In [2]:
```
# BERT
# Created by: Harsh Bari
# From: SVNIT, Gujarat
# Mtech Data Science - p23ds004 (2023-25)
# Subject: NLP Project
# Last Update:
```

In [5]:
```
import pandas as pd
import numpy as np
from transformers import BertTokenizer, BertModel
import torch
from google.colab import files
```

In [6]:
```
# Load pre-trained BERT model and tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')
model.eval()  # switch model to evaluation mode
```

```
tokenizer_config.json:   0%|          | 0.00/48.0 [00:00<?, ?B/s]

C:\Users\Harsh Bari\AppData\Local\Programs\Python\Python310\lib\site-packa
ges\huggingface_hub\file_download.py:148: UserWarning: `huggingface_hub` c
ache-system uses symlinks by default to efficiently store duplicated files
but your machine does not support them in C:\Users\Harsh Bari\.cache\huggi
ngface\hub\models--bert-base-uncased. Caching files will still work but in
a degraded version that might require more space on your disk. This warnin
g can be disabled by setting the `HF_HUB_DISABLE_SYMLINKS_WARNING` environ
ment variable. For more details, see https://huggingface.co/docs/huggingfa
ce_hub/how-to-cache#limitations. (https://huggingface.co/docs/huggingface_
hub/how-to-cache#limitations.)
To support symlinks on Windows, you either need to activate Developer Mode
or to run Python as an administrator. In order to see activate developer m
ode, see this article: https://docs.microsoft.com/en-us/windows/apps/get-s
tarted/enable-your-device-for-development (https://docs.microsoft.com/en-u
s/windows/apps/get-started/enable-your-device-for-development)
  warnings.warn(message)

vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]

config.json:   0%|          | 0.00/570 [00:00<?, ?B/s]
```

```
---------------------------------------------------------------------------
-
ImportError                               Traceback (most recent call las
t)
Cell In[6], line 3
      1 # Load pre-trained BERT model and tokenizer
      2 tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
----> 3 model = BertModel.from_pretrained('bert-base-uncased')
      4 model.eval()  # switch model to evaluation mode

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\transform
ers\utils\import_utils.py:1450, in DummyObject.__getattribute__(cls, key)
   1448 if key.startswith("_") and key != "_from_config":
   1449     return super().__getattribute__(key)
-> 1450 requires_backends(cls, cls._backends)

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\transform
ers\utils\import_utils.py:1429, in requires_backends(obj, backends)
   1427 # Raise an error for users who might not realize that classes with
out "TF" are torch-only
   1428 if "torch" in backends and "tf" not in backends and not is_torch_a
vailable() and is_tf_available():
-> 1429     raise ImportError(PYTORCH_IMPORT_ERROR_WITH_TF.format(name))
   1431 # Raise the inverse error for PyTorch users trying to load TF clas
ses
   1432 if "tf" in backends and "torch" not in backends and is_torch_avail
able() and not is_tf_available():

ImportError:
BertModel requires the PyTorch library but it was not found in your enviro
nment.
However, we were able to find a TensorFlow installation. TensorFlow classe
s begin
with "TF", but are otherwise identically named to our PyTorch classes. Thi
s
means that the TF equivalent of the class you tried to import would be "TF
BertModel".
If you want to use TensorFlow, please use TF classes instead!

If you really do want to use PyTorch please go to
https://pytorch.org/get-started/locally/ (https://pytorch.org/get-started/
locally/) and follow the instructions that
match your environment.
```

In [ ]:
```
def generate_bert_vectors(text):
    # Tokenize input text
    input_ids = tokenizer.encode(text, add_special_tokens=True, max_length=

    # Convert tokens to PyTorch tensors
    input_ids = torch.tensor(input_ids).unsqueeze(0)  # Batch size 1

    # Generate BERT embeddings
    with torch.no_grad():
        outputs = model(input_ids)
        last_hidden_states = outputs[0]  # Last-layer hidden states

    # Average pooling to get a fixed-size vector
    pooled_output = last_hidden_states.mean(dim=1).squeeze()

    # Truncate or pad to get a fixed-size vector of length 150
    if pooled_output.size(0) > 150:
        pooled_output = pooled_output[:150]
    elif pooled_output.size(0) < 150:
        padded_output = torch.nn.functional.pad(pooled_output, (0, 150 - po
        pooled_output = padded_output

    return pooled_output.numpy()
```

In [ ]:
```
# Read tweets from CSV file
def process_tweets(csv_file):
    tweets_df = pd.read_csv(csv_file)

    # Assuming tweets are in a column named 'tweets'
    tweets = tweets_df['tweets'].tolist()

    bert_vectors = []
    for tweet in tweets:
        bert_vector = generate_bert_vectors(tweet)
        bert_vectors.append(bert_vector)

    # Add the BERT vectors to the DataFrame and save to CSV
    tweets_df['bert_vectors'] = bert_vectors
    tweets_df.to_csv(csv_file, index=False)

    # Download the updated CSV file
    files.download(csv_file)

    return np.array(bert_vectors)
```

In [ ]:
```
# Example usage
csv_file_path = 'dataset.csv'
tweet_vectors = process_tweets(csv_file_path)
print(tweet_vectors.shape)  # Shape should be (num_tweets, 150)
```

In [ ]:
```

```