IDM:Assignment

News Classification based on Their Head lines

ChanpisethChap

Royal University of PhnomPenh

Group: MISA Pisatto, Hok Lenghak, Yorn Chanvisal

1. Data Collection

1.1 Scraping News Articles

Certainly! Below is the additional section for data collection in the report:

1. Data Collection

1.1 Scraping News Articles

In the initial phase of the data collection process, the script employs Selenium and the Chrome WebDriver to systematically scrape news articles from the CNN website. The retrieved data is subsequently stored in a CSV file named 'data_set.csv'. To ensure data integrity, the script first checks for the existence of the CSV file; if absent, it creates the file with a header. The Chrome WebDriver is configured to navigate to a specific webpage, namely, 'https://www.nbcnews.com/archive/articles/2023/december', where news articles are archived. Following this, the script identifies the main HTML tag ('MonthPage') and extracts relevant information from anchor tags ('a') assumed to contain article titles. Each title, along with its corresponding ID and URL, is appended to a list for further processing.

Subsequently, the script iterates through the collected titles, visiting each article's URL to extract additional details such as category, timestamp, author, and hostname. To maintain transparency and diagnose potential issues, the script prints the collected data and logs it into a file named 'logs.log'. Any encountered errors during the scraping process are caught, printed, and logged, ensuring a comprehensive record of the data collection process. Finally, after completing the scraping and data appending procedures, the Chrome WebDriver is gracefully closed, and a confirmation message is displayed, indicating the successful addition of data to the 'data_set.csv' file.

We eexecuted this process a total of 12 times, spanning a duration of 12 months, with the objective of accumulating an extensive dataset of over 10,000 rows of news articles.

After completing the aforementioned process, the resulting dataset includes the following entry:

```
'ID', 'TITLE', 'URL', 'Source', 'CATEGORY', 'Key', 'Website', 'Timestamp'
1, "Michael Latt, Hollywood social justice advocate, fatally shot in his home by an intruder,
police say", https://www.nbcnews.com/news/us-news/hollywood-social-justice-advocate-killed-home-
intruder-rcna127406, U.S. NEWS, 2023-12-01T00:25:52.224Z, By Natalie Kainz and Phil Helsel,
www.nbcnews.com
```

This entry signifies a news article titled "Michael Latt, Hollywood social justice advocate, fatally shot in his home by an intruder, police say," categorized under U.S. NEWS. The article was published on December 1, 2023, at 00:25:52.224 UTC. It was authored by Natalie Kainz and Phil Helsel and is accessible via the URL: https://www.nbcnews.com/news/us-news/hollywood-social-justice-advocate-killed-home-intruder-rcna127406.

In [ ]:
```python
# Import dependency
import logging
import csv

from selenium import webdriver
from selenium.webdriver.common.by import By

# Declare and use logger incase app crashes scv won't be created
logging.basicConfig(filename='logs.log', level=logging.INFO, format='%(asctime)s [%(levelname)s]: %(message)s')

# CSV file path
csv_file_path = 'data_set.csv'

# Check if the file exists, if not create it with header
file_exists = False
try:
    with open(csv_file_path, 'r') as file:
        reader = csv.reader(file)
        if any(reader):
            file_exists = True
except FileNotFoundError:
```

```
    pass


# Set up the web driver (you need to have the appropriate web driver executable installed)
driver = webdriver.Chrome()

# Navigate to the NBC news 2023 december archive
driver.get("https://www.nbcnews.com/archive/articles/2023/december")

# Get all the titles from the page
main_tag = driver.find_element(By.CLASS_NAME, 'MonthPage')
title_tags = main_tag.find_elements(By.TAG_NAME, 'a')
titles = []
index = 0

# Loop through the titles and append them to the titles list
for title in title_tags:
    index += 1
    title = {"id": index, "title": title.text, "url": title.get_attribute('href')}
    titles.append(title)

# Append data to the CSV file
with open(csv_file_path, 'a', newline='') as file:

    # Loop through the titles and get the category, timestamp, author and hostname
    for title in titles:
        try:
            driver.get(title.get('url'))
            span_tag = driver.find_element(By.CSS_SELECTOR,  'span[data-testid="unibrow-text"]')
            data_tag = driver.find_element(By.CSS_SELECTOR,  'time[data-testid="timestamp__datePublished"]')
            div_tag = driver.find_element(By.CSS_SELECTOR,  'div[data-activity-map="inline-byline-article-top"]')
            title['category'] = span_tag.text
            title['timestamp'] = data_tag.get_attribute('content')
            title['author'] = div_tag.text
            title['hostname'] = 'www.nbcnews.com'

            fieldnames = title.keys()
            logging.info(f"Appended to {csv_file_path}: {title}")
            writer = csv.DictWriter(file, fieldnames=fieldnames)
            writer.writerow(title)
        except:
            continue

driver.close()

print(f"Data has been appended to {csv_file_path}.")
```

