In [3]:
```
import pandas as pd
df = pd.read_csv("train.csv")
df
```

Out[3]:

	Class Index	Title	Description
0	3	Wall St. Bears Claw Back Into the Black (Reuters)	Reuters - Short-sellers, Wall Street's dwindli...
1	3	Carlyle Looks Toward Commercial Aerospace (Reu...	Reuters - Private investment firm Carlyle Grou...
2	3	Oil and Economy Cloud Stocks' Outlook (Reuters)	Reuters - Soaring crude prices plus worries\ab...
3	3	Iraq Halts Oil Exports from Main Southern Pipe...	Reuters - Authorities have halted oil export\f...
4	3	Oil prices soar to all-time record, posing new...	AFP - Tearaway world oil prices, toppling reco...
...	...	...	...
119995	1	Pakistan's Musharraf Says Won't Quit as Army C...	KARACHI (Reuters) - Pakistani President Perve...
119996	2	Renteria signing a top-shelf deal	Red Sox general manager Theo Epstein acknowled...
119997	2	Saban not going to Dolphins yet	The Miami Dolphins will put their courtship of...
119998	2	Today's NFL games	PITTSBURGH at NY GIANTS Time: 1:30 p.m. Line: ...
119999	2	Nets get Carter from Raptors	INDIANAPOLIS -- All-Star Vince Carter was trad...

120000 rows × 3 columns

Combining Description and Title into text column

In [4]:
```
df.columns = ["category" , "title" , "text1"]
df["text"] = df["title"] + " " + df["text1"]
df = df[["category", "text"]]
df
```

Out[4]:

	category	text
0	3	Wall St. Bears Claw Back Into the Black (Reute...
1	3	Carlyle Looks Toward Commercial Aerospace (Reu...
2	3	Oil and Economy Cloud Stocks' Outlook (Reuters...
3	3	Iraq Halts Oil Exports from Main Southern Pipe...
4	3	Oil prices soar to all-time record, posing new...
...	...	...
119995	1	Pakistan's Musharraf Says Won't Quit as Army C...
119996	2	Renteria signing a top-shelf deal Red Sox gene...
119997	2	Saban not going to Dolphins yet The Miami Dolp...
119998	2	Today's NFL games PITTSBURGH at NY GIANTS Time...
119999	2	Nets get Carter from Raptors INDIANAPOLIS -- A...

120000 rows × 2 columns

In [5]:
```
numlabels = len(df.category.unique())
numlabels
```

Out[5]: 4

In [6]:
```
df['category'] = df['category'].astype(str)
```

```
C:\Users\AY\AppData\Local\Temp\ipykernel_18484\648958893.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['category'] = df['category'].astype(str)
```

Created sample df in case it takes a long time to train original dataset

In [19]:
```
grouped = df.groupby('category')

sampled_data = []
for label, group in grouped:
    sampled_group = group.sample(min(10000, len(group)))
    sampled_data.append(sampled_group)

sampled_df = pd.concat(sampled_data)
#df = sampled_df.copy()
```

Converting string to integer because bert needs ints that start form 0

In [1]:
```
import torch
import numpy as np
from transformers import BertTokenizer

tokenizer = BertTokenizer.from_pretrained('bert-base-cased')
labels = {'1':0,
          '2':1,
          '3':2,
          '4':3,}

class Dataset(torch.utils.data.Dataset):

    def __init__(self, df):

        self.labels = [labels[label] for label in df['category']]
        self.texts = [tokenizer(text,
                                padding='max_length', max_length = 128, truncation=True,
                                 return_tensors="pt") for text in df['text']]

    def classes(self):
        return self.labels

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

    def get_batch_labels(self, idx):
        # Fetch a batch of labels
        return np.array(self.labels[idx])

    def get_batch_texts(self, idx):
        # Fetch a batch of inputs
        return self.texts[idx]

    def __getitem__(self, idx):

        batch_texts = self.get_batch_texts(idx)
        batch_y = self.get_batch_labels(idx)

        return batch_texts, batch_y
```

Train test split, standard pratice

In [22]:
```
np.random.seed(112)
df_train, df_val, df_test = np.split(df.sample(frac=1, random_state=42),
                                     [int(.8*len(df)), int(.9*len(df))])
```

