lab6

October 17, 2023

[42]:
```
import pandas as pd
```

[43]:
```
pd.set_option('display.max_colwidth', 2000)
pd.set_option('display.max_rows', 2000)
# Load the data
file_path = './dataSets/complaints_processed.csv'
data = pd.read_csv(file_path)

# Print the first 5 rows
data.head()

data['narrative'] = data['narrative'].astype(str)


#Count number of rows in data set
print(len(data))

#Only use 1000 rows of data
data = data.sample(n=1000, random_state=42)

#Count number of rows in data set
print(len(data))
```

```
162421
1000
```

[44]:
```
#Clean the data
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk
import string


# It's a good practice to download the necessary NLTK data beforehand
nltk.download('punkt')
nltk.download('stopwords')
def preprocess_text(text):
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = text.lower()
    stop_words = set(stopwords.words('english'))
    word_tokens = word_tokenize(text)
    filtered_text = [word for word in word_tokens if word not in stop_words]
    filtered_text = [word for word in filtered_text if len(word) >= 3]
    text = " ".join(filtered_text)
    return text

# Apply the preprocess_text function to the 'narrative' column of your data
data['narrative'] = data['narrative'].astype(str).apply(preprocess_text)
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/hudsonshimanyula/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/hudsonshimanyula/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[45]:
```
#Display the first 5 rows of your data to make sure the preprocessing happened␣
 ↪correctly
data.head()
```

[45]:
```
        Unnamed: 0             product  \
156566      156566  mortgages_and_loans
1498          1498    credit_reporting
134991      134991    credit_reporting
56391        56391  mortgages_and_loans
9067          9067    credit_reporting


                                                                  narrative
156566
penfed asking copy driver license finalizing loan american customer
1498
collection account removed credit report franklin collection service credit
score increase removal collection account credit report increased credit score
least point
134991
bureau falsely reporting alleged debt fdcpa section violation usc alleged debt
verified yet receive response day another violation fcra
56391   mortgage well fargo bank since meet condition streamline refinance filed
refi application provided documentation requested immediately online updated
document month dragged foot kept asking documentation rate lock extended expires
rate lock extended numerous time law firm closing trying get give date closing
provide date given asked month ago representative well fargo explain low
interest rate swamped refinances therefore behind month plus received loan
estimate told loan passed final underwriting approval known date close point
rate lock expires
```

```
9067
bank xxxxi credit card mine
```

[46]:
```python
import torch
from transformers import BertTokenizer, BertForSequenceClassification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_recall_fscore_support
import pandas as pd
from torch.utils.data import DataLoader, TensorDataset
```

[47]:
```python
# Preprocess the data
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
inputs = tokenizer(data['narrative'].tolist(), truncation=True, padding=True,␣
 ↪return_tensors='pt')
labels = torch.tensor(data['product'].astype('category').cat.codes.tolist())
```

[48]:
```python
# Split the data into training and testing sets
train_inputs, test_inputs, train_labels, test_labels = train_test_split(
    inputs['input_ids'], labels, test_size=0.2, random_state=42)
```

[49]:
```python
# Create torch DataLoaders for training and testing data
train_data = TensorDataset(train_inputs, train_labels)
train_dataloader = DataLoader(train_data, batch_size=32, shuffle=True)

test_data = TensorDataset(test_inputs, test_labels)
test_dataloader = DataLoader(test_data, batch_size=32, shuffle=False)

num_labels = data['product'].nunique()
print(num_labels)
```

```
5
```

[50]:
```
!pip install tqdm
```

```
Requirement already satisfied: tqdm in
/Users/hudsonshimanyula/anaconda3/envs/AI_MASTERS_ENV/lib/python3.11/site-
packages (4.65.0)
```

[51]:
```python
from tqdm import tqdm
# Load the model
model = BertForSequenceClassification.from_pretrained('bert-base-uncased',␣
 ↪num_labels=num_labels)

# Define the training parameters
optimizer = torch.optim.Adam(model.parameters(), lr=2e-5)
loss_fn = torch.nn.CrossEntropyLoss()
```

```
# Train the model
for epoch in range(3):
    model.train()
    epoch_loss = 0  # Initialize the epoch loss
    # Wrap your dataloader with tqdm to show a progress bar
    for batch in tqdm(train_dataloader, desc=f"Epoch {epoch + 1}"):
        optimizer.zero_grad()
        input_ids, labels = batch
        outputs = model(input_ids, labels=labels)
        loss = loss_fn(outputs.logits, labels)
        loss.backward()
        optimizer.step()
        epoch_loss += loss.item()  # Update the epoch loss
```

```
Some weights of BertForSequenceClassification were not initialized from the
model checkpoint at bert-base-uncased and are newly initialized:
['classifier.weight', 'classifier.bias']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
Epoch 1: 100%|      | 25/25 [12:24<00:00, 29.76s/it]
Epoch 2: 100%|      | 25/25 [12:15<00:00, 29.43s/it]
Epoch 3: 100%|      | 25/25 [12:16<00:00, 29.47s/it]
```

[54]:
```
# Evaluate the model
model.eval()
predictions = []
true_labels = []
for batch in test_dataloader:
    input_ids, labels = batch
    with torch.no_grad():
        outputs = model(input_ids)
    logits = outputs.logits
    predicted_labels = torch.argmax(logits, dim=1).tolist()
    predictions.extend(predicted_labels)
    true_labels.extend(labels.tolist())
```

[53]:
```
# Calculate performance metrics
accuracy = accuracy_score(true_labels, predictions)
precision, recall, f1, _ = precision_recall_fscore_support(true_labels,␣
 ↪predictions, average='weighted')

# Output the evaluation metrics
print(f'Accuracy: {accuracy*100:.2f}%')
print(f'Precision: {precision*100:.2f}%')
print(f'Recall: {recall*100:.2f}%')
print(f'F1 Score: {f1*100:.2f}%')
```

```
Accuracy: 58.00%
Precision: 44.26%
Recall: 58.00%
F1 Score: 49.15%
```

```
/Users/hudsonshimanyula/anaconda3/envs/AI_MASTERS_ENV/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```