```
!wget https://data.deepai.org/quora_question_pairs.zip
!unzip -q quora_question_pairs
!unzip -q train.csv.zip
!unzip -q test.csv.zip
```

```
--2021-11-14 06:44:21--  https://data.deepai.org/quora_question_pairs.zip
Resolving data.deepai.org (data.deepai.org)... 138.201.36.183
Connecting to data.deepai.org (data.deepai.org)|138.201.36.183|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 323899260 (309M) [application/x-zip-compressed]
Saving to: ‘quora_question_pairs.zip’

quora_question_pair 100%[===================>] 308.89M  20.4MB/s    in 16s

2021-11-14 06:44:38 (19.6 MB/s) - ‘quora_question_pairs.zip’ saved [323899260/323899260]

replace test.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
```

```
!pip install transformers
```

```
Collecting transformers
  Downloading transformers-4.12.3-py3-none-any.whl (3.1 MB)
     |████████████████████████████████| 3.1 MB 7.7 MB/s
Collecting pyyaml>=5.1
  Downloading PyYAML-6.0-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux20
     |████████████████████████████████| 596 kB 73.8 MB/s
Collecting tokenizers<0.11,>=0.10.1
  Downloading tokenizers-0.10.3-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.many
     |████████████████████████████████| 3.3 MB 27.0 MB/s
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from transformers) (3.3.2
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from transforme
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-packages (from transformers)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from transformers) (2.23.
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/dist-packages (from transformer
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-packages (from transformers) (4.6
Collecting huggingface-hub<1.0,>=0.1.0
  Downloading huggingface_hub-0.1.2-py3-none-any.whl (59 kB)
     |████████████████████████████████| 59 kB 8.6 MB/s
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (from transformers) (1.
Collecting sacremoses
  Downloading sacremoses-0.0.46-py3-none-any.whl (895 kB)
     |████████████████████████████████| 895 kB 44.8 MB/s
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.7/dist-packages (from hu
Requirement already satisfied: pyparsing<3,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata-
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->transf
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->t
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from sacremoses->transformers)
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (from sacremoses->transformer
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from sacremoses->transforme
Installing collected packages: pyyaml, tokenizers, sacremoses, huggingface-hub, transformers
  Attempting uninstall: pyyaml
    Found existing installation: PyYAML 3.13
    Uninstalling PyYAML-3.13:
      Successfully uninstalled PyYAML-3.13
Successfully installed huggingface-hub-0.1.2 pyyaml-6.0 sacremoses-0.0.46 tokenizers-0.10.3 transformers-4.1
```

```
# from preprocessing import preprocess
from transformers import DistilBertTokenizer, DistilBertModel
import torch
from torch import nn, optim
import copy
import random
import sklearn.metrics
import tqdm
import pickle
import pandas as pd
import numpy as np
import math


def shuffle_data(input_1, input_2, labels):
    shuffled_input_1 = []
    shuffled_input_2 = []
    shuffled_labels      = []
    indices = list(range(len(input_1)))
    random.shuffle(indices)
    for i in indices:
        shuffled_input_1.append(input_1[i])
        shuffled_input_2.append(input_2[i])
        shuffled_labels.append(labels[i])
```

```
      return (shuffled_input_1, shuffled_input_2, shuffled_labels)


def train(model, tokenizer, X_1, X_2, Y, learning_rate=0.1, batch_size=8, num_epochs=5):
  optimizer = optim.Adam(model.parameters(), lr=learning_rate)
  for epoch in range(num_epochs):
      total_loss = 0.0
      (shuffled_input_1, shuffled_input_2, shuffled_labels) = shuffle_data(X_1, X_2, Y)
      for batch in tqdm.notebook.tqdm(range(0, len(X_1), batch_size), leave=False):
          #Randomly shuffle examples in each epoch
          input_1 = shuffled_input_1[batch:(batch + batch_size)]
