural Language Inference dataset. What is the minimal number of parameters you need to update?
   - the number of parameters in the pre-trained GPT model

     Incorrect. Full model fine-tuning updates all parameters, but minimal tuning requires updating only part of the model.

- the number of parameters in the final Linear layer

  Correct. The minimal number of parameters that need to be updated corresponds to the final Linear layer, which maps the model's hidden representations to the output task.

- the sum of these two

  Incorrect. While full fine-tuning updates both, the minimal required update involves only the final Linear layer.

4. What do we mean when we say that GPT-3 exhibits zero-shot behaviour?

   - It can solve tasks without any task-specific fine-tuning.

     Correct. Zero-shot learning means that GPT-3 can perform tasks it was not explicitly trained on, relying only on its general language understanding.

   - It can solve tasks without any training.

     Incorrect. GPT-3 is extensively trained on a large corpus of text before demonstrating zero-shot capabilities.

   - It can solve tasks without receiving any input.

     Incorrect. GPT-3 requires input prompts to generate responses, even in a zero-shot setting.

5. What is a reasonable explanation for the observation that GPT-3 can translate from English to French?

   - The data sets used for pre-training contain example translations.

     Correct. GPT-3 is trained on large-scale internet text, which includes many instances of translations, allowing it to learn translation patterns.

   - The number of model parameters approaches the number of neurons in the human brain.

     Incorrect. While GPT-3 has more parameters than there are neurons in the human brain (175B vs. 86B), the model's translation ability is primarily due to the data it was trained on, not its parameter count.

   - The model has been trained based on feedback from professional translators.

     Incorrect. GPT-3 is trained using unsupervised learning on large text corpora rather than direct feedback from translators.

Lecture 2.6

1. What is the purpose of the segment encoding in BERT?
   - to distinguish between different segments of sentence pairs

     Correct. In BERT, segment embeddings are used to differentiate between two input sentences in tasks like next sentence prediction.
   - to distinguish between different segments of words

     Incorrect. Words in BERT are represented using WordPiece embeddings, not segment encodings.
   - to distinguish between different segments of the word embeddings

     Incorrect. Segment encodings operate at the sentence level, not at the embedding level.

2. BERT is pre-trained on the masked language modelling task. Why is this task not suitable for pre-training GPT models?
   - GPT is based on the Transformer decoder, and as such can only "look back".

     Correct. GPT is an autoregressive model that generates text sequentially, meaning it cannot use bidirectional context like masked language modeling in BERT.
   - Masked language modelling does not scale up to the number of parameters in GPT.

     Incorrect. The scalability of masked language modelling is not the limiting factor; the autoregressive nature of GPT is.
   - GPT is trained on individual sentences, not sentence pairs (like BERT).

     Incorrect. While GPT does not rely on sentence pairs, this is not the reason masked language modeling is unsuitable; rather, GPT's causal attention prevents it from utilising masked tokens.

3. What is the main purpose of the `[CLS]` token in BERT?
   - It is used as a representation of the complete input sentence pair.

     Correct. The `[CLS]` token provides a fixed-length representation of the entire input and is commonly used for classification tasks.
   - It is used as a padding token.

     Incorrect. Padding is handled separately and does not involve the `[CLS]` token.

- It is used for the masked language modelling task.

  Incorrect. Masked language modelling applies to other tokens, but the `[CLS]` token itself is not masked.

4. In masked language modelling, we generate training examples by randomly branching into one of three cases. Which of these would typically make the token representation at the selected position more dissimilar to the representations of the surrounding tokens?

   - replace the selected token with the `[MASK]` token

     Incorrect. The `[MASK]` token still allows the model to infer meaning from surrounding tokens.

   - replace the selected token with a random word

     Correct. Replacing a token with a random word disrupts contextual consistency, making the token representation more dissimilar to its surroundings.

   - not replace the selected token

     Incorrect. Keeping the original token preserves its contextual relationship with surrounding words.

5. Which advantage does replaced token detection have over masked language modelling?

   - It learns from all input tokens.

     Correct. Unlike masked language modelling, which modifies only a subset of tokens, replaced token detection operates on all tokens, improving data efficiency.

   - It only needs two classes.

