In [1]:
```
!pip install -U transformers accelerate --quiet
```

In [2]:
```
import pandas as pd

# Load datasets
train_df = pd.read_csv('/content/drive/MyDrive/archive/twitter_training.csv'
val_df = pd.read_csv('/content/drive/MyDrive/archive/twitter_validation.csv'

# Display sample rows
print(train_df.head())
print(val_df.head())
```

```
      0            1         2  \
0  2401  Borderlands  Positive
1  2401  Borderlands  Positive
2  2401  Borderlands  Positive
3  2401  Borderlands  Positive
4  2401  Borderlands  Positive

                                                   3
0  im getting on borderlands and i will murder yo...
1  I am coming to the borders and I will kill you...
2  im getting on borderlands and i will kill you ...
3  im coming on borderlands and i will murder you...
4  im getting on borderlands 2 and i will murder ...
      0          1           2  \
0  3364   Facebook  Irrelevant
1   352     Amazon     Neutral
2  8312  Microsoft    Negative
3  4371      CS-GO    Negative
4  4433     Google     Neutral

                                                   3
0  I mentioned on Facebook that I was struggling ...
1  BBC News - Amazon boss Jeff Bezos rejects clai...
2  @Microsoft Why do I pay for WORD when it funct...
3  CSGO matchmaking is so full of closet hacking,...
4  Now the President is slapping Americans in the...
```

In [3]:
```
train_df.columns = ['id', 'entity', 'sentiment', 'text']
val_df.columns = ['id', 'entity', 'sentiment', 'text']
```

In [4]:
```
pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packag
es (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packa
ges (from nltk) (8.2.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-pack
ages (from nltk) (1.5.1)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/
dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packag
es (from nltk) (4.67.1)
```

In [5]:

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
nltk.download('stopwords')
nltk.download('punkt_tab')
nltk.download('wordnet')


# Initialize tools
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def clean_text(text):
    if not isinstance(text, str):
        return ""

    # Remove URLs, mentions, punctuation
    text = re.sub(r"http\S+", "", text)
    text = re.sub(r"@\w+", "", text)
    text = re.sub(r"[^A-Za-z\s]", "", text)
    text = text.lower().strip()

    # Tokenize
    words = nltk.word_tokenize(text)

    # Remove stopwords and lemmatize
    cleaned = [lemmatizer.lemmatize(word) for word in words if word not in s

    return " ".join(cleaned)

train_df['clean_text'] = train_df['text'].apply(clean_text)
val_df['clean_text'] = val_df['text'].apply(clean_text)

# Normalize sentiment values
train_df['sentiment'] = train_df['sentiment'].str.lower().str.strip()
val_df['sentiment'] = val_df['sentiment'].str.lower().str.strip()

# Map to numeric labels
label_map = {'positive': 0, 'negative': 1, 'neutral': 2, 'irrelevant': 3}
train_df['label'] = train_df['sentiment'].map(label_map)
val_df['label'] = val_df['sentiment'].map(label_map)

# Drop invalid rows
train_df = train_df.dropna(subset=['label'])
val_df = val_df.dropna(subset=['label'])
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

In [6]:
```
from transformers import BertTokenizer

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

train_encodings = tokenizer(list(train_df['clean_text']), truncation=True, p
val_encodings = tokenizer(list(val_df['clean_text']), truncation=True, paddi

import torch

train_labels = torch.tensor(train_df['label'].values)
val_labels = torch.tensor(val_df['label'].values)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: U
serWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings t
ab (https://huggingface.co/settings/tokens), set it as secret in your Google
Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access
public models or datasets.
  warnings.warn(
```

In [7]:
```
from torch.utils.data import Dataset

class TwitterDataset(Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, idx):
        item = {key: val[idx] for key, val in self.encodings.items()}
        item['labels'] = self.labels[idx]
        return item

train_dataset = TwitterDataset(train_encodings, train_labels)
val_dataset = TwitterDataset(val_encodings, val_labels)
```

In [9]:
```
import transformers
print(transformers.__version__)
```

```
4.54.1
```