          input_2 = shuffled_input_2[batch:(batch + batch_size)]
          encoded_input_1 = tokenizer(input_1, return_tensors='pt', is_split_into_words=False, padding=True)
          encoded_input_2 = tokenizer(input_2, return_tensors='pt', is_split_into_words=False, padding=True)
          labels = shuffled_labels[batch:(batch + batch_size)]
          labels_onehot = torch.zeros(len(labels), num_classes).cuda()
          for i in range(len(labels)):
              labels_onehot[i][labels[i]] = 1.0
          model.zero_grad()
          log_probs = model.forward(encoded_input_1, encoded_input_2, train=True)
          # print(log_probs)
          loss_batch = 0
          for idx in range(labels_onehot.shape[0]):
              loss_iteration = torch.neg(log_probs[idx]).dot(labels_onehot[idx])
              loss_batch += loss_iteration
          loss_batch /= labels_onehot.shape[0]
          loss_batch.backward()
          nn.utils.clip_grad_norm_(model.parameters(), 1.0)
          optimizer.step()
          total_loss += loss_batch.detach()
      num_batches = math.ceil(len(X_1) / batch_size)
      print(f"avg loss on epoch {epoch} = {total_loss / num_batches}")


def get_predictions(model, X_1, X_2, batch_size=8):
  all_predictions = np.array([])
  for batch in tqdm.notebook.tqdm(range(0, len(X_1), batch_size), leave=False):
    encoded_input_1 = tokenizer(X_1[batch:batch + batch_size], return_tensors='pt', is_split_into_words=False, pad
    encoded_input_2 = tokenizer(X_2[batch:batch + batch_size], return_tensors='pt', is_split_into_words=False, pad
    log_probs = model.forward(encoded_input_1, encoded_input_2, train=False)
    prediction_batch = torch.argmax(log_probs, dim=1)
    all_predictions = np.concatenate((all_predictions, prediction_batch.cpu().numpy()))
  return all_predictions


def get_predictions_cosine_similarity(model, tokenizer, X_1, X_2, threshold=0.96, batch_size=8):
  all_predictions = np.array([])
  for batch in tqdm.notebook.tqdm(range(0, len(X_1), batch_size), leave=False):
    encoded_input_1 = tokenizer(X_1[batch:batch + batch_size], return_tensors='pt', is_split_into_words=False, pad
    encoded_input_2 = tokenizer(X_2[batch:batch + batch_size], return_tensors='pt', is_split_into_words=False, pad
    pooler_output_1 = model(encoded_input_1['input_ids'].cuda(), encoded_input_1['attention_mask'].cuda()).last_hi
    pooler_output_2 = model(encoded_input_2['input_ids'].cuda(), encoded_input_2['attention_mask'].cuda()).last_hi
    cos = nn.CosineSimilarity(dim=1, eps=1e-6)
    output = cos(pooler_output_1, pooler_output_2).cpu().detach().numpy()
    preds = []
    for i in range(output.shape[0]):
      if output[i] > threshold:
        preds.append(1)
      else:
        preds.append(0)
    all_predictions = np.concatenate((all_predictions, preds))
  return all_predictions


def evaluate(Y, predictions):
  print("Accuracy: {}".format(sklearn.metrics.accuracy_score(Y, predictions)))
  print("F1 score: {}".format(sklearn.metrics.f1_score(Y, predictions)))
  print("Precision: {}".format(sklearn.metrics.precision_score(Y, predictions)))
  print("Recall: {}".format(sklearn.metrics.recall_score(Y, predictions)))
  print("Confusion matrix: \n{}\n".format(sklearn.metrics.confusion_matrix(Y, predictions)))


# Raw csv
# df = pd.read_csv('train.csv')
# data = [list(df["question1"]), list(df["question2"]), list(df["is_duplicate"])]

# Preprocessed w/o transitivity
# with open('/content/drive/MyDrive/processed_data_wo_transitive.pkl', 'rb') as f:
#    data = pickle.load(f)

# Preprocessed w/ transitivity
# Finalized transitivity based on its superiority
with open('/content/drive/MyDrive/processed_data_1.pkl', 'rb') as f:
```

```
      data = pickle.load(f)

print("Original data has {} question pairs".format(len(data[0])))
```

```
    Original data has 799498 question pairs
```

```
size = 100000
dataset = [data[i][:size] for i in range(len(data))]
print("Reduced the dataset to first {} pairs".format(len(dataset[0])))
```

```
    Reduced the dataset to first 100000 pairs
```

```
train_ratio = 0.8
indices = list(range(size))
random.shuffle(indices)
train_indices = indices[:int(size*train_ratio)]
test_indices = indices[int(size*train_ratio):]