     Incorrect. The advantage of replaced token detection is not related to the number of classification categories.

   - It does not need the `[MASK]` token.

     Incorrect. While replaced token detection avoids the need for a special masking token, its primary advantage is the ability to learn from all tokens.

Unit 3

Lecture 3.1

1. Which of the models mentioned in the slide on Decoder-based language models has the largest context length?

- GPT

  Incorrect. OpenAI's o3 model (release: 2025-01) has a context size of 200K tokens.

- Gemini

  Correct. Gemini 2.0 (release: 2025-01) has a context size of 1M tokens.

- DeepSeek

  Incorrect. DeepSeek R1 (release: 2025-01) has a context size of 128K tokens.

2. What are adapters not typically used for?

   - To adapt a pretrained language model to a new computing hardware.

     Correct. Adapters are designed to fine-tune models for new tasks or languages without retraining the entire model, not for adapting to different hardware.

   - To adapt a pretrained language model to a new task.

     Incorrect. Adapters are commonly used to fine-tune models for specific tasks efficiently.

   - To adapt a pretrained language model to a new language.

     Incorrect. Adapters can be employed to extend a model's capabilities to additional languages.

3. What speedup is quoted for the H200 architecture over the H100 architecture when it comes to inference with the Llama 2 70B model?

   - 1.9

     Correct. The H200 architecture offers a 1.9x speedup over the H100 for Llama 2 70B model inference.

   - 1.6

     Incorrect.

   - 1.4

     Incorrect.

4. What task is addressed in the SWE-bench benchmark?

   - resolving GitHub issues

     Correct.

   - strategic workflow extrapolation

     Incorrect.

- answering factual questions about Sweden

  Incorrect.

5. Which were the three best-performing models on the Chatbot Arena leaderboard?

   - Gemini, ChatGPT, DeepSeek

     Correct (as of the date when the slides were produced).

   - ChatGPT, DeepSeek, Step-2

     Incorrect.

   - DeepSeek, Llama 3, o1

     Incorrect.

Lecture 3.2

1. How does Adam solve the zig-zagging problem?
   - It keeps averages of past gradient directions.

     Correct.
   - It keeps averages of past gradient magnitudes.

     Correct. While momentum-based optimisers keep track of past gradient magnitudes to set adaptive learning rates, the main source of the zig-zagging problem are gradient directions.
   - It keeps averages of past gradient similarities.

     Incorrect. Adam does not compute averages based on gradient similarities; it focuses on magnitudes and squared gradients.
2. What is the total norm as referred to in the context of gradient clipping?
   - The norm of the vector containing all parameter norms.

     Correct.
   - The norm of the vector containing all parameters.

     Incorrect. Gradient clipping concerns the gradients' norms, not the parameters' norms.
   - The sum of the norms of all parameter vectors.

     Incorrect. In gradient clipping, the total norm is computed over all gradient norms as if they were concatenated into a single vector.
3. In the learning rate scheduler example, what is the learning rate after 280B tokens?
   - 0.006

     Incorrect. This learning rate is too high for the given point in training.
   - 0.0006

     Incorrect.
   - 0.00006

     Correct.
4. Consider a case of batch accumulation over three micro-batches with sizes [800, 1000, 600] where the summed micro-batch losses are [960, 1300, 900]. What is the loss over the full accumulation?

- 1.23

  Incorrect.

- 1.32

  Correct. The total loss is calculated by dividing the sum of all micro-batch losses by the sum of all micro-batch sizes:

  $$\frac{960 + 1300 + 900}{800 + 1000 + 600} = \frac{3160}{2400} \approx 1.3167$$

  Rounding to two decimal places gives 1.32.

- 1.33

  Incorrect.

5. To which of the following would we typically not apply weight decay?

   - the weights of the layer norms

     Correct. Weight decay is generally not applied to the weights of layer normalisation layers, as these parameters are crucial for maintaining the normalised distribution of activations.

   - the weights of the linear layers

     Incorrect. Applying weight decay to the weights of linear layers is a common regularisation practice to prevent overfitting.

   - the weights of the attention layers

     Incorrect. Weight decay is often applied to the weights within attention mechanisms to promote generalisation.