2. Data Preprocessing

2.1 Loading Data

- The script then loads the collected data from the CSV file using the pandas library.

In [ ]:
```
import pandas as pd

# Data set file path
src_file = 'all.csv'

# Read the CSV file
dataframe = pd.read_csv(src_file, quotechar="\"", engine='python', usecols=["TITLE", "CATEGORY"])
dataframe.shape
```

Out[ ]:
```
(11861, 2)
```

2.2 Data Filtering

- The dataset is filtered based on the number of occurrences of each category.
- Categories with counts less than or equal to 10 are excluded from the dataset.

In [ ]:
```
# Reduce the data set to only the categories with more than 10 data points
category_count = dataframe.groupby("CATEGORY").size().reset_index(name='COUNT')
category_count = category_count[category_count["COUNT"] > 10]
dataframe = dataframe[dataframe["CATEGORY"].isin(category_count["CATEGORY"])]


# Remove the categories that are not needed
dataframe = dataframe[~dataframe["CATEGORY"].isin(["U.S. NEWS", "NEWS", "WORLD"])]


print(dataframe.groupby("CATEGORY").size().reset_index(name='COUNT'))
print(dataframe.head())
dataframe.shape
```

```
              CATEGORY  COUNT
0     #METOO RECKONING     12
1        2022 ELECTION     81
2        2024 ELECTION    165
3      ABORTION RIGHTS    123
4   AFTER GEORGE FLOYD     16
..                 ...    ...
87         U.K. ROYALS     28
88      WAR IN UKRAINE    509
89            WEATHER     55
90   WESTERN WILDFIRES     19
91         WHITE HOUSE    202

[92 rows x 2 columns]
                                                TITLE         CATEGORY
0  A college professor called the police on two s...  CULTURE MATTERS
1  Oscars producer says police were prepared to a...        CELEBRITY
2  Jared Kushner interviewed by Jan. 6 committee ...     DONALD TRUMP
3  House passes bill to cap out-of-pocket insulin...         CONGRESS
5  Senate negotiators reach 'agreement in princip...         CONGRESS
```

Out[ ]: (7698, 2)

2.3 Handling Missing Data

- The script checks for missing data and reports if any are found.

In [ ]:
```python
#check for missing data
if(any(dataframe.isnull().any())):
    print('Missing Data\n')
    print(dataframe.isnull().sum())
else:
    print('NO missing data')
```

```
NO missing data
```

2.4 Handling Duplicate Data

- Duplicate rows are identified and removed from the dataset.

In [ ]:
```python
# check for duplicate
if(any(dataframe.duplicated())==True):
    print('Duplicate rows found')
    print('Number of duplicate rows= ', dataframe[dataframe.duplicated()].shape[0])
    dataframe.drop_duplicates(inplace=True,keep='first')
    dataframe.reset_index(inplace=True,drop=True)
    print('Dropping duplicates\n')
    print(dataframe.shape)
else:
    print('NO duplicate data')
```

```
Duplicate rows found
Number of duplicate rows=  40
Dropping duplicates

(7658, 2)
```

2.5 Text Cleaning and Tokenization

- NLTK libraries are used to download necessary resources (stopwords, punkt, wordnet).
- A custom tokenizer function is defined to lowercase text, remove digits, punctuation, and stopwords, and lemmatize the remaining tokens.
- A pipeline is created to perform Count Vectorization and TF-IDF transformation on the 'TITLE' column of the dataset.

