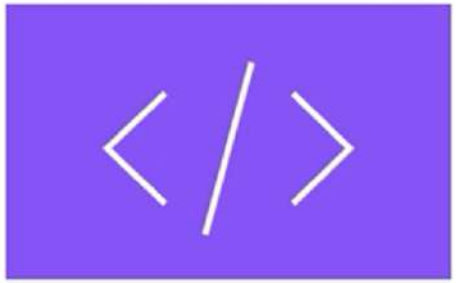
Skills Network

RAG, Encoders, and Faiss

IBM

RAG, Encoders, and Faiss

What you will learn



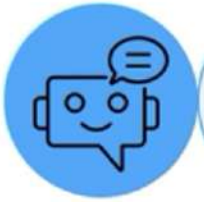
Describe context encoders and how they work

Explain Facebook AI Similarity Search (Faiss)

Explain question encoders

Discuss how to generate answers

Introduction

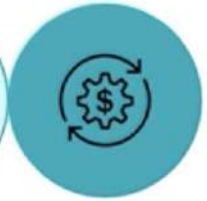


AI engineer to create chatbot for policies

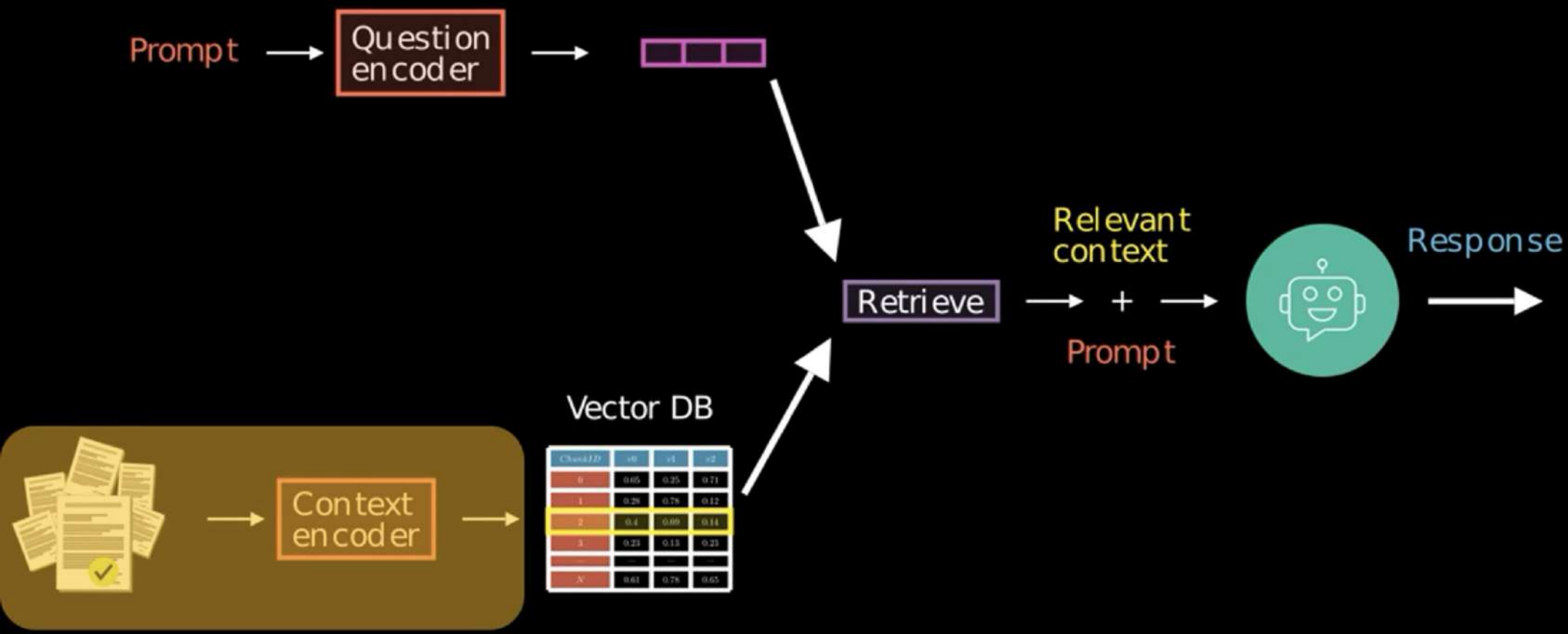