In [11]:
```
import torch
from transformers import BertTokenizer, BertForSequenceClassification, Trair

model = BertForSequenceClassification.from_pretrained('bert-base-uncased', r

training_args = TrainingArguments(
    output_dir='./results',
    eval_strategy="epoch",  # Changed from 'evaluation_strategy'
    save_strategy="epoch",
    num_train_epochs=2,
```

```
    per_device_train_batch_size=16,
    per_device_eval_batch_size=64,
    logging_dir='./logs',
    logging_steps=10,
    load_best_model_at_end=True,
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
)

trainer.train()
```

```
Some weights of BertForSequenceClassification were not initialized from the
model checkpoint at bert-base-uncased and are newly initialized: ['classifie
r.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use
it for predictions and inference.
wandb: WARNING The `run_name` is currently set to the same value as `Trainin
gArguments.output_dir`. If this was not intended, please specify a different
run name by setting the `TrainingArguments.run_name` parameter.
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: htt
ps://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/auth
orize?ref=models
wandb: Paste an API key from your profile and hit enter:wandb: WARNING If yo
u're specifying your api key in code, ensure this code is not shared publicl
y.
wandb: WARNING Consider setting the WANDB_API_KEY environment variable, or r
unning `wandb login` from the command line.
wandb: No netrc file found, creating one.
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
wandb: Currently logged in as: arjunmenon21102003 (arjunmenon21102003-go) to
https://api.wandb.ai. Use `wandb login --relogin` to force relogin
```

Tracking run with wandb version 0.21.0

Run data is saved locally in `/content/wandb/run-20250801_091337-9jyc1mek`

Syncing run **./results** to Weights & Biases (docs)

View project at https://wandb.ai/arjunmenon21102003-go/huggingface

View run at https://wandb.ai/arjunmenon21102003-go/huggingface/runs/9jyc1mek

```
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py:1750: Fut
ureWarning: `encoder_attention_mask` is deprecated and will be removed in ve
rsion 4.55.0 for `BertSdpaSelfAttention.forward`.
  return forward_call(*args, **kwargs)
```

[6512/9336 49:49 < 21:36, 2.18 it/s, Epoch 1.39/2]

Epoch	Training Loss	Validation Loss
1	0.776100	0.281535

```
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py:1750: Fut
ureWarning: `encoder_attention_mask` is deprecated and will be removed in ve
rsion 4.55.0 for `BertSdpaSelfAttention.forward`.
  return forward_call(*args, **kwargs)
```

[9336/9336 1:11:37, Epoch 2/2]

Epoch	Training Loss	Validation Loss
1	0.776100	0.281535
2	0.280600	0.152226

Out[11]:
```
TrainOutput(global_step=9336, training_loss=0.5790159923200844, metrics={'t
rain_runtime': 4461.4985, 'train_samples_per_second': 33.478, 'train_steps_
per_second': 2.093, 'total_flos': 1.2741805103107392e+16, 'train_loss': 0.5
790159923200844, 'epoch': 2.0})
```

In [12]:
```python
preds_output = trainer.predict(val_dataset)
preds = torch.argmax(torch.tensor(preds_output.predictions), axis=1)

from sklearn.metrics import classification_report

print(classification_report(val_labels, preds, target_names=label_map.keys()
```

```
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py:1750: Fut
ureWarning: `encoder_attention_mask` is deprecated and will be removed in ve
rsion 4.55.0 for `BertSdpaSelfAttention.forward`.
  return forward_call(*args, **kwargs)
              precision    recall  f1-score   support

    positive       0.95      0.96      0.96       277
    negative       0.97      0.98      0.97       266
     neutral       0.96      0.95      0.96       285
  irrelevant       0.95      0.94      0.95       172

    accuracy                           0.96      1000
   macro avg       0.96      0.96      0.96      1000
weighted avg       0.96      0.96      0.96      1000
```

In [20]:
```python
def predict_sentiment(text):
    text = clean_text(text)
    tokens = tokenizer(text, return_tensors='pt', truncation=True, padding=T
    tokens = {k: v.to(model.device) for k, v in tokens.items()}
    with torch.no_grad():
        output = model(**tokens)
        pred = torch.argmax(output.logits, dim=1).item()
    return list(label_map.keys())[pred]

# Example
print(predict_sentiment("I absolutely love this product!"))  # → positive
print(predict_sentiment("This is the worst ever."))          # → negative
print(predict_sentiment("I am coming to the borders and I will kill you all"
print(predict_sentiment("i am sad"))
print(predict_sentiment("i am angry"))
```

```python
print(predict_sentiment("nice shirt"))
print(predict_sentiment("bbsusuwu"))
```

```
positive
negative
positive
negative
negative
positive
irrelevant
```