

Imports

```
!pip install transformers
!pip install sentencepiece

Collecting transformers
  Downloading transformers-4.12.5-py3-none-any.whl (3.1 MB)
    |████████████████████████████████████████| 3.1 MB 12.6 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.manylinux2014_x86_64.whl (596 kB)
    |████████████████████████████████████████| 596 kB 57.5 MB/s
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from transformers) (3.4.0)
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.manylinux2014_x86_64.whl (3.3 MB)
    |████████████████████████████████████████| 3.3 MB 54.2 MB/s
Collecting sacremoses
  Downloading sacremoses-0.0.46-py3-none-any.whl (895 kB)
    |████████████████████████████████████████| 895 kB 60.6 MB/s
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (from transformers) (1.21.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-packages (from transformers) (21.3)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/dist-packages (from transformers) (2022.10.31)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from transformers) (2.28.1)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-packages (from transformers) (4.64.1)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from transformers) (4.11.0)
Collecting huggingface-hub<1.0,>=0.1.0
  Downloading huggingface_hub-0.2.1-py3-none-any.whl (61 kB)
    |████████████████████████████████████████| 61 kB 661 kB/s
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.7/dist-packages (from huggingface-hub) (4.5.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging) (3.1.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata) (3.15.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests) (2022.12.7)
Requirement already satisfied: urllib3!=1.25.0,!<1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests) (1.26.15)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests) (3.4)
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (from sacremoses->transformers) (8.1.3)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from sacremoses->transformers) (1.2.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from sacremoses->transformers) (1.16.0)
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-0.2.1 pyyaml-6.0 sacremoses-0.0.46 tokenizers-0.10.3 transformers-4.12.5

Collecting sentencepiece
  Downloading sentencepiece-0.1.96-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.2 MB)
    |████████████████████████████████████████| 1.2 MB 14.5 MB/s
Installing collected packages: sentencepiece
Successfully installed sentencepiece-0.1.96
```

```
# from preprocessing import preprocess
from transformers import XLNetTokenizer, XLNetModel
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
```

Helper Functions

```
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, padding=True)
        encoded_input_2 = tokenizer(X_2[batch:batch + batch_size], return_tensors='pt', is_split_into_words=False, padding=True)
        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, padding=True)
        encoded_input_2 = tokenizer(X_2[batch:batch + batch_size], return_tensors='pt', is_split_into_words=False, padding=True)
        pooler_output_1 = model(encoded_input_1['input_ids'].cuda(), encoded_input_1['attention_mask'].cuda()).last_hidden_state
        pooler_output_2 = model(encoded_input_2['input_ids'].cuda(), encoded_input_2['attention_mask'].cuda()).last_hidden_state
        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)))
```

▼ Data Loader

```
# 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_2.pkl', 'rb') as f:
    data = pickle.load(f)

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

    Original data has 577240 question pairs

size = len(data[0])
dataset = data[:][:size]
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

    461792 461792 461792
    115448 115448 115448
```

▼ Fine Tune Model

```
class SimilarityModelFineTuneXLNet(nn.Module):
    def __init__(self, dropout_rate=0.25):
        super(SimilarityModelFineTuneXLNet, self).__init__()
        self.xlnet = XLNetModel.from_pretrained("xlnet-base-cased").cuda()
        self.dropout = nn.Dropout(p=dropout_rate)
        self.feedforward_1 = nn.Linear(768*2, 300).cuda()
        self.non_lin_1 = nn.PReLU().cuda()
        self.dropout = nn.Dropout(p=dropout_rate)
        self.feedforward_2 = nn.Linear(300, 300).cuda()
        self.non_lin_2 = nn.PReLU().cuda()
        self.dropout = nn.Dropout(p=dropout_rate)
        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.xlnet(input_ids=encoded_input_1['input_ids'].cuda(), attention_mask=encoded_input_1['attention_mask'].cuda())
        pooler_output_2 = self.xlnet(input_ids=encoded_input_2['input_ids'].cuda(), attention_mask=encoded_input_2['attention_mask'].cuda())
        # print(pooler_output_1.keys())
        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))
```

▼ Config

```
dropout_rate = 0.25
batch_size = 64
learning_rate = 0.0001
num_epochs = 5
```

▼ Train

```
tokenizer = XLNetTokenizer.from_pretrained("xlnet-base-cased")
```

```
print("Training fine tune model")
fine_tune_model = SimilarityModelFineTuneXLNet()
train(fine_tune_model, tokenizer, train_input_1, train_input_2, train_Y, learning_rate=learning_rate, num_epochs=r

Downloading: 100% 779k/779k [00:00<00:00, 628kB/s]

Downloading: 100% 1.32M/1.32M [00:00<00:00, 1.27MB/s]

Downloading: 100% 760/760 [00:00<00:00, 22.1kB/s]
Training fine tune model
Downloading: 100% 445M/445M [00:08<00:00, 54.8MB/s]
Some weights of the model checkpoint at xlnet-base-cased were not used when initializing XLNetModel: ['lm_lo:
- This IS expected if you are initializing XLNetModel from the checkpoint of a model trained on another task
- This IS NOT expected if you are initializing XLNetModel from the checkpoint of a model that you expect to l
avg loss on epoch 0 = 4.157436370849609
avg loss on epoch 1 = 4.157904148101807
avg loss on epoch 2 = 4.157471656799316
avg loss on epoch 3 = 4.156764507293701
avg loss on epoch 4 = 4.15702486038208

with open("xlnet-ffn-fine-tune-pickle.pkl", 'wb') as f:
    pickle.dump(fine_tune_model, f)

torch.save(fine_tune_model.state_dict(), 'xlnet-ffn-fine-tune-state-dict.pt')
```

▼ Eval

```
print("Evaluating fine tune model on train dataset")
predictions = get_predictions(fine_tune_model, train_input_1, train_input_2, batch_size=batch_size)
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=batch_size)
evaluate(test_Y, predictions)

[ ] Evaluating fine tune model on train dataset
Accuracy: 0.6460159552352575
F1 score: 0.0
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: UndefinedMetricWarning: Prec:
    _warn_prf(average, modifier, msg_start, len(result))
Precision: 0.0
Recall: 0.0
Confusion matrix:
[[298325      0]
 [163467      0]]

Evaluating fine tune bert model on test dataset
Accuracy: 0.6437703554847204
F1 score: 0.0
Precision: 0.0
Recall: 0.0
Confusion matrix:
[[74322      0]
 [41126      0]]

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: UndefinedMetricWarning: Prec:
    _warn_prf(average, modifier, msg_start, len(result))
```

!

1s

completed at 2:01 AM

×