FAQ - Attention Mechanism and Transformers

[Previous](https://olympus.mygreatlearning.com/courses/111352/modules/items/5520681?pb_id=17388)

[Next](https://olympus.mygreatlearning.com/courses/111352/modules/items/5520684?pb_id=17388)

#### **1. How does the following function work?**

def cosine\_score(text):

embeddings = model.encode(text)

norm1 = np.linalg.norm(embeddings[0])

norm2 = np.linalg.norm(embeddings[1])

cosine\_similarity\_score = ((np.dot(embeddings[0],embeddings[1]))/(norm1\*norm2))

return cosine\_similarity\_score

To understand how the function works, let's go over the function one line at a time.

def cosine\_score(text):

This line defines a Python function named *cosine\_score* that takes a single parameter *text*, which is a list or array of two sentences.

embeddings = model.encode(text)

Here, the input text is passed to the *encode()* method of the model instance (which is assumed to be a SentenceTransformer model). This method converts the input text into numerical embeddings (vectors) using pre-trained models designed for generating sentence embeddings. The result is stored in the variable *embeddings*.

norm1 = np.linalg.norm(embeddings[0])

norm2 = np.linalg.norm(embeddings[1])

These lines calculate the Euclidean norms (magnitudes) of the vectors in embeddings. *embeddings* is assumed to be a list or array with at least two elements, and the norms of the first and second elements are stored in *norm1* and *norm2*, respectively.

cosine\_similarity\_score = ((np.dot(embeddings[0], embeddings[1])) / (norm1 \* norm2))

This line calculates the cosine similarity score between the two vectors in embeddings. It is computed by taking the dot product of the vectors and dividing it by the product of their Euclidean norms. This normalization accounts for differences in the magnitudes of the vectors.

return cosine\_similarity\_score

Finally, the function returns the computed cosine similarity score.

#### **2. How does the following function work?**

def top\_k\_similar\_sentences(embedding\_matrix,query\_text,k):

query\_embedding = model.encode(query\_text)

score\_vector = np.dot(embedding\_matrix,query\_embedding)

top\_k\_indices = np.argsort(score\_vector)[::-1][:k]

return data.loc[list(top\_k\_indices), 'review']

To understand how the function works, let's go over the function one line at a time.

def top\_k\_similar\_sentences(embedding\_matrix, query\_text, k):

This line defines a function named *top\_k\_similar\_sentences* that takes three parameters:

* *embedding\_matrix*: A matrix containing embeddings of sentences
* *query\_text*: The input query for which we want to find similar sentences
* *k*: The number of top similar sentences to retrieve

query\_embedding = model.encode(query\_text)

In this line, the function uses the *encode()* method of a SentenceTransformer instance (model) to convert the input *query\_text* into a numerical vector (query\_embedding). This vector represents the semantic content of the input text.

score\_vector = np.dot(embedding\_matrix, query\_embedding)

Here, the function calculates the cosine similarity between the *query\_embedding* vector and all other vectors in the *embedding\_matrix*. The dot product operation (*np.dot*) results in a vector (*score\_vector*) where each element represents the cosine similarity score between the query and the corresponding sentence in the dataset.

top\_k\_indices = np.argsort(score\_vector)[::-1][:k]

This line sorts the *score\_vector* in ascending order using *np.argsort* and then reverses the order (*[::-1]*). The result is an array of indices indicating the positions of the most similar sentences in the dataset, arranged in descending order of similarity scores. The first *k* indices are selected, representing the top *k* most similar sentences.

return data.loc[list(top\_k\_indices), 'review']

Finally, the function returns the '*review*' column values from the dataset corresponding to the top *k* indices. This provides the actual text of the top *k* most similar sentences to the input query, based on their cosine similarity scores.