In [ ]:
```python
import nltk
import re
import string

from sklearn import set_config
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from nltk import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')


# Set the sklearn pipeline to return pandas dataframe
set_config(transform_output="pandas")
```

```
wnl = WordNetLemmatizer()

# Function for cleaning and tokenize the headline
def tokenize(doc):
  document = doc.lower() # convert the content of the headline to lowercase
  document = re.sub(r'\d+', '', document) # remove all of the digits inside of the content (using regular expressio
  document = document.translate(str.maketrans('', '', string.punctuation)) # remove the puntuations (, . ! # ...)
  document = document.strip() # remove the spaces at the start and end of the headline
  return [wnl.lemmatize(token) for token in word_tokenize(document) if token not in stopwords.words('english')]
  # tokenize the headlines
  # and then filter only the words that are not in the english stopwords (words that are commonly used and give no
  # and finally lemmatize all of the tokens

# The preprocess pipeline
preprocessor = Pipeline([
    ('vect', CountVectorizer(tokenizer = tokenize)), # passing custom tokenizer method for the CountVectorizer to u
    ('tfidf', TfidfTransformer()),
])

tfidf_dataset = preprocessor.fit_transform(dataframe["TITLE"].values) # process the training dataset
```

3. Training Models

3.1 Decision Tree Classifier

3.1.1 Splitting the Dataset

- The dataset is split into training and testing sets.

In [ ]:
```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

# Encode the category labels
le = LabelEncoder()
class_label = le.fit_transform(dataframe["CATEGORY"])

# Split the data set into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    tfidf_dataset.toarray(),
    class_label,
    test_size = 0.3 # the size of the testing dataset (in percentage between 0 and 1)
)
```

3.1.2 Decision Tree Classifier

- A Decision Tree Classifier is trained on the TF-IDF transformed data.
- The accuracy of the model is evaluated on the testing set.
- An example prediction is demonstrated using a sample input.

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import numpy as np

#Decision Tree
DTClass = DecisionTreeClassifier(criterion="gini", splitter="best", random_state=40)
DTClass.fit(X_train, y_train)
y_pred = DTClass.predict(X_test)


print("accuracy score of Decision Tree:")
print(accuracy_score(y_test, y_pred))

#Predicting the category of a new headline
predicted = DTClass.predict(preprocessor.transform(["Slashing Central American aid could drive more migrants to the
print(o)
print(dataframe["CATEGORY"].values[np.where(class_label == predicted)[0]][0])
```

```
[41]
IMMIGRATION
accuracy score of Decision Tree:
0.4225413402959095
```

3.2 Multinomial Naive Bayes Classifier

3.2.1 Splitting the Dataset

- The dataset is split into training and testing sets.

In [ ]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

naive_bayes_model = MultinomialNB()
```

```
vectorizer = CountVectorizer()

X_train, X_test, y_train, y_test = train_test_split(dataframe['TITLE'], dataframe['CATEGORY'], test_size=0.1, rando
```

3.2.2 Naive Bayes Classifier

- A Multinomial Naive Bayes Classifier is trained on the Count Vectorized data.
- The accuracy of the model is evaluated on the testing set.
- An example prediction is demonstrated using a sample input.

In [ ]:
```
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)

naive_bayes_model.fit(X_train_vectorized, y_train)

y_pred = naive_bayes_model.predict(X_test_vectorized)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print("accuracy score of Naive Bayes:")

new_title_vectorized = vectorizer.transform(["At least five states are considering requiring full minimum wages for

predicted_category = naive_bayes_model.predict(new_title_vectorized)

print(f'Predicted Category: {predicted_category[0]}')
```

```
accuracy score of Naive Bayes:
Predicted Category: CULTURE MATTERS
```

3.3 Artificial Neural Network

3.3.1 Splitting the Dataset

- The dataset is split into training and testing sets.

3.3.2 Neural Network Training

- An Artificial Neural Network (ANN) is configured and trained on the dataset.
- The accuracy of the model is evaluated on the testing set.
- An example prediction is demonstrated using a sample input.

4. Conclusion

- The script successfully collects and preprocesses the text data, encodes labels, and trains three classification models: Decision Tree, Naive Bayes, and Artificial Neural Network.
- The accuracy scores and classification reports provide insights into the performance of each model on the testing set.