Lecture 3.3

1. Assuming a token/byte ratio of 0.75 for English text, how many tokens would we expect to be in the text-only version of the latest Common Crawl dump?
   - 600 billion

     Incorrect.
   - 6 trillion

     Correct. The text extracted from the latest Common Crawl dump (2024-51) comprises approximately 7.37 TiB. Converting terabytes to bytes and multiplying with 0.75 gives approximately 6 trillion tokens.
   - 67 trillion

     Incorrect.
2. Looking at the slide Impact of filtering, which set of filters is the second-best set?
   - C4 Filters

     Correct.
   - FineWeb

     Incorrect.
   - Gopher

     Incorrect.
3. Why is deduplication so important in dataset curation for pretraining LLMs?
   - It prevents overfitting, enhances data diversity, and reduces computational costs.

     Correct. Deduplication ensures that models do not memorise repeated data, promotes exposure to diverse information, and reduces unnecessary computational load.
   - It ensures that the model memorises frequently occurring text patterns for better accuracy.

     Incorrect. Memorising repeated patterns can lead to overfitting and reduced generalisation.
   - It increases the total dataset size, allowing the model to train on more tokens.

     Incorrect. Deduplication reduces the dataset size by removing redundant data, but this is beneficial for training efficiency and model performance.
4. What model is used in the educational quality classifier in FineWeb-EDU?

- linear regression

  Correct. The educational quality classifier is a simple linear regression model built upon embeddings obtained via the Snowflake-arctic-embed architecture.

- Transformer

  Incorrect. While Transformers are used to produce the *input* to the classifier (using the Snowflake-arctic-embed architecture), the classifier itself uses a linear regression model.

- recurrent neural network

  Incorrect.

5. For the largest ablation model considered, how many points of average accuracy is FineWeb-EDU better than standard FineWeb?

   - 2 points

     Correct.

   - 4 points

     Incorrect.

   - 6 points

     Correct.

Lecture 3.4

1. What is the estimated computational cost for the smallest GPT-2 model and a dataset consisting of 2.5B tokens?

   - 15,000,000,000

     Incorrect.

   - 310,000,000,000,000,000

     Incorrect.

   - 1,860,000,000,000,000,000

     Correct. The computational cost C for training a language model can be estimated using the formula $C = 6 \times P \times T$, where P is the number of parameters and T is the number of tokens. For the smallest GPT-2 model with 124 million parameters and a dataset of 2.5 billion tokens:

$$C = 6 \times 124 \times 10^6 \times 2.5 \times 10^9 = 1.86 \times 10^{18} \text{ FLOPs}$$

2. In the slide "Performance improves smoothly with scale", what does each point of the thick black line in the leftmost plot correspond to?

   - the model with the lowest test loss for a specific compute budget

     Correct. Each point on the thick black line represents a model that achieves the lowest test loss given a particular compute budget, illustrating the optimal trade-off between model size and training duration.

   - the model with the lowest test loss for a specific number of parameters

     Incorrect. The plot focuses on compute budgets rather than solely on the number of parameters.

   - the model with the lowest test loss for a specific training set size

     Incorrect. The plot is centered on compute budgets, not directly on training set sizes.

3. An 8x A100 system can do approximately 64 exaflops per 24 hours. How many tokens can we expect to train a small GPT-2 model (124M parameters) on during that time?

   - 8.6T

     Incorrect.

- 860B

  Incorrect.

- 86B

  Correct. Given the computational capacity ($C = 64 \times 10^{18}$ FLOPs) and the model size ($P = 124 \times 10^6$), we can estimate the number of tokens T as

  $$T = \frac{C}{6 \times P} = \frac{64 \times 10^{18}}{6 \times 124 \times 10^6} \approx 86 \times 10^9 \text{ tokens}$$

  This equates to approximately 86 billion tokens.

4. Based on the results from the Chinchilla paper, approximately how many tokens should we train on per model parameter in a compute-optimal model?

   - 6

     Incorrect.

   - 20

     Correct. Looking at the slide "Compute-optimal models", the optimal model size of 63B parameters is achieved for 1.4T tokens. This is approximately 20 tokens per parameter.

   - 100

     Incorrect.