In [ ]:
```
```

In [23]:
```
from torch import nn
from transformers import BertModel

class BertClassifier(nn.Module):

    def __init__(self, dropout=0.5):

        super(BertClassifier, self).__init__()

        self.bert = BertModel.from_pretrained('bert-base-cased')
        self.dropout = nn.Dropout(dropout)
        self.linear = nn.Linear(768, numlabels)
        self.relu = nn.ReLU()

    def forward(self, input_id, mask):

        _, pooled_output = self.bert(input_ids= input_id, attention_mask=mask,return_dict=False)
        dropout_output = self.dropout(pooled_output)
        linear_output = self.linear(dropout_output)
        final_layer = self.relu(linear_output)
        return final_layer
```

In [ ]:
```
```

In [24]:
```
from torch.optim import Adam
from tqdm import tqdm

def train(model, train_data, val_data, learning_rate, epochs):

    train, val = Dataset(train_data), Dataset(val_data)

    train_dataloader = torch.utils.data.DataLoader(train, batch_size=8, shuffle=True)
    val_dataloader = torch.utils.data.DataLoader(val, batch_size=8)

    use_cuda = torch.cuda.is_available()
    device = torch.device("cuda" if use_cuda else "cpu")

    criterion = nn.CrossEntropyLoss()
```

```
optimizer = Adam(model.parameters(), lr= learning_rate)

if use_cuda:

        model = model.cuda()
        criterion = criterion.cuda()

for epoch_num in range(epochs):

        total_acc_train = 0
        total_loss_train = 0

        for train_input, train_label in tqdm(train_dataloader):

            train_label = train_label.to(device)
            mask = train_input['attention_mask'].to(device)
            input_id = train_input['input_ids'].squeeze(1).to(device)

            output = model(input_id, mask)

            batch_loss = criterion(output, train_label.long())
            total_loss_train += batch_loss.item()

            acc = (output.argmax(dim=1) == train_label).sum().item()
            total_acc_train += acc

            model.zero_grad()
            batch_loss.backward()
            optimizer.step()

        total_acc_val = 0
        total_loss_val = 0

        with torch.no_grad():

            for val_input, val_label in val_dataloader:

                val_label = val_label.to(device)
                mask = val_input['attention_mask'].to(device)
                input_id = val_input['input_ids'].squeeze(1).to(device)

                output = model(input_id, mask)

                batch_loss = criterion(output, val_label.long())
                total_loss_val += batch_loss.item()

                acc = (output.argmax(dim=1) == val_label).sum().item()
                total_acc_val += acc

        print(
            f'Epochs: {epoch_num + 1} | Train Loss: {total_loss_train / len(train_data): .3f} \
            | Train Accuracy: {total_acc_train / len(train_data): .3f} \
            | Val Loss: {total_loss_val / len(val_data): .3f} \
            | Val Accuracy: {total_acc_val / len(val_data): .3f}')

EPOCHS = 3
model = BertClassifier()
LR = 7e-6

train(model, df_train, df_val, LR, EPOCHS)
```

```
100%|██████████| 12000/12000 [14:08<00:00, 14.14it/s]
Epochs: 1 | Train Loss:  0.029 | Train Accuracy:  0.922 | Val Loss:  0.023 | Val Accuracy:  0.937
100%|██████████| 12000/12000 [14:26<00:00, 13.86it/s]
Epochs: 2 | Train Loss:  0.015 | Train Accuracy:  0.960 | Val Loss:  0.023 | Val Accuracy:  0.942
100%|██████████| 12000/12000 [14:04<00:00, 14.21it/s]
Epochs: 3 | Train Loss:  0.007 | Train Accuracy:  0.982 | Val Loss:  0.025 | Val Accuracy:  0.944
```

In [ ]:

In [25]:
```
def evaluate(model, test_data):

    test = Dataset(test_data)

    test_dataloader = torch.utils.data.DataLoader(test, batch_size=2)

    use_cuda = torch.cuda.is_available()
    device = torch.device("cuda" if use_cuda else "cpu")

    if use_cuda:

        model = model.cuda()

    total_acc_test = 0
    with torch.no_grad():

        for test_input, test_label in test_dataloader:

              test_label = test_label.to(device)
```

```
            mask = test_input['attention_mask'].to(device)
            input_id = test_input['input_ids'].squeeze(1).to(device)

            output = model(input_id, mask)

            acc = (output.argmax(dim=1) == test_label).sum().item()
            total_acc_test += acc

    print(f'Test Accuracy: {total_acc_test / len(test_data): .3f}')

evaluate(model, df_test)
```

```
Test Accuracy:  0.942
```