Integrate RAG

Generate updated responses

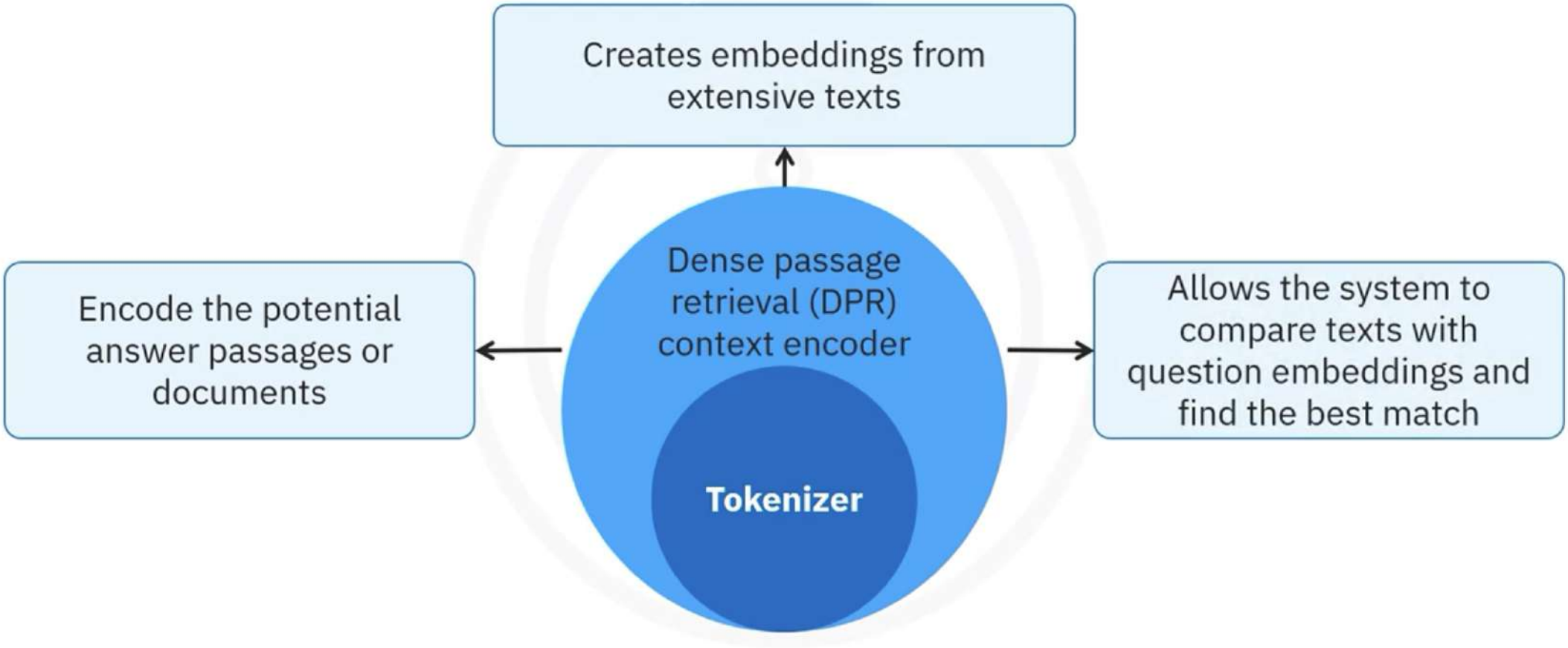
Increase accuracy and reduce overheads



RAG process


Prompt
Question encoder
Retrieve
Relevant context
+
Prompt
Response
Vector DB
Context encoder

Context encoder



Context tokenizer

```
from transformers import DPRContextEncoderTokenizer
```

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

Context tokenizer

```
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]])}
```