5. Based on the results from the Chinchilla paper, which of the following models was least compute-optimal?

   - Gopher (280B)

     Incorrect. While Gopher was not compute-optimal, Megatron-Turing NLG is even less so.

   - GPT-3 (175B)

     Incorrect. GPT-3 also deviated from compute-optimal training, but not to the extent of Megatron-Turing NLG.

   - Megatron-Turing NLG (530B)

     Correct. Megatron-Turing NLG, with 530 billion parameters, was significantly undertrained relative to the compute-optimal guidelines suggested in the Chinchilla paper.

Lecture 3.5

1. How many trainable parameters does the GPT-3 model have?

   - 117 B

     Incorrect.

   - 175 B

     Correct.

   - 1542 B

     Incorrect.

2. What was the main insight that came with GPT-1?

   - Next word prediction is an effective pre-training strategy for many tasks in NLP

     Correct. GPT-1 demonstrated that unsupervised pre-training using next word prediction could significantly improve performance across various NLP tasks.

   - Even 117-M-parameter models can be effectively trained on suitable hardware

     Incorrect. While GPT-1 had 117 million parameters, the main insight was related to the effectiveness of unsupervised pre-training, not hardware capabilities.

   - Masked language modelling is more important than next sentence prediction

     Incorrect. This insight is associated with BERT, not GPT-1.

3. Consider the Winograd example in the slide on zero-shot learning. Which of the following would be the matching prompt for a next-word distribution where $p(\textit{demonstrators}) > p(\textit{councilmen})$ and $\textit{they} = \textit{demonstrators}$?

   - The city councilmen refused the demonstrators a permit because they advocated violence.

     Correct. In this sentence, *they* refers to *demonstrators*.

   - The demonstrators attacked the city councilmen because they refused them a permit.

     Incorrect. Here, *they* refers to *councilmen*, not *demonstrators*.

   - The city councilmen refused the demonstrators additional permits because they were too expensive.

     Incorrect. In this context, *they* refers to *permits*, not *demonstrators*.

4. How is in-context learning different from zero-shot learning?

- The model can learn new tasks from examples.

  Correct. In-context learning involves providing the model with examples within the input context, enabling it to adapt to new tasks without parameter updates.

- The model has more than one attempt at predicting the next word.

  Incorrect. Both in-context and zero-shot learning involve single forward passes for predictions.

- The model can update its gradients based on the words in the context of the prediction.

  Incorrect. In-context learning does not involve gradient updates; the model adapts based on the input context alone.

5. In the slide on prompt engineering, how much better was the LLM-designed prompt compared to the best human-designed prompt?

   - 3.3 points

     Correct. The LLM-designed prompt outperformed the best human-designed prompt by 3.3 points, indicating a measurable improvement in performance.

   - 3.3 percent

     Incorrect. A 3.3 percent improvement from 78.7 (the accuracy of the best human-designed prompt) would correspond to an accuracy of 81.3, not 82.0 (the accuracy obtained with the LLM-designed prompt).

   - 3.3 times

     Incorrect. The improvement was not a multiple of 3.3 times but an increase of 3.3 points.

Lecture 3.6

1. Assistant models are trained in several stages. Which of the following stages does not involve the language modelling objective?

   - reward modelling

     Correct. Reward modelling focuses on learning to predict human preferences and does not use the language modelling objective, which involves next-token prediction.

   - unsupervised pre-training

     Incorrect. Unsupervised pre-training directly uses the language modelling objective to predict the next token in a sequence.

   - instruction fine-tuning

     Incorrect. Instruction fine-tuning still uses the language modelling objective but incorporates task-specific instructions to guide responses.

2. Consider a customer support chatbot with a female avatar. Which of the following outputs would be an example showing the limitation of language modelling as an objective in instruction fine-tuning?

   - I'm a woman so I just don't understand.

     Correct. This output is okay from a language modelling perspective but demonstrates a failure in aligning the model to avoid generating biased or inappropriate statements.

   - Not I understand sorry sorry.

     Incorrect. While grammatically incorrect, this example reflects a language modeling failure, not a limitation in the objective.

   - Can you repeat that? I am not sure I understand.

     Incorrect. This is an appropriate and coherent response, showing no clear limitation of the language modeling objective.