train_dataset = [[dataset[i][j] for j in train_indices] for i in range(len(dataset))]
test_dataset = [[dataset[i][j] for j in test_indices] for i in range(len(dataset))]

train_input_1 = [" ".join(train_dataset[0][i]) for i in range(len(train_dataset[0]))]
train_input_2 = [" ".join(train_dataset[1][i]) for i in range(len(train_dataset[1]))]
# train_input_1 = [""].join(train_dataset[0][i]) for i in range(len(train_dataset[0]))]
# train_input_2 = [""].join(train_dataset[1][i]) for i in range(len(train_dataset[1]))]
train_Y = train_dataset[2]
print(len(train_input_1), len(train_input_2), len(train_Y))
num_classes = 2

test_input_1 = [" ".join(test_dataset[0][i]) for i in range(len(test_dataset[0]))]
test_input_2 = [" ".join(test_dataset[1][i]) for i in range(len(test_dataset[1]))]
test_Y = test_dataset[2]
print(len(test_input_1), len(test_input_2), len(test_Y))
num_classes = 2
```

```
    80000 80000 80000
    20000 20000 20000
```

```
class SimilarityModelFineTuneBert(nn.Module):
    def __init__(self):
        super(SimilarityModelFineTuneBert, self).__init__()
        self.bert = DistilBertModel.from_pretrained("distilbert-base-uncased").cuda()
        self.feedforward_1 = nn.Linear(768*2, 300).cuda()
        self.non_lin_1 = nn.PReLU().cuda()
        self.dropout = nn.Dropout(p=0.5)
        self.feedforward_2 = nn.Linear(300, 300).cuda()
        self.non_lin_2 = nn.PReLU().cuda()
        self.feedforward_3 = nn.Linear(300, 2).cuda()
        self.log_softmax = nn.LogSoftmax(dim=0).cuda()

    def forward(self, encoded_input_1, encoded_input_2, train=False):
        if train:
          self.train()
        else:
          self.eval()
        pooler_output_1 = self.bert(encoded_input_1['input_ids'].cuda(), encoded_input_1['attention_mask'].cuda())
        pooler_output_2 = self.bert(encoded_input_2['input_ids'].cuda(), encoded_input_2['attention_mask'].cuda())
        # print(pooler_output_1)
        concatenated_output = torch.cat([pooler_output_1, pooler_output_2], axis=1).cuda()
        f1 = self.dropout(self.non_lin_1(self.feedforward_1(concatenated_output)))
        f2 = self.dropout(self.non_lin_2(self.feedforward_2(f1)))
        return self.log_softmax(self.feedforward_3(f2))


tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
print("Training fine tune model")
fine_tune_model = SimilarityModelFineTuneBert()
train(fine_tune_model, tokenizer, train_input_1, train_input_2, train_Y, learning_rate=0.0001, num_epochs=5, batch
```

```
    Training fine tune model
    Some weights of the model checkpoint at distilbert-base-uncased were not used when initializing DistilBertMo
    - This IS expected if you are initializing DistilBertModel from the checkpoint of a model trained on another
    - This IS NOT expected if you are initializing DistilBertModel from the checkpoint of a model that you expec
    avg loss on epoch 0 = 0.03698580712080002
    avg loss on epoch 1 = 0.03617806360125542
    avg loss on epoch 2 = 0.035314884036779404
    avg loss on epoch 3 = 0.03456328436732292
    avg loss on epoch 4 = 0.034070178866386414
```

```
print("Evaluating fine tune model on train dataset")
```

```
predictions = get_predictions(fine_tune_model, train_input_1, train_input_2, batch_size=128)
evaluate(train_Y, predictions)

print("Evaluating fine tune bert model on test dataset")
predictions = get_predictions(fine_tune_model, test_input_1, test_input_2, batch_size=128)
evaluate(test_Y, predictions)
```

```
Evaluating fine tune model on train dataset
Accuracy: 0.9733625
F1 score: 0.9650202721557427
Precision: 0.943326594140111
Recall: 0.9877352150537635
Confusion matrix:
[[48474  1766]
 [  365 29395]]

Evaluating fine tune bert model on test dataset
Accuracy: 0.7609
F1 score: 0.6953363914373089
Precision: 0.6653255303584492
Recall: 0.7281825460368294
Confusion matrix:
[[9761 2745]
 [2037 5457]]
```

```
# Stopped working on this after evaluating a baseline

tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
vanilla = DistilBertModel.from_pretrained("distilbert-base-uncased").cuda()
print("Evaluating vanilla model on train dataset")
print("NOTE: For vanilla model with cosine based similarity, train-test split doesn't matter\n")
predictions = get_predictions_cosine_similarity(vanilla, tokenizer, train_input_1, train_input_2, 0.96)