Context Encoder

```
from transformers import DPRContextEncoder
```

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

Context Encoder

```
outputs=context_encoder(**tokens_info)
```

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

Loading the dataset

```
def read_and_split_text(filename):
 with open(filename, 'r', encoding='utf-8') as file:
  text = file.read()
 paragraphs = text.split('\n')
 paragraphs = [para.strip() for para in paragraphs if len(para.strip()) > 0]
 return paragraphs
```

```
paragraphs = read_and_split_text('companyPolicies.txt')
```

```
sample: 1 paragraph: Our Code of Conduct outlines the fundamental principles and
ethical standards that guide every member of our organization. We are committed to
maintaining a workplace that is built on integrity, respect, and accountability.

sample: 2 paragraph: Integrity: We hold ourselves to the highest ethical standards.
This means acting honestly and transparently in all our interactions, whether with
colleagues, clients, or the broader community. We respect and protect sensitive
information, and we avoid conflicts of interest.
```

Putting it together

```
def encode_contexts(text_list):
 embeddings = []
 for text in text_list:
  inputs = context_tokenizer(text, return_tensors='pt', padding=True, \
   truncation=True, max_length=256)
  outputs = context_encoder(**inputs)
  embeddings.append(outputs.pooler_output)
 return torch.cat(embeddings).detach().numpy()
```

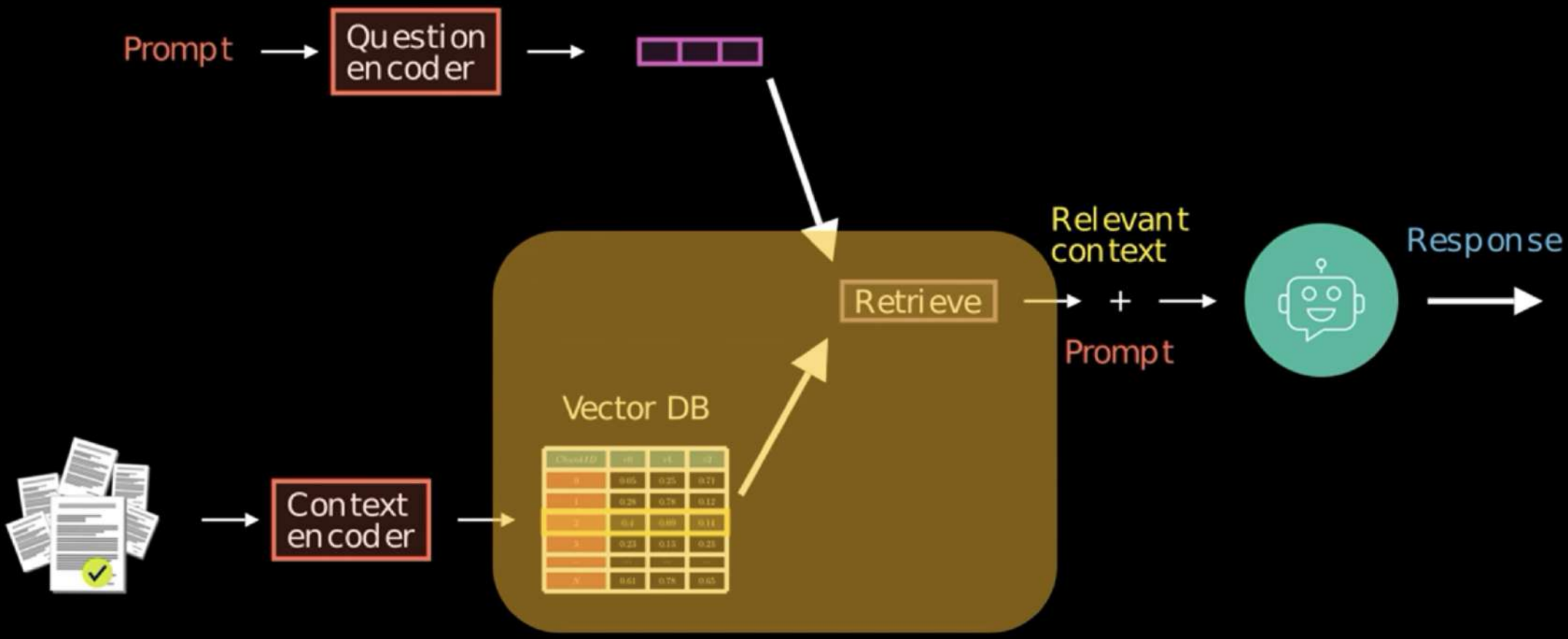
```
context_embeddings = encode_contexts(paragraphs)
```

```
context_embeddings.shape:
(76, 768)
```