3. What is the purpose of reward models in LLM training?

   - to act as a proxy for immediate human feedback

     Correct. Reward models are trained to predict human preferences and serve as a proxy for human feedback during reinforcement learning.

   - to suggest suitable payment for human annotators

     Incorrect. Reward models are unrelated to annotator compensation.

   - to increase the computational efficiency of the training process

     Incorrect. Reward models are used to align the model's behaviour with human preferences, not to improve computational efficiency.

4. For a well-aligned LLM, which of the following is the most plausible reward model loss for the two completions from the moon landing example?

   - 0.11

     Correct. A low loss (e.g., 0.11) indicates that the reward model successfully distinguishes between the preferred and non-preferred completions.

   - 0.69

     Incorrect. A loss around 0.69 suggests random guessing, indicating the reward model is not distinguishing between the outputs effectively.

   - 2.30

     Incorrect. A higher loss like 2.30 implies the model is failing to align with human preferences.

5. What is the goal of policy gradient methods?

   - maximise the probability of generating outputs with high reward

     Correct. Policy gradient methods aim to optimise the model's parameters to increase the likelihood of generating outputs that receive higher rewards.

   - maximise the reward of generated outputs

     Incorrect. While related, policy gradient focuses on increasing the probability of high-reward outputs rather than the absolute reward.

   - minimise the perplexity of the generated outputs

     Incorrect. Minimising perplexity is associated with language modelling, not reinforcement learning using policy gradients.

Unit 4

Lecture 4.1

1. What is the main challenge with sequence labelling tasks?

   - The search space grows exponentially with the length of the input sequence.

     Correct. In sequence labelling tasks, the number of possible label sequences grows exponentially with the length of the input, making the task computationally challenging.

   - The number of atomic labels per sequence element is extremely large.

     Incorrect. While some tasks may have many labels, the primary challenge lies in the complexity of the sequence itself.

   - The amount of training data for sequence labelling is generally very limited.

     Incorrect. Although limited data can be a challenge, it is not the main issue; the core challenge is the size of the search space.

2. In the part-of-speech tagging example, what is the tag for the word *live*?

   - verb

     Correct. In the sentence "I want to live in peace", the word *live* functions as a verb.

   - adjective

     Incorrect. The word *live* can be an adjective in contexts like "live broadcast", but that is not the case here.

   - noun

     Incorrect. The word *live* cannot function as a noun.

3. Which of the following types of tasks is *not* a type of sequence labelling?

   - translation

     Correct. Translation is a sequence-to-sequence task, not a sequence labelling task, as it involves generating an entirely new sequence.

   - segmentation

     Incorrect. Segmentation involves labelling sequence boundaries and is a form of sequence labelling.

- bracketing

  Incorrect. Bracketing, used in syntactic parsing, involves labelling structures within a sequence and is considered sequence labelling.

4. In the BIO tagging scheme, what information does the O-tag represent?

   - the tagged word does not belong to an entity

     Correct. In the BIO (Begin, Inside, Outside) tagging scheme, the O-tag indicates that a word is outside of any named entity.

   - the tagged word belongs to an organisation

     Incorrect. This would be tagged as B-ORG or I-ORG in the BIO scheme.

   - the tagged word is an ordinal number

     Incorrect. The O-tag does not relate to specific types of entities but rather indicates non-entity tokens.

5. The following matrix represents the confusion matrix for a part-of-speech tagger:

$$\begin{bmatrix} 58 & 6 & 1 \\ 5 & 11 & 2 \\ 0 & 7 & 43 \end{bmatrix}$$

   The entry in row A, column B gives the number of times the tagger predicted tag B whereas the gold-standard tag was A. What is the accuracy of the tagger?

   - 16%

     Incorrect.

   - 58%

     Incorrect.

   - 84%

     Correct. The tagger's accuracy is calculated as:

$$\frac{58 + 11 + 43}{\sum \text{all entries}} = \frac{112}{133} \approx 84.2\%$$

Lecture 4.2

1. Which of the following features can be used in a model for sequence labelling, but not in a model for language modelling?

   - the word to the left of the current word

     Incorrect. Language models (especially bidirectional models like BERT) can use the word to the left of the current word.

   - the current word

     Incorrect. Both sequence labelling and language modelling use the current word as input.

