Arifun Nabi

Prof. William Farmer

INST 414

Memo

May 14th, 2023

Memo:

Preprocessing: For preprocessing, I used regex to remove numerical values. I also converted the text into lowercase and used NLTK library for tokenization. Lemmatization was used to reduce words to their root form. Stop words were removed using NLTK library.

Features: The TfidfVectorizer was used to convert the preprocessed text into a matrix of TF-IDF features. Max feature was set to 10000 to limit the number of features and ngram_range was set to (1,2).

Models: LinearSVC and Logistic Regression were tested, and Logistic Regression model performed the most.

Parameter Tuning: GridSearchCV was used for hyperparameter tuning on the models and vect__max_df and clf__C parameter was tuned.

Result: Logistic Regression model had the highest accuracy score of 0.876

Source Code for Logistic Regression:

```
import pandas as pd
import os
from google.colab import files
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.dummy import DummyClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from nltk.stem import WordNetLemmatizer
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import PolynomialFeatures, MinMaxScaler
import nltk
from nltk.corpus import stopwords
import re
from sklearn.feature_extraction.text import TfidfVectorizer
train = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/train.csv")
from google.colab import drive
drive.mount('/content/drive')
nltk.download('stopwords')
nltk.download('wordnet')
stop_words = set (stopwords.words("english"))
test = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/test.csv")
lemmatizer = WordNetLemmatizer()
X_train, X_test, y_train, y_test = train_test_split(train.review,
train.label, test_size=0.2, random_state=42)
def preprocess_text(text):
      text = re.sub(r"[^a-zA-Z0-9]", " ", text)
      text = text.lower()
      words = nltk.word_tokenize(text)
      words = ' '.join([lemmatizer.lemmatize(w) for w in words])
      return text
nltk.download('punkt')
X_train = X_train.apply(preprocess_text)
pipeline = Pipeline([
    ('vect', TfidfVectorizer(max_features=10000, ngram_range=(1, 2))),
    ('clf', LogisticRegression())
])
parameter = {
    'vect__max_df': [0.5, 0.75, 1.0],
    'clf__C': [0.1, 1, 10]
```

```
}
grid_search = GridSearchCV(pipeline, param_grid = parameter, cv=5)
grid_search.fit(X_train, y_train)
X_test = X_test.apply(preprocess_text)
lr_prediction = grid_search.predict(X_test)
lr_accuracy = accuracy_score(y_test, lr_prediction)
test["review"] = test["review"].apply(preprocess_text)
lr_prediction = grid_search.predict(test.review)
prediction_df = pd.DataFrame({"Id": test.Id, "Category": lr_prediction})
prediction_df.to_csv("lr_prediction.csv", index=False)
files.download("lr_prediction.csv")
```

Source Code for LinearSVC:

```
import pandas as pd
import os
from google.colab import files
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.dummy import DummyClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from nltk.stem import WordNetLemmatizer
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import PolynomialFeatures, MinMaxScaler
import nltk
from nltk.corpus import stopwords
import re
from sklearn.feature_extraction.text import TfidfVectorizer
train = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/train.csv")
from google.colab import drive
drive.mount('/content/drive')
nltk.download('stopwords')
nltk.download('wordnet')
stop_words = set (stopwords.words("english"))
test = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/test.csv")
lemmatizer = WordNetLemmatizer()
X_train, X_test, y_train, y_test = train_test_split(train.review,
train.label, test_size=0.2, random_state=42)
```

```
def preprocess_text(text):
      text = re.sub(r"[^a-zA-Z0-9]", " ", text)
      text = text.lower()
      words = nltk.word_tokenize(text)
      words = ' '.join([lemmatizer.lemmatize(w) for w in words])
      return text
nltk.download('punkt')
X_train = X_train.apply(preprocess_text)
pipeline = Pipeline([
    ('vect', TfidfVectorizer(max_features=10000, ngram_range=(1, 2))),
    ('clf', LinearSVC())
])
parameter = {
    'vect__max_df': [0.5, 0.75, 1.0],
    'clf__C': [0.1, 1, 10]
}
grid_search = GridSearchCV(pipeline, param_grid = parameter, cv=5)
grid_search.fit(train.review, train.label)
grid_search.best_score_
svm_prediction = grid_search.predict(test.review)
prediction_df = pd.DataFrame({"Id": test.Id, "Category": svm_prediction})
prediction_df.to_csv("svm_prediction.csv", index=False)
files.download("svm_prediction.csv")
```