

Kishore0420G /
FakenewsdetectionusingNLP-phase-5

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

FakenewsdetectionusingNLP-phase-5 / main.py



Kishore0420G Add files via upload

3 minutes ago



222 lines (178 loc) · 13.5 KB

Code

Blame

Raw



```
1  # %% [markdown]
2  # *Fake News Detection*
3  #
4  # It has become humanly impossible to identify fake news on the online portals across the globe.The sheer v
5  #
6  # The most crucial thing here is data which has been already available in the kaggle. We will be using diff
7
8  # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:19:57.550035Z","iopub.execute_input":"2023-10-
9  # This Python 3 environment comes with many helpful analytics libraries installed
10 # It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
11 # For example, here's several helpful packages to load
12
13 import numpy as np # linear algebra
14 import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
15
16 # Input data files are available in the read-only "../input/" directory
17 # For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input
18
19 import os
20 for dirname, _, filenames in os.walk('/kaggle/input'):
21     for filename in filenames:
22         print(os.path.join(dirname, filename))
23
24 # You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when y
25 # You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current sessi
26
27 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:19:57.573835Z","iopub.execute_input":"2023-10-
28 !pip install gensim # Gensim is an open-source library for unsupervised topic modeling and natural language
29 import nltk
30 nltk.download('punkt')
31
32 import pandas as pd
33 import numpy as np
34 import matplotlib.pyplot as plt
35 import seaborn as sns
36 from wordcloud import WordCloud, STOPWORDS
37 import nltk
38 import re
39 from nltk.corpus import stopwords
40 import seaborn as sns
41 import gensim
42 from gensim.utils import simple_preprocess
43 from gensim.parsing.preprocessing import STOPWORDS
44
45 import plotly.express as px
```

```

46 from sklearn.model_selection import train_test_split
47 from sklearn.feature_extraction.text import CountVectorizer
48 from sklearn.linear_model import LogisticRegression
49 from sklearn.metrics import roc_auc_score
50 from sklearn.metrics import confusion_matrix
51
52 # %% [markdown]
53 # *Import the data & Clean ups*
54
55 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:12.47781Z","iopub.execute_input":"2023-10-2
56 #importing data
57 fake_data = pd.read_csv('/kaggle/input/fake-and-real-news-dataset/Fake.csv')
58 print("fake_data",fake_data.shape)
59
60 true_data= pd.read_csv('/kaggle/input/fake-and-real-news-dataset/True.csv')
61 print("true_data",true_data.shape)
62
63 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:14.043858Z","iopub.execute_input":"2023-10-
64 fake_data.head(5)
65
66 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:14.066765Z","iopub.execute_input":"2023-10-
67 true_data.head(5)
68
69 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:14.0866Z","iopub.execute_input":"2023-10-24
70 #adding additonal column to seperate between true & fake data
71 # true =1, fake =0
72 true_data['target'] = 1
73 fake_data['target'] = 0
74 df = pd.concat([true_data, fake_data]).reset_index(drop = True)
75 df['original'] = df['title'] + ' ' + df['text']
76 df.head()
77
78 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:14.737583Z","iopub.execute_input":"2023-10-
79 df.isnull().sum()
80
81 # %% [markdown]
82 # *Data Clean up*
83 # - create a function here that will be responsible to remove any unnecesary words (Stopwords) from the da
84
85 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:14.781215Z","iopub.execute_input":"2023-10-
86 stop_words = stopwords.words('english')
87 stop_words.extend(['from', 'subject', 're', 'edu', 'use'])
88 ✓ def preprocess(text):
89     result = []
90     for token in gensim.utils.simple_preprocess(text):
91         if token not in gensim.parsing.preprocessing.STOPWORDS and len(token) > 2 and token not in stop_wor
92             result.append(token)
93
94     return result
95
96 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:14.79379Z","iopub.execute_input":"2023-10-2
97 # Transforming the unmatched subjects to the same notation
98 df.subject=df.subject.replace({'politics':'PoliticsNews','politicsNews':'PoliticsNews'})
99
100 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:14.819181Z","iopub.execute_input":"2023-10-
101 sub_tf_df=df.groupby('target').apply(lambda x:x['title'].count()).reset_index(name='Counts')
102 sub_tf_df.target.replace({0:'False',1:'True'},inplace=True)
103 fig = px.bar(sub_tf_df, x="target", y="Counts",
104               color='Counts', barmode='group',
105               height=350)
106 fig.show()
107