   - the word to the right of the current word

     Correct. In sequence labelling tasks, the model can use the entire input sequence, including future context (the word to the right), whereas traditional autoregressive language models can only use past context.

2. We train an autoregressive part-of-speech tagger. Which additional input do we give to the model in the teacher forcing regime?

   - the previous word's gold-standard label

     Correct. In teacher forcing, the model is provided with the gold-standard label for the previous word to predict the current word's tag, improving convergence during training.

   - the next word's gold-standard label

     Incorrect. Teacher forcing uses past gold-standard labels, not future ones.

   - the label which the model predicted for the previous word

     Incorrect. This approach is used during inference but not in teacher forcing, where gold-standard labels are used.

3. What problem can the exposure bias lead to?

   - error propagation

     Correct. Exposure bias arises when a model is trained with gold-standard sequences (e.g., in teacher forcing) but, during inference, must rely on its own predictions. Early mistakes can propagate, leading to compounding errors.

   - excessive amounts of padding

     Incorrect. Padding is related to handling sequences of varying lengths, not exposure bias.

- vanishing gradients

  Incorrect. Vanishing gradients are a training issue in deep networks but are unrelated to exposure bias.

4. What does it mean for a scoring function to be factorised?

   - Its value on a given sequence can be computed from the values of simple parts.

     Correct. A factorised scoring function allows the score of a complete sequence to be decomposed into scores of smaller, simpler parts, which facilitates efficient computation.

   - Its value on a given sequence can be written as a product.

     Incorrect. While some factorised functions can be represented as a product, factorisation generally refers to decomposability into simpler components, not strictly products.

   - The code for the computation of the scoring function can be refactored.

     Incorrect. Factorisation refers to mathematical decomposition, not code structure.

5. What is the decoding problem for Maximum Entropy Markov Models (MEMMs)?

   - Find the highest-probability label sequence, given the input sequence

     Correct. In MEMMs, decoding involves finding the sequence of labels that maximises the overall probability given the input sequence, often using algorithms like Viterbi.

   - Minimise the negative log likelihood of the training data

     Incorrect. This describes the objective during training, not decoding.

   - Find the label sequence with the highest negative log probability, given the input sequence

     Incorrect. The goal is to maximise probability, not the negative log probability.

Lecture 4.3

1. As its main data structure, the Viterbi algorithm uses a matrix with *m* rows and *n* columns. What does the value *n* represent?

   - the length of the input sequence

     Correct. In the Viterbi algorithm, *n* represents the number of time steps or the length of the input sequence, where each column corresponds to one input token.

   - the number of labels (or tags)

     Incorrect. The number of labels or tags corresponds to the *m* rows, not the columns.

   - the number of words in the vocabulary

     Incorrect. The size of the vocabulary is not directly represented in the Viterbi matrix structure.

2. In the central matrix for the run of the Viterbi algorithm on the example sentence, what is the intended value in row VERB, column *see*?

   - the maximal score when tagging the first 4 words in the sentence and where *see* is tagged as a VERB

     Correct. The Viterbi algorithm computes the highest scoring path to each cell, so this value represents the highest score for tagging the first four words with *see* as a VERB.

   - the maximal score when tagging the first 4 words in the sentence as DET NOUN AUX VERB

     Incorrect. The algorithm does not commit to a specific sequence; it computes the best path to each state independently.

   - the maximal score when tagging the word *see* as a VERB

     Incorrect. The value incorporates the full sequence up to *see*, not just the individual word's tag.

3. Which of the following numbers are used in the calculation of the value for row NOUN, column *can(6)*?

   - −25.31, −0.53, −9.54

     Correct. The Viterbi algorithm combines the previous state's best score (−25.31), the transition probability (−0.53), and the emission probability for *can* being tagged as a NOUN (−9.54) to compute the new value.

- $-33.72, -0.53, -1.13$

  Incorrect.

- $-25.31, -4.93, -9.54$

  Incorrect.

4. The most probable tag sequence for the input sentence in the example is DET NOUN AUX VERB DET NOUN. Which tag sequence would we have obtained, had we picked the tags by the maximal value in each column?

   - DET NOUN AUX VERB DET AUX

     Correct.