Facebook AI Similarity Search (Faiss)

- Library developed by Facebook AI Research
- Offers efficient algorithms for searching through large collections of high-dimensional vectors

RAG process


Prompt
Question encoder
Relevant context
Response
Retrieve
+
Prompt
Vector DB
Context encoder

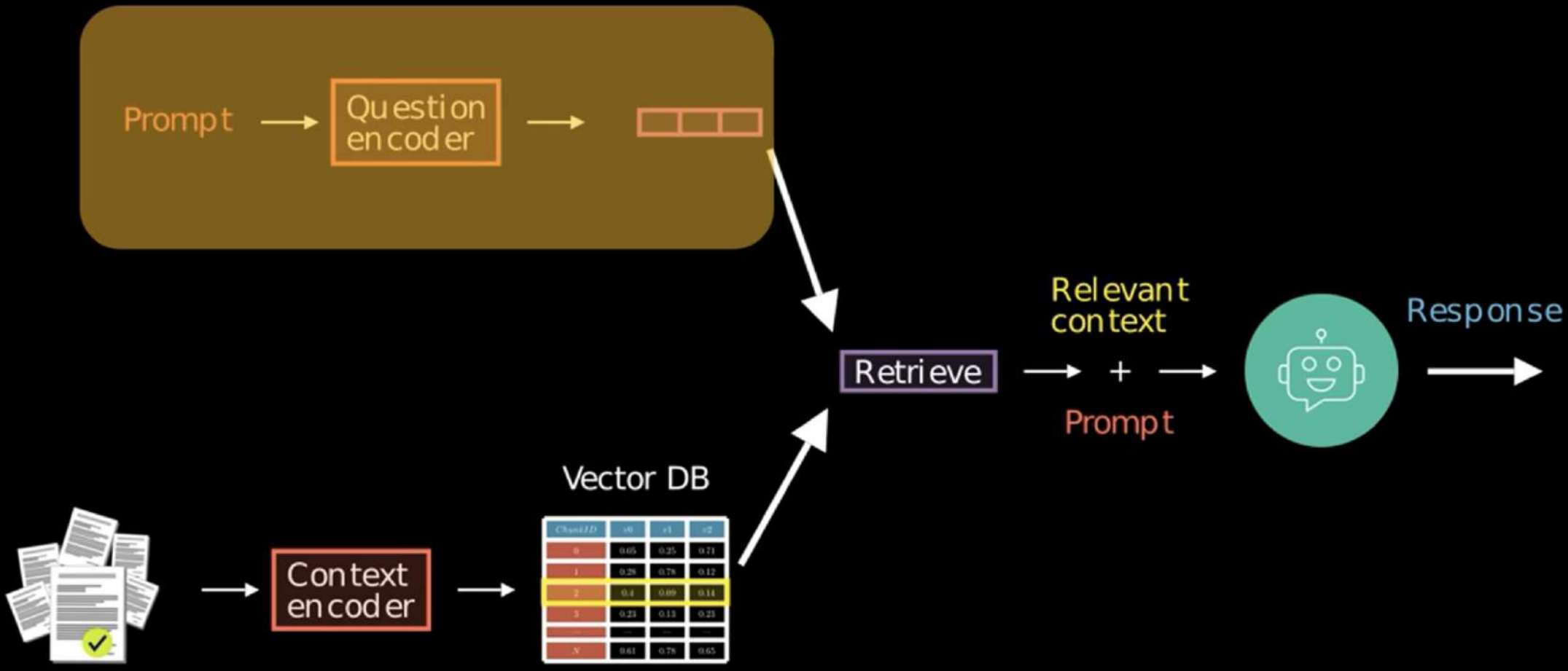
Faiss

```
import faiss
```

```
embedding_dim = 768
context_embeddings_np = np.array(context_embeddings).astype('float32')
```

```
index = faiss.IndexFlatL2(embedding_dim)
index.add(context_embeddings_np)
```

