Task4_simple

March 21, 2025

1 Find top 10000 vocab + Logistic Rgression

Training

[1]:
```python
import re
import numpy as np
import pandas as pd
import nltk
from collections import Counter
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
import requests
import ast

# Download nltk packages if necessary
# nltk.download('punkt')
# nltk.download('stopwords')
```

[2]:
```python
# ---- 1. Load and prepare data ----

# Mapping of types to "fake" or "reliable"
type_mapping = {
    'unreliable': 'fake',
    'fake': 'fake',
    'conspiracy': 'fake',
    'bias': 'fake',
    'junksci': 'fake',
    'clickbait': 'reliable',
    'reliable' : 'reliable',
    'state': 'fake',
    'political': 'reliable',
    'satire': 'fake',
    'hate': 'fake',
    'rumor': 'fake',
}

# Load data and filter relevant columns
data = pd.read_csv("15,000_rows_preprocessed.csv", usecols=["content", "type"],␣
 ↪dtype=str)
#data = pd.read_csv("995,000_rows_preprocessed.csv", usecols=["content",␣
 ↪"type"], dtype=str)

# Load BBC-data and add to dataset
bbc_data = pd.read_csv("BBC_preprocessed.csv", usecols=["content", "type"],␣
 ↪dtype=str)
data = pd.concat([data, bbc_data], ignore_index=True)

# Remove rows with unknown type
data = data[data['type'] != 'unknown']

# Map types to labels
data["label"] = data["type"].map(type_mapping)

# Remove NaN
data = data.dropna(subset=["label"])

# ---- 2. Split dataset to training, validation and test (80/10/10) ----

train, valid, test = np.split(
    data.sample(frac=1, random_state=42),  # Shuffle
    [int(0.8 * len(data)), int(0.9 * len(data))]  # Index for split
)
```

```
/opt/anaconda3/lib/python3.12/site-packages/numpy/core/fromnumeric.py:59:
FutureWarning: 'DataFrame.swapaxes' is deprecated and will be removed in a
future version. Please use 'DataFrame.transpose' instead.
  return bound(*args, **kwds)
```

Load “LIAR” dataset

[3]:

```
# Load evaluation data from LIAR-dataset

liar_test_data = pd.read_csv("liar_test_data_preprocessed.csv",␣
 ↪usecols=['type', 'content'], dtype=str)

type_mapping_liar = {
    'true': 'reliable',
    'false': 'fake',
    'half-true': 'fake',
    'pants-fire': 'fake',
    'barely-true': 'reliable',
    'mostly-true': 'reliable'
}

liar_test_data["label"] = liar_test_data["type"].map(type_mapping_liar)
```

2 Logistic Regression

[4]:

```
from sklearn.preprocessing import MaxAbsScaler
import re
import numpy as np
import pandas as pd
import nltk
from collections import Counter
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# ---- 3. Convert text to Bag-of-Words feature matrix ----

content_as_lists = data['content'].apply(ast.literal_eval)

all_words = content_as_lists.explode().tolist()

word_counts = Counter(all_words)

top_10000_words = [word for word, _ in word_counts.most_common(10000)]

vectorizer = CountVectorizer(vocabulary=top_10000_words)
```

[5]:

```
# Transform training, valid og test data
X_train = vectorizer.transform(train['content'])
X_valid = vectorizer.transform(valid['content'])
X_test = vectorizer.transform(test['content'])

# Transform LIAR test data to Bag-of-Words
X_test_liar = vectorizer.transform(liar_test_data['content'])
```

[6]:

```
scaler = MaxAbsScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_valid_scaled = scaler.transform(X_valid)
X_test_scaled = scaler.transform(X_test)
# LIAR not scaled
```

```python
# Labels
y_train = train['label']
y_valid = valid['label']
y_test = test['label']

y_test_liar = liar_test_data['label']

# ---- 4. Train Logistic Regression model ----

clf = LogisticRegression(max_iter=100000, solver='saga', random_state=42)
clf.fit(X_train_scaled, y_train)

# ---- 5. Evaluate model ----

y_pred_liar = clf.predict(X_test_liar)

# ---- 6. Evaluate model on LIAR test. Notice that it isn't scaled since␣
 ↪scaling gave an even worse result ----

print(classification_report(y_test_liar, y_pred_liar))

from sklearn.metrics import confusion_matrix

print(confusion_matrix(y_test_liar, y_pred_liar))

# ---- DEBUGGING ----

print(X_train.shape, X_test_liar.shape)

print(set(y_train), set(y_test_liar))

from collections import Counter
print(Counter(y_train))
print(Counter(y_test_liar))

# ---- Original test with test set ----
print(classification_report(y_test, clf.predict(X_test)))
```

```
              precision    recall  f1-score   support

        fake       0.48      0.32      0.38       606
    reliable       0.52      0.69      0.59       661

    accuracy                           0.51      1267
   macro avg       0.50      0.50      0.49      1267
weighted avg       0.50      0.51      0.49      1267
```

```
[[191 415]
 [207 454]]
(10959, 10000) (1267, 10000)
{'fake', 'reliable'} {'fake', 'reliable'}
Counter({'reliable': 5574, 'fake': 5385})
Counter({'reliable': 661, 'fake': 606})
              precision    recall  f1-score   support

        fake       0.83      0.75      0.79       676
    reliable       0.78      0.85      0.81       694

    accuracy                           0.80      1370
   macro avg       0.80      0.80      0.80      1370
weighted avg       0.80      0.80      0.80      1370
```