Your grade: 100%

Your latest: 100% • Your highest: 100% • To pass you need at least 66%. We keep your highest score.

Next item

1. In word2vec models, the number of neurons in input and output layers corresponds to which of the following? 1 / 1 point

- Context words
- Numerical representations
- (selected) Vocabulary size
- Word vector dimensions

Correct
The number of neurons in both input and output layers corresponds to the vocabulary size.

2. While creating the skip-gram model in PyTorch, what does the following code do? 1 / 1 point

```
self.embeddings = nn.Embedding(num_embeddings=vocab_size, embedding_dim=embed_dim)
```

- (selected) It defines the embeddings layer.
- It passes the input text through the embedding layer.
- It performs the forward pass.
- It defines the fully connected layer.

Correct
When creating a skip-gram in PyTorch, the given code defines the embeddings layer using 'nn.Embedding,' which creates word embeddings for the given vocabulary size and embedding dimension.

3. Which of the following statements is correct? 1 / 1 point

- Sequence-to-label tasks generate a part of a sequence from a descriptive input, as seen in generative models for image creation.
- Neural networks can only operate under the assumption that each sample is dependent on others and distinctly distributed.
- (selected) Sequence-to-sequence models help with code generation, where you can describe your task, and the AI generates the appropriate code.
- Label-to-label tasks take a single input to produce multiple labels, which is useful in document classification.

Correct
The statement is correct. Sequence-to-sequence models within generative AI are used in machine translation, such as converting English phrases into French. Chatbots use sequence-to-sequence models to transform your queries into conversational responses, while summarization algorithms condense extensive texts into concise summaries. With code generation, you describe your task, and the AI generates the appropriate code.

4. Consider the following code, which is used to reshape the output tensor while training a sequence-to-sequence model. 1 / 1 point

```
output_dim = output.shape[-1]

output = output[1:].view(-1, output_dim)

trg = trg[1:].contiguous().view(-1)
```

For sequence models like RNNs, where the input shape often differs from other model types, why is it crucial to reshape the output tensor as well?

- To calculate the average loss per batch
- To initialize the model in training mode to activate essential layers
- (selected) To align the rows and columns correctly for loss calculation
- To generate predictions by output

Correct
In the given code, target length '-1' represents the rows, excluding the initial <bos> or (beginning-of-sequence) token, ensuring you don't include

it in loss computation. 'Batch size' is the columns, each representing a separate sequence in the batch, and 'output dimension' corresponds to the columns representing the predicted output for each token in the sequence. This reshaping aligns the rows and columns correctly for loss calculation, matching the output predictions to the target dimensions.

5. "For the encoder, the interest lies only in the hidden state." What is the reason for this? 1 / 1 point

- Because the encoder generates the output text.
- Because the encoder autoregressively generates the translation as one token at a time.
- (selected) Because the encoder is only responsible for encoding the input sequence.
- Because the encoder seamlessly integrates with the PyTorch training pipeline.

Correct
For the encoder, the interest lies only in the hidden state and to not use the output. This is because only the decoder will generate the output text. The encoder is only responsible for encoding the input sequence.

6. Consider the following reference and hypothesis: 1 / 1 point

Reference: The dog runs on the ground

Hypothesis: The big dog runs in the park

Find the count of matching n-grams by comparing a hypothesis sequence with a reference sequence.

- (selected) Unigrams: 4; Bigrams: 1
- Unigrams: 4; Bigrams: 2
- Unigrams: 5; Bigrams: 2
- Unigrams: 3; Bigrams: 1

Correct
Comparing the hypothesis sequence with the reference sequence.

"The" is matched so the unigram count increases to 1.

"big" is not matched so the unigram count remains 1.

"dog" is matched so the unigram count increments to 2.

"runs" is matched so the unigram count increments to 3.

Also, "dog runs" is the first matched bigram, so the bigram count increases to 1.

"in" is not matched so the unigram count stays at 3.

"the" is matched so the unigram count goes to 4.

"park" is not matched, so the unigram count remains 4 and the bigram count remains 1.