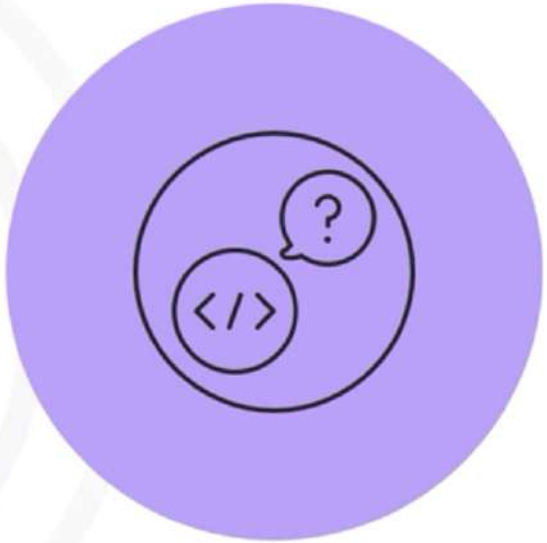
RAG process


Prompt
Question encoder
Retrieve
Relevant context
+
Prompt
Response
Vector DB
Context encoder

Question encoder

In contrast:

- DPR question encoder and its tokenizer focus on encoding input questions into fixed-dimensional vector representations
- Grasp the meaning and context of the questions
- Facilitate answering questions



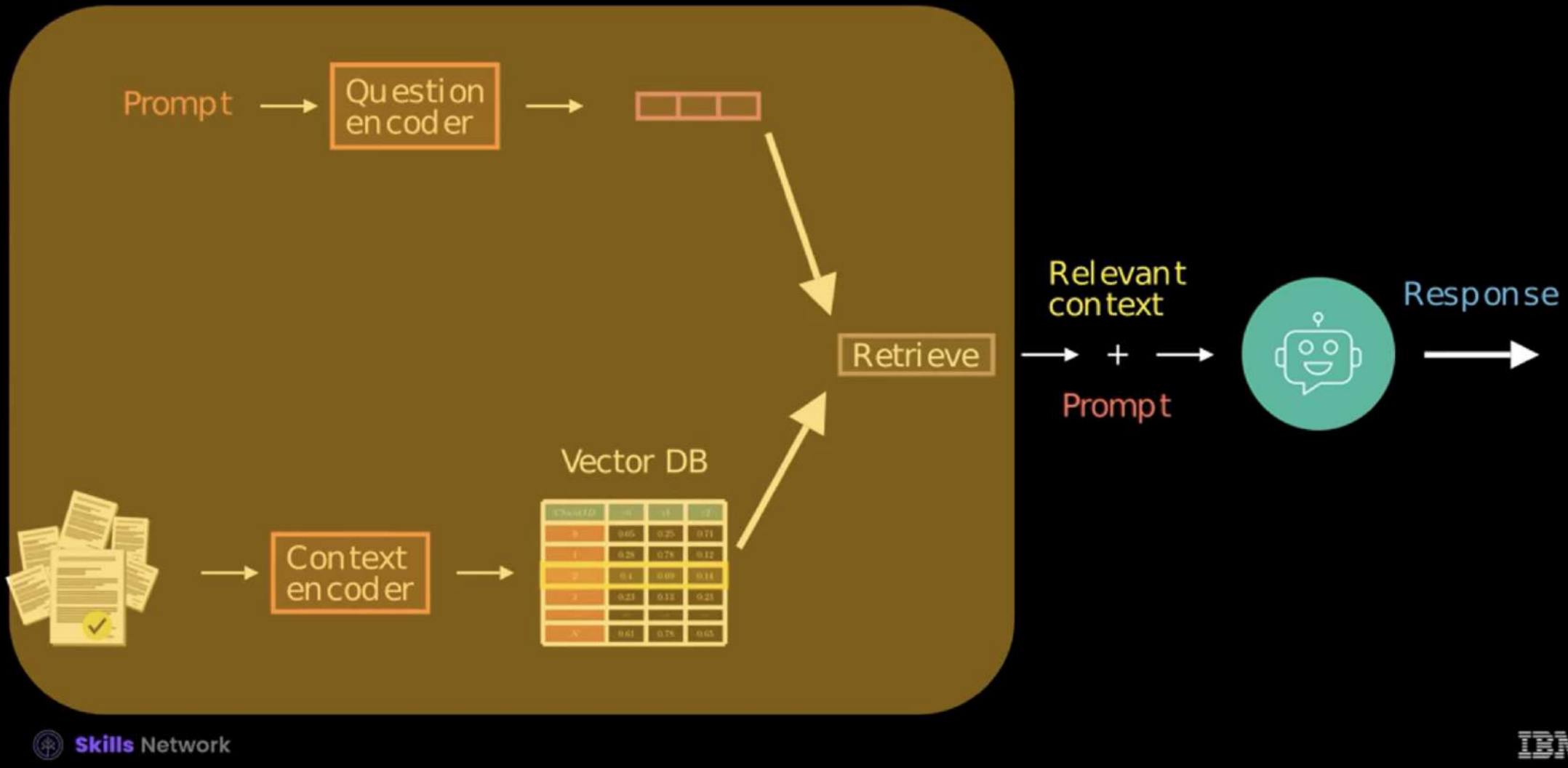
DPR question encoder and tokenizer

```
from transformers import DPRQuestionEncoder, DPRQuestionEncoderTokenizer
```

```
tokenizer_model = 'facebook/dpr-question_encoder-single-nq-base'
question_tokenizer = DPRQuestionEncoderTokenizer.from_pretrained(tokenizer_model)
```

```
encoder_model = 'facebook/dpr-question_encoder-single-nq-base'
question_encoder = DPRQuestionEncoder.from_pretrained(encoder_model)
```

