Sentiment analysis

Thursday, 23 January 2025 6:28 PM

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

# Function to read CSV files
def read_csv(path):
    return pd.read_csv("/kaggle/input/contradictory-my-dear-watson/" + path)

# Load training and test datasets
train_data = read_csv("train.csv")
test_data = read_csv("test.csv")

# Extract features and labels
X_train = train_data["premise"]  # Premises (text data)
y_train = train_data["label"]    # Sentiment labels (0, 1, or 2)
X_test = test_data["premise"]    # Premises in the test set

# TF-IDF Vectorization
vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = vectorizer.fit_transform(X_train)  # Fit and transform train data
X_test_tfidf = vectorizer.transform(X_test)       # Transform test data

# Logistic Regression Model
clf = LogisticRegression(random_state=42, max_iter=1000)
clf.fit(X_train_tfidf, y_train)

# Predict on test data
y_test_pred = clf.predict(X_test_tfidf)

# Save predictions to CSV for submission
output = pd.DataFrame({'id': test_data["id"], 'prediction': y_test_pred})
output.to_csv('submission.csv', index=False)
print("Your submission was successfully saved!")
```

==========>><<==========

Advanced considering the language and other features and stacking them..

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import OneHotEncoder
from scipy.sparse import hstack
```

```
# Function to read CSV files
def read_csv(path):
   return pd.read_csv("/kaggle/input/contradictory-my-dear-watson/" + path)

# Load training and test datasets
train_data = read_csv("train.csv")
test_data = read_csv("test.csv")

# Combine 'premise' and 'hypothesis' into a single feature
train_data["combined_text"] = train_data["premise"] + " " + train_data["hypothesis"]
test_data["combined_text"] = test_data["premise"] + " " + test_data["hypothesis"]

# Extract text features and language
X_train_text = train_data["combined_text"]
X_test_text = test_data["combined_text"]
X_train_language = train_data["language"].values.reshape(-1, 1)
X_test_language = test_data["language"].values.reshape(-1, 1)

# TF-IDF Vectorization for combined text
vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = vectorizer.fit_transform(X_train_text)
X_test_tfidf = vectorizer.transform(X_test_text)

# One-Hot Encoding for language
encoder = OneHotEncoder(sparse=True, handle_unknown="ignore")
X_train_language_encoded = encoder.fit_transform(X_train_language)
X_test_language_encoded = encoder.transform(X_test_language)

# Combine TF-IDF features and one-hot encoded language features
X_train_combined = hstack([X_train_tfidf, X_train_language_encoded])
X_test_combined = hstack([X_test_tfidf, X_test_language_encoded])

# Labels
y_train = train_data["label"]

# Logistic Regression Model
clf = LogisticRegression(random_state=42, max_iter=1000)
clf.fit(X_train_combined, y_train)

# Predict on test data
y_test_pred = clf.predict(X_test_combined)

# Save predictions to CSV for submission
output = pd.DataFrame({'id': test_data["id"], 'prediction': y_test_pred})
output.to_csv('submission.csv', index=False)
print("Your submission was successfully saved!")
```