# Your grade: 100%

Your latest: **100%** • Your highest: **100%** • To pass you need at least 80%. We keep your highest score.

Next item

**1.** Which of the following options explains the RAG process in the correct sequence? 1 / 1 point

- ○ The retriever retrieves the prompts, encodes them into vectors, and stores them for further use.
- ○ The retriever decodes user-provided prompts into vectors, stores them, and retrieves them to generate a response.
- ◉ The retriever encodes prompts into vectors, stores them, and then retrieves the context vectors.
- ○ The retriever stores the prompts, retrieves them when needed, and then encodes them into vectors.

**Correct**
The retriever encodes user-provided prompts into vectors, stores them in a vector database, and retrieves relevant context vectors based on the distance between the encoded prompt and documents. The generator then combines the retrieved context with the original prompt to produce a response.

**2.** How does RAG help convert contextual data from the knowledge base into vectors using the correct sequential steps? 1 / 1 point

- ○ RAG only encodes the text chunks for the vector representation by transforming them into high-dimensional vectors to generate responses.
- ○ Only distance operations on the embeddings help chunk IDs identify relevant information to generate responses.
- ◉ In the context of RAG, the contextual data gets converted into the knowledge base vectors by breaking down the text into chunks, embedding each chunk into vectors, followed by indexing, encoding, averaging the information, generating chunk IDs, and generating relevant context for the inserted prompt.
- ○ The chunk IDs help in finding the vectors for generating responses.

**Correct**
During context encoding, the data gets split into smaller, manageable texts and inserted into the vectors to generate accurate and relevant responses from the knowledge base for the inserted prompt.

**3.** In the RAG process, which of the following is the primary step? 1 / 1 point

- ○ Generate the model
- ◉ Retrieve information
- ○ Create an augmented query
- ○ Embed the texts

**Correct**
Retrieval-augmented generation (RAG) process is the primary step for the RAG process. It helps in retrieving relevant information from the knowledge base or database. This information is then used to augment the input, enabling the model to generate precise output.

**4.** Identify the correct statement for context tokenization in natural language processing (NLP). 1 / 1 point

- ○ To translate text from one language to another.
- ◉ Context tokenization converts the text into a numerical format for machine learning to process.
- ○ To identify the main topic of a document.
- ○ To generate a summary of a given text.

**Correct**
Context tokenization splits the text into tokens, which are then converted into a numerical format.

**5.** Which of the following code snippets indicate that the context tokenizer will process the input text by tokenizing, padding, and truncating it to a maximum length of 256 tokens and then converting the text into a dictionary of PyTorch tensors? 1 / 1 point

```
from transformers import DPRContextEncoderTokenizer

model_name = 'facebook/dpr-ctx_encoder-single-nq-base'
context_tokenizer = DPRContextEncoderTokenizer.from_pretrained(model_name)
```

```
from transformers import DPRContextEncoder

encoder_model = 'facebook/dpr-ctx_encoder-single-nq-base'
context_encoder = DPRContextEncoder.from_pretrained(encoder_model)
```

```
text = [("How are you?", "I am fine."), ("What's up?", "Not much.")]

tokens_info=context_tokenizer(text, return_tensors='pt', padding=True, \
truncation=True, max_length=256)

tokens_info:
{'input_ids': tensor([[ 101, 2129, 2024, 2017, 1029,  102, 1045, 2572,
2986, 1012,  102],
[ 101, 2054, 1005, 1055, 2039, 1029,  102, 2025, 2172, 1012,  102]]),
'token_type_ids': tensor([[0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1],
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1]]),
'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])}
```

```
outputs=context_encoder(**tokens_info)

outputs.pooler_output.shape:
torch.Size([2, 768])
```

**Correct**

In the context tokenizer, a list of tuples contains a pair of sentences, and the output in the context tokenizes the information. This helps the input text to tokenize, pad, and truncate it to a maximum length of 256 tokens.