   - DET NOUN AUX VERB DET NOUN

     Incorrect.

   - DET NOUN VERB VERB DET AUX

     Correct.

5. When one is only interested in the *score* of the least expensive tag sequence, not in the sequence itself, then the memory required by the Viterbi algorithm is in $O(m)$. Why?

   - We can forget all but the previous column when computing the next column.

     Correct. Since each column only depends on the previous one, we can discard older columns, reducing memory usage to $O(m)$, where m is the number of tags.

   - The chain of backpointers cannot become longer than m elements.

     Incorrect. The backpointer chain can grow with the length of the input sequence, not just the number of tags.

   - We only need to store the first m rows of the matrix.

     Incorrect. All rows (representing tags) are needed at each step, but only for the current and previous columns.

Lecture 4.4

1. Why are dependency trees interesting for natural language processing?
   - They contain information about the predicate–argument structure of a sentence.

     Correct. Dependency trees capture syntactic relations between words, including predicate–argument structures, which are crucial for understanding sentence meaning.
   - They contain information about the semantic similarity of the words in a sentence.

     Incorrect. Dependency trees focus on syntactic relationships, not semantic similarity.
   - They contain information about the part-of-speech tags of the words in a sentence.

     Incorrect. While dependency trees rely on part-of-speech tags, they primarily represent syntactic dependencies between words.
2. Which of the following statements about dependency trees is *not* generally true?
   - Every node has at most one dependent.

     Correct. This is false. In dependency trees, nodes (words) can have multiple dependents representing various syntactic relationships.
   - Dependency trees do not contain cycles.

     Incorrect. This is true; dependency trees are acyclic by definition.
   - Every node is reachable from the tree's root node.

     Incorrect. This is true; all nodes in a dependency tree are connected through the root.
3. We are interested in the number of potential dependency trees for a sentence consisting of 20 words. How many digits are in that number?
   - 15

     Incorrect.
   - 25

     Correct. The number of dependency trees with n nodes is given by the formula n^{n-1}. For 20 words, 20^{19} has 25 digits.
   - 35

     Incorrect.

4. We implement a dependency parser based on the Chiu–Liu/Edmonds algorithm. By which factor would we expect the parser runtime to grow when we double the sentence length?

   - 2

     Incorrect.

   - 3

     Incorrect.

   - 8

     Correct. The Chiu–Liu/Edmonds algorithm has a time complexity of $O(n^3)$. Doubling the input size increases runtime by $2^3 = 8$ times.

5. In the context of dependency parsing, what is an oracle?

   - A function that maps a dependency tree to a sequence of transitions.

     Correct. In transition-based dependency parsing, an oracle guides the parser by mapping the gold-standard tree to the sequence of parsing transitions needed to reconstruct it.

   - A function that maps a sentence to its gold-standard dependency tree.

     Incorrect. This describes the treebank or an ideal parser, not the oracle.

   - A function that maps a sequence of transitions to a dependency tree.

     Incorrect. This is the role of the parser itself, not the oracle.

Lecture 4.5

1. Here are three dependency trees on three-word sentences. Each tree is specified by the list of the head positions, as introduced in Lecture 4.4.

   A: 2 0 2   B: 2 0 1   C: 0 3 1

   Which of these trees is not projective?

   - A

     Incorrect.

   - B

     Correct. In Tree B, word 1 has head 2, and word 3 has head 1, leading to the "gap" 2 in the subtree rooted at word 1.

   - C

     Incorrect.

2. Which of the following does *not* hold for terminal configurations in the arc-standard algorithm?

   - the stack is empty

     Correct. In the terminal configuration of the arc-standard algorithm, the stack contains exactly the root node; it is not empty.

   - the buffer is empty

     Incorrect. In the terminal state, the buffer is empty, indicating that all words have been processed.

   - the dependency tree is projective

     Incorrect. The arc-standard algorithm only builds projective dependency trees, so this holds true.

3. The following transition sequence creates a dependency tree for a six-word sentence:

   SH SH SH SH RA SH SH LA RA LA RA

   In this tree, which of the following words is not a dependent of word 3?

   - word 2

     Incorrect.

   - word 4

     Incorrect.

- word 5

  Correct. Word 5 is attached to word 6.