evaluate(train_Y, predictions)

print("Evaluating vanilla model on test dataset")
print("NOTE: For vanilla model with cosine based similarity, train-test split doesn't matter\n")
predictions = get_predictions_cosine_similarity(vanilla, tokenizer, test_input_1, test_input_2, 0.96)
evaluate(test_Y, predictions)
```

```
Some weights of the model checkpoint at distilbert-base-uncased were not us
- This IS expected if you are initializing DistilBertModel from the checkpc
- This IS NOT expected if you are initializing DistilBertModel from the che
Evaluating vanilla model on train dataset
NOTE: For vanilla model with cosine based similarity, train-test split does

Accuracy: 0.598375
F1 score: 0.6064911206368647
Precision: 0.4746932515337423
Recall: 0.8396066463207867
Confusion matrix:
[[2311 2740]
 [ 473 2476]]

Evaluating vanilla model on test dataset
NOTE: For vanilla model with cosine based similarity, train-test split does

Accuracy: 0.6035
F1 score: 0.6170931916948333
Precision: 0.4881588999236058
Recall: 0.8385826771653543
Confusion matrix:
[[568 670]
 [123 639]]
```

```
#Stopped training this after superiority of finetuning BERT was realized

class SimilarityModelStaticBert(nn.Module):
    def __init__(self):
        super(SimilarityModelStaticBert, self).__init__()
        self.bert = DistilBertModel.from_pretrained("distilbert-base-uncased").cuda()
        for param in self.bert.parameters():
            param.requires_grad = False
        self.feedforward_1 = nn.Linear(768*2, 300).cuda()
        self.non_lin_1 = nn.PReLU().cuda()
        self.feedforward_2 = nn.Linear(300, 300).cuda()
        self.non_lin_2 = nn.PReLU().cuda()
        self.feedforward_3 = nn.Linear(300, 2).cuda()
        self.log_softmax = nn.LogSoftmax(dim=0).cuda()

    def forward(self, encoded_input_1, encoded_input_2, train=False):
        if train:
          self.train()
```

```
        else:
          self.eval()
        pooler_output_1 = self.bert(encoded_input_1['input_ids'].cuda(), encoded_input_1['attention_mask'].cuda())
        pooler_output_2 = self.bert(encoded_input_2['input_ids'].cuda(), encoded_input_2['attention_mask'].cuda())
        # print(pooler_output_1)
        concatenated_output = torch.cat([pooler_output_1, pooler_output_2], axis=1).cuda()
        f1 = self.non_lin_1(self.feedforward_1(concatenated_output))
        f2 = self.non_lin_2(self.feedforward_2(f1))
        return self.log_softmax(self.feedforward_3(f2))


tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
print("Training static bert model")
static_bert_model = SimilarityModelStaticBert()
train(static_bert_model, tokenizer, train_input_1, train_input_2, train_Y, learning_rate=0.0001, num_epochs=10, ba
```

```
Training static bert model
Some weights of the model checkpoint at distilbert-base-uncased were not us
- This IS expected if you are initializing DistilBertModel from the checkpo
- This IS NOT expected if you are initializing DistilBertModel from the che
avg loss on epoch 0 = 0.10724116116762161
avg loss on epoch 1 = 0.10655724257230759
avg loss on epoch 2 = 0.10627871006727219
avg loss on epoch 3 = 0.10622921586036682
avg loss on epoch 4 = 0.10600847005844116
avg loss on epoch 5 = 0.10582157224416733
avg loss on epoch 6 = 0.10573685169219971
avg loss on epoch 7 = 0.10562307387590408
avg loss on epoch 8 = 0.10542453825473785
avg loss on epoch 9 = 0.10536253452301025
```

```
print("Evaluating static bert model on train dataset")
predictions = get_predictions(static_bert_model, train_input_1, train_input_2, batch_size=32)
evaluate(train_Y, predictions)

print("Evaluating static bert model on test dataset")
predictions = get_predictions(static_bert_model, test_input_1, test_input_2, batch_size=32)
evaluate(test_Y, predictions)
```

```
Evaluating static bert model on train dataset
Accuracy: 0.70625
F1 score: 0.6392385630948726
Precision: 0.5840112201963534
Recall: 0.7060020345879959
Confusion matrix:
[[3568 1483]
 [ 867 2082]]

Evaluating static bert model on test dataset
Accuracy: 0.6835
F1 score: 0.6165960024227742
Precision: 0.5725534308211474
Recall: 0.6679790026246719
Confusion matrix:
[[858 380]
 [253 509]]
```

✓ 0s completed at 5:03 AM