```

```

108 # %% [markdown]
109 # - The data looks balanced and no issues on building the model
110
111 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:14.938929Z","iopub.execute_input":"2023-10-
112 sub_check=df.groupby('subject').apply(lambda x:x['title'].count()).reset_index(name='Counts')
113 fig=px.bar(sub_check,x='subject',y='Counts',color='Counts',title='Count of News Articles by Subject')
114 fig.show()
115
116 # %% [markdown]
117 #
118
119 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:15.056981Z","iopub.execute_input":"2023-10-
120 df['clean_title'] = df['title'].apply(preprocess)
121 df['clean_title'][0]
122
123 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:18.393978Z","iopub.execute_input":"2023-10-
124 df['clean_joined_title']=df['clean_title'].apply(lambda x:" ".join(x))
125
126 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:18.444948Z","iopub.execute_input":"2023-10-
127 plt.figure(figsize = (20,20))
128 wc = WordCloud(max_words = 2000 , width = 1600 , height = 800 , stopwords = stop_words).generate(" ".join(c
129 plt.imshow(wc, interpolation = 'bilinear')
130
131 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:40.057751Z","iopub.execute_input":"2023-10-
132 maxlen = -1
133 for doc in df.clean_joined_title:
134     tokens = nltk.word_tokenize(doc)
135     if(maxlen<len(tokens)):
136         maxlen = len(tokens)
137 print("The maximum number of words in a title is =", maxlen)
138 fig = px.histogram(x = [len(nltk.word_tokenize(x)) for x in df.clean_joined_title], nbins = 50)
139 fig.show()
140
141 # %% [markdown]
142 # *Creating Prediction Model*
143
144 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:57.606819Z","iopub.execute_input":"2023-10-
145 X_train, X_test, y_train, y_test = train_test_split(df.clean_joined_title, df.target, test_size = 0.2,rand
146 vec_train = CountVectorizer().fit(X_train)
147 X_vec_train = vec_train.transform(X_train)
148 X_vec_test = vec_train.transform(X_test)
149
150 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:59.134487Z","iopub.execute_input":"2023-10-
151 #model
152 model = LogisticRegression(C=2)
153
154 #fit the model
155 model.fit(X_vec_train, y_train)
156 predicted_value = model.predict(X_vec_test)
157
158 #accuracy & predicted value
159 accuracy_value = roc_auc_score(y_test, predicted_value)
160 print(accuracy_value)
161
162 # %% [markdown]
163 # *Create the confusion matrix*
164
165 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:21:01.621433Z","iopub.execute_input":"2023-10-
166 cm = confusion_matrix(list(y_test), predicted_value)
167 plt.figure(figsize = (7, 7))
168 sns.heatmap(cm, annot = True,fmt='g',cmap='viridis')
169

```

```

170 # %% [markdown]
171 # - 4465 Fake News have been Classified as Fake
172 # - 4045 Real News have been classified as Real
173
174 # %% [markdown]
175 # *Checking the content of news*
176
177 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:21:02.055182Z","iopub.execute_input":"2023-10-
178 df['clean_text'] = df['text'].apply(preprocess)
179 df['clean_joined_text']=df['clean_text'].apply(lambda x: " ".join(x))
180
181 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:22:28.723648Z","iopub.execute_input":"2023-10-
182 plt.figure(figsize = (20,20))
183 wc = WordCloud(max_words = 2000 , width = 1600 , height = 800 , stopwords = stop_words).generate(" ".join(c
184 plt.imshow(wc, interpolation = 'bilinear')
185
186 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:23:30.622781Z","iopub.execute_input":"2023-10-
187 maxlen = -1
188 for doc in df.clean_joined_text:
189     tokens = nltk.word_tokenize(doc)
190     if(maxlen<len(tokens)):
191         maxlen = len(tokens)
192 print("The maximum number of words in a News Content is =", maxlen)
193 fig = px.histogram(x = [len(nltk.word_tokenize(x)) for x in df.clean_joined_text], nbins = 50)
194 fig.show()
195
196
197 # %% [markdown]
198 # *Predicting the Model*
199
200 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:27:23.232691Z","iopub.execute_input":"2023-10-
201 X_train, X_test, y_train, y_test = train_test_split(df.clean_joined_text, df.target, test_size = 0.2,rand
202 vec_train = CountVectorizer().fit(X_train)
203 X_vec_train = vec_train.transform(X_train)
204 X_vec_test = vec_train.transform(X_test)
205 model = LogisticRegression(C=2.5)
206 model.fit(X_vec_train, y_train)
207 predicted_value = model.predict(X_vec_test)
208 accuracy_value = roc_auc_score(y_test, predicted_value)
209 print(accuracy_value)
210
211 # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:30:15.444444Z","iopub.execute_input":"2023-10-
212 prediction = []
213 for i in range(len(predicted_value)):
214     if predicted_value[i].item() > 0.5:
215         prediction.append(1)
216     else:
217         prediction.append(0)
218 cm = confusion_matrix(list(y_test), prediction)
219 plt.figure(figsize = (6, 6))
220 sns.heatmap(cm, annot = True,fmt='g')
221
222 # %% [code]

```