spam_detection_model_training

March 30, 2025

[32]:
```
import os
for file in ['spam_model.pkl', 'spam_data.npz', 'vectorizer.pkl']:
    if os.path.exists(file):
        os.remove(file)
        print(f"{file} removed")
```

[33]:
```
import joblib
import os
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import SGDClassifier
from typing import List
import pandas as pd

class SpamDetectionModel:
    MODEL_FILE = "spam_model.pkl"
    DATA_FILE = "spam_data.npz"
    VECTOR_FILE = "vectorizer.pkl"

    def __init__(self):
        self.texts = []
        self.labels = []

        self.check_and_create_files()

        self.vectorizer = joblib.load(self.VECTOR_FILE)
        self.classifier = joblib.load(self.MODEL_FILE)
        self.load_training_data()

    def check_and_create_files(self):
        if not os.path.exists(self.VECTOR_FILE):
            # print(f"File {self.VECTOR_FILE} not found. Creating new")
            joblib.dump(CountVectorizer(), self.VECTOR_FILE)

        if not os.path.exists(self.MODEL_FILE):
            # print(f"File {self.MODEL_FILE} not found. Creating new")
            joblib.dump(SGDClassifier(loss='log_loss'), self.MODEL_FILE)
        if not os.path.exists(self.DATA_FILE):
            # print(f"File {self.DATA_FILE} not found. Creating new")
            self.save_training_data()

    def load_training_data(self):
        if os.path.exists(self.DATA_FILE) and os.path.getsize(self.DATA_FILE) >␣
↪0:
            data = np.load(self.DATA_FILE, allow_pickle=True)
            self.texts = data['texts'].tolist()
            self.labels = data['labels'].tolist()
        else:
            self.texts = []
            self.labels = []

    def save_training_data(self):
        np.savez(self.DATA_FILE, texts=self.texts, labels=self.labels)

    def train(self, spam_texts: List[str], non_spam_texts: List[str]):
        new_texts = spam_texts + non_spam_texts
        new_labels = [1] * len(spam_texts) + [0] * len(non_spam_texts)

        if not new_texts:
            raise ValueError("Training data cannot be empty")

        self.texts.extend(new_texts)
        self.labels.extend(new_labels)
        self.save_training_data()

        X_all = self.vectorizer.fit_transform(self.texts)
        y_all = np.array(self.labels)
        joblib.dump(self.vectorizer, self.VECTOR_FILE)

        self.classifier.partial_fit(X_all, y_all, classes=np.array([0, 1]))
        joblib.dump(self.classifier, self.MODEL_FILE)

    def predict(self, text: str) -> bool:
        X = self.vectorizer.transform([text])
        return self.classifier.predict(X)[0]

    def test(self, spam_texts: List[str], non_spam_texts: List[str]) -> float:
        texts = spam_texts + non_spam_texts
        labels = [1] * len(spam_texts) + [0] * len(non_spam_texts)

        if not texts:
            raise ValueError("Test data cannot be empty")

        X = self.vectorizer.transform(texts)
        return self.classifier.score(X, labels)

    def save_all(self):
        joblib.dump(self.vectorizer, self.VECTOR_FILE)
        joblib.dump(self.classifier, self.MODEL_FILE)
        self.save_training_data()
```

[34]:
```
train_df = pd.read_csv('/content/drive/MyDrive/spam_text_train_dataset.csv')
test_df = pd.read_csv('/content/drive/MyDrive/spam_text_test_dataset.csv')

train_df = train_df.dropna(subset=["text"])
test_df = test_df.dropna(subset=["text"])

print(len(train_df))
print(len(test_df))

num_epochs = 3
batch_size = 1000

spam_train_texts = train_df[train_df["label"] == 1]["text"].tolist()
non_spam_train_texts = train_df[train_df["label"] == 0]["text"].tolist()

spam_test_texts = test_df[test_df["label"] == 1]["text"].tolist()
non_spam_test_texts = test_df[test_df["label"] == 0]["text"].tolist()
```

```
97565
24396
```

[35]:
```
spam_detector = SpamDetectionModel()

all_train_texts = spam_train_texts + non_spam_train_texts

X_all = spam_detector.vectorizer.fit_transform(all_train_texts)

for epoch in range(num_epochs):
  for i in range(0, len(spam_train_texts), batch_size):
      spam_batch = spam_train_texts[i:i+batch_size]
      non_spam_batch = non_spam_train_texts[i:i+batch_size]

      X_batch = spam_detector.vectorizer.transform(spam_batch + non_spam_batch)

      y_batch = [1] * len(spam_batch) + [0] * len(non_spam_batch)
      spam_detector.classifier.partial_fit(X_batch, y_batch, classes=np.
 ↪array([0, 1]))

spam_detector.save_all()
```

```
print("Model is trained")
```

```
Model is trained
```

[36]:
```
X_test = spam_detector.vectorizer.transform(spam_test_texts +␣
 ↪non_spam_test_texts)
y_test = [1] * len(spam_test_texts) + [0] * len(non_spam_test_texts)
accuracy = spam_detector.classifier.score(X_test, y_test)

print(f"Accuracy: {accuracy*100:.2f}%")
```

```
Accuracy: 89.45%
```