**Approach**

***~Part A~***

This is a problem of Natural Language Processing (NLP).

First we have to convert the texts to numerical vectors i.e Text embedding

For this we are using Universal Sentence Encoder(USE) version 4. It is trained on the whole wikipedia data.

The pre-trained Universal Sentence Encoder(USE) is publicly available in tensorflow hub.

It encodes text into higher dimensional vectors that can be used for our semantic similarity task.

We are not only converting just based on keyword but the context and meaning.

We give this sentence to our model (USE4), it gives us a "dense numeric vector".

We passed sentence pair and got a vector pair.

After converting the sentences into vectors we try find the similarity b/w our sentences using cosine similarity.

After finding cosine similarity we get our Similarity\_Score value ranging from -1 to 1.

So to over come this we add 1 to Similarity\_Score column, by doing this we get our value ranging from 0 to 2

Then we use Normalization to range our Similarity\_Score values b/w 0 to 1.

***~Part B~***

For building an API we are using Streamlit (open source app framework in Python language)

First we taking input from the user in term of text1 and text2 and passing it to our Semantic\_textual\_similarity function

(which is build in and importing from STS.py)

In that function:

First we are performing Text embedding using the same USE4 (Universal Sentence Encoder).

Then after receiving the vector of type tensorflow.python.framework.ops.EagerTensor which we cannot use it to compute the cosine similarity.

So we are converting it into a numpy array.

Then are finding the cosine similarity and appending it to result list.

Then again we are normalizing it in b/w 0 to 1, but this time we are using the expit function, also known as the logistic sigmoid function, is defined as expit(x) = 1/(1+exp(-x)).

Then we are returning the this normalized value and printing as similarity score.