question. I did this by calling the predict method of the model and passing in the vectorized question.

3

Below are the steps thought for this approach:

1.Pre-process the question and paragraphs to create a list of tokens (individual words)
2.Use a natural language processing model (BERT) to generate embeddings for the tokens
3.Compare the embeddings using a similarity measure (cosine similarity) to find the most similar paragraph
4.Use a threshold to determine if the most similar paragraph is "sufficiently" similar to the question
5.Return the paragraph if it is above the threshold
6.Return None if the paragraph is not sufficiently similar to the question

The code will use a natural language processing (NLP) model to generate embeddings for the tokens in the question and paragraphs, and then compare the embeddings using a similarity measure (cosine similarity) to find the most similar paragraph. Then I used a threshold to determine if the most similar paragraph is "sufficiently" similar to the question. If the paragraph is above the threshold, it is returned as the answer; otherwise, None is returned.

We would require additional implementation for preprocessing the text, generating embeddings with an NLP model, and calculating similarity.