4. Which of the following statements about the arc-standard algorithm is true?
   - Every valid transition sequence builds some dependency tree.

     Correct. Any valid sequence of transitions in the arc-standard algorithm results in a well-formed dependency tree.
   - Every dependency tree can be built by some valid transition sequence.

     Incorrect. The arc-standard algorithm can only build projective trees, not every dependency tree.
   - Every valid transition sequence builds the correct dependency tree.

     Incorrect. A valid sequence may lead to a well-formed but incorrect dependency tree relative to the gold-standard tree.

5. For the arc-standard algorithm, how many *additional* transitions will we need to take if we double the sentence length from 10 to 20?
   - 19

     Incorrect.
   - 20

     Correct. The arc-standard algorithm requires $2n - 1$ transitions for a sentence of length n. Doubling from 10 to 20 words increases transitions from $2 \times 10 - 1 = 19$ to $2 \times 20 - 1 = 39$, resulting in exactly 20 additional transitions.
   - 39

     Incorrect. 39 is the total number of transitions for the 20-word sentence, not the number of additional transitions.

Lecture 4.6

1. What was the main innovation in the architecture of Chen and Manning (2014)?

   - Instead of crafting feature combinations by hand, learn them using a neural network.

     Correct. Chen and Manning (2014) introduced a neural network-based dependency parser that automatically learns feature combinations, eliminating the need for manual feature engineering.

   - Instead of using only atomic features, also use feature combinations.

     Incorrect. The key innovation was automating feature combination learning, not simply adding feature combinations.

   - Replacing the ReLU activation function in their FNN with the cube function.

     Incorrect. While Chenning and Manning (2014) used a cube function, this was not the main innovation. (Also, the cube function was not used by later models.)

2. Consider the second configuration from the Chen and Manning (2014) example, and suppose that in this configuration, the parser makes the transitions SH RA. In the new configuration, what is the value of the stack 1 feature?

   - wanted

     Incorrect.

   - try

     Correct. The configuration the question refers to has *wanted* and *try* on top of the stack. Making the transitions SH RA adds the *someplace* to the stack, creates an arc from *try* to *someplace* and then pops *someplace*. After this, the new top-of-the-stack is again *try*.

   - someplace

     Incorrect.

3. What was the main innovation in the architecture of Kiperwasser and Goldberg (2016)?

   - Reduce the number of core features, but make each feature more expressive.

     Correct. Kiperwasser and Goldberg (2016) introduced BiLSTMs to replace handcrafted feature extraction, allowing for richer, context-aware features while reducing the need for manual engineering.

- Reduce the number of feature combinations, increasing parser efficiency.

  Incorrect. The focus was on enhancing feature representation through BiLSTMs, not merely reducing feature combinations.

- Replace the activation function in the Chen and Manning (2014) network with a ReLU.

  Incorrect. The innovation was in using BiLSTMs, not changing activation functions.

4. How are arc scores computed in the application of the Kiperwasser and Goldberg (2016) framework to graph-based parsing?

   - feed the concatenation of the head and the dependent embeddings through a feed-forward NN

     Correct. In graph-based parsing, Kiperwasser and Goldberg used a BiLSTM to embed words and computed arc scores by feeding the concatenated head and dependent embeddings into a feed-forward network.

   - feed the concatenation of the embeddings for the top two words on the stack through a feed-forward NN

     Incorrect. This approach aligns more with transition-based parsing rather than graph-based parsing.

   - embed each word by unrolling a Bi-LSTM over the words of the sentence

     Incorrect. While a BiLSTM is used to embed words, the arc scoring step specifically involves concatenating head and dependent embeddings.

5. The architecture of Glavaš and Vulić (2021) uses a biaffine layer with weight matrix W. Suppose that words are embedded into 100-dimensional vectors w_i, w_j. How many entries does W have?

   - 100

     Incorrect. This reflects only the dimensionality of the input vectors, not the size of the weight matrix.

   - 200

     Incorrect. This is the combined size of the concatenated input vectors, but not the size of W.

   - 10,000

     Correct. In a biaffine transformation, the weight matrix W has dimensions corresponding to the product of the input vector sizes. With two 100-dimensional embeddings, W has $100 \times 100 = 10,000$ entries.