RAG process


Prompt
Question encoder
Retrieve
Relevant context
+
Prompt
Response
Vector DB
Context encoder

Example query and context retrieval

```
question = 'Drug and Alcohol Policy'
question_inputs = question_tokenizer(question, return_tensors='pt')
question_embedding = question_encoder(**question_inputs).pooler_output.detach().numpy()
```

```
D, I = index.search(question_embedding, k=3)
```

```
print("D:",D)
print("I:",I)
```

```
D: [[72.765366 74.716156 84.38809 ]]
I: [[48 49 53]]
```

Example query and context retrieval

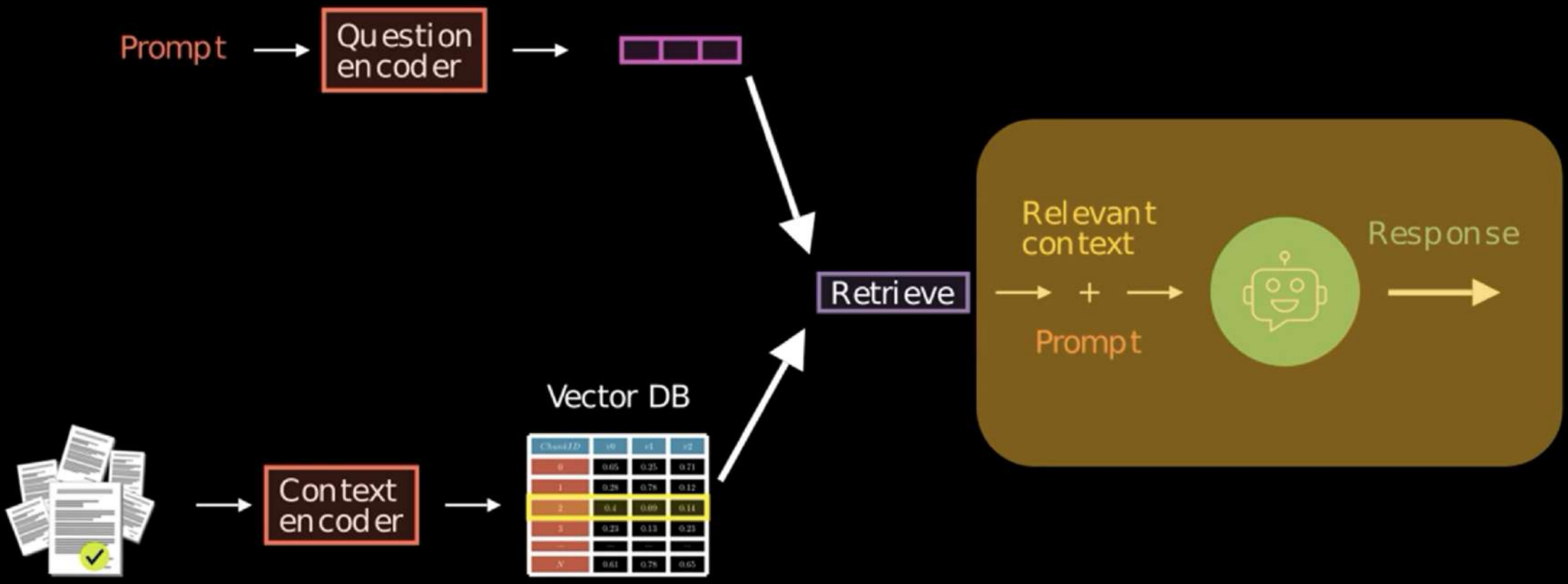
```
for i, idx in enumerate(I[0]):
 print(f"{i+1}: {paragraphs[idx]}")
 print(f"distance {D[0][i]}"
```

```
Top 3 relevant contexts:
1: 6.Drug and Alcohol Policy
distance 72.765365600585 94

2: Policy Objective: The Drug and Alcohol Policy is established to establish clear
expectations and guidelines for the responsible use of drugs and alcohol within the
organization. This policy aims to maintain a safe, healthy, and productive workplace.
distance 74.716156005859 38

3: Testing and Searches: The organization reserves the right to conduct drug and
alcohol testing as per applicable laws and regulations. Employees may be subject to
testing in cases of reasonable suspicion, post-accident, or as part of routine
workplace safety measures.
distance 84.388092041015 62
```

