

# 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  $\rightarrow$  1 byte (7-bit ASCII), ó  $\rightarrow$  2 bytes (cf. the example from Icelandic), □  $\rightarrow$  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?

- $a + a \rightarrow [aa]$

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

- $[aa] + a \rightarrow [aaa]$

Incorrect. The second merge rule is not applied to a 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 90 degrees.

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



- 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.

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  $U$  have?

- 10,000

Incorrect.

- 1,500,000

Correct. The matrix  $U$  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  $U$ .

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

- The matrix  $U$  has 100 columns.

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

- The matrix  $V$  has 100 rows.

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

- The matrix  $\Sigma$  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  $P(c | w)$  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  $P(c | w)$  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  $P(c | w)$  using a simpler prediction task

Correct. Negative sampling approximates  $P(c | w)$  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  $P(c | w)$ , 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 to the maximum size, the model samples these sizes at random to speed up the training.