RAG process


Prompt
Question encoder
Retrieve
Relevant context
+
Prompt
Response
Vector DB
Context encoder

BART

```
from transformers import BartForConditionalGeneration, BartTokenizer
```

```
model = BartForConditionalGeneration.from_pretrained('facebook/bart-large-cnn')
tokenizer = BartTokenizer.from_pretrained('facebook/bart-large-cnn')
```

```
inputs = tokenizer(question, return_tensors='pt', max_length=1024, truncation=True)
summary_ids = model.generate(inputs['input_ids'], max_length=150, min_length=40, \
 length_penalty=2.0, num_beams=4, early_stopping=True)
answer = tokenizer.decode(summary_ids[0], skip_special_tokens=True)
```

```
answer: What is a  large language model? What is a large language models? How do we
use them? What are some of the most common models we use? What do you use? Share
your ideas in the comments below.
```

Generating answers without context

```
def generate_answer_without_context(question):
 inputs = tokenizer(question, return_tensors='pt', max_length=1024, truncation=True)
 summary_ids = model.generate(inputs['input_ids'], max_length=150, min_length=40, \
   length_penalty=2.0, num_beams=4, early_stopping=True)
 answer = tokenizer.decode(summary_ids[0], skip_special_tokens=True)
 return answer
```

```
question = "what is mobile policy?"
answer = generate_answer_without_context(question)
```

```
Answer: what is mobile policy?. What is the government'\s policy on mobile phones?
What are its plans for the future of mobile phones. What are the plans for mobile
phones in the future?
```

Generating answers with RAG

```
def generate_answer(contexts):
 input_text = ' '.join(contexts)
 inputs = tokenizer(input_text, return_tensors='pt', max_length=1024, truncation=True)
 summary_ids = model.generate(inputs['input_ids'], max_length=150, min_length=40, \
  length_penalty=2.0, num_beams=4, early_stopping=True)
 return tokenizer.decode(summary_ids[0], skip_special_tokens=True)
```

```
question = "what is mobile policy?"
_,I=search_relevant_contexts(question,question_tokenizer,question_encoder,index,k=5)
top_contexts = [paragraphs[idx] for idx in I[0]]
answer = generate_answer(top_contexts)
```

```
Generated Answer: The Mobile Phone Policy is aimed at promoting the responsible and
secure use of mobile devices in line with legal and ethical standards. Every employee
is expected to comprehend and abide by these guidelines. Regular reviews of the policy
ensure its ongoing alignment with evolving technology and security best practices.
```

Recap

- RAG process: Encoding, storing, and retrieving prompts as vectors to produce a response
- DPR Context encoder and its tokenizer encode potential answer passages or documents
- Faiss: Library by Facebook AI Research that uses algorithms to search through high-dimensional vectors
- Question encoder and its tokenizer:
  - Encode input questions into fixed-dimensional vector representations
  - Grasp meaning and context of questions to answer them
- RAG used to generate answers without context