Main

April 29, 2023

[1]:
```
#
import pandas as pd
import numpy as np
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from collections import Counter
```

[2]:
```
#           ,
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /home/artem627/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/artem627/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[2]:
```
True
```

[3]:
```
#
max_words = 10000
random_state = 42
```

[4]:
```
#
train_data = pd.read_csv('dataset.csv', sep=',', index_col='idx');
```

[5]:
```
#
def preprocess(text, stop_words, punctuation_marks):
    tokens = word_tokenize(text.lower())
    preprocessed_text = []
    for token in tokens:
        if token not in punctuation_marks:
            if token not in stop_words:
                preprocessed_text.append(token)
    return preprocessed_text
```

[6]:
```
#
punctuation_marks = ['!', ',', '(', ')', ':', '-', '?', '.', '..', '...', '«',␣
 ↪'»', ';', '–', '--']

#
stop_words = stopwords.words("english")
stop_words.append("'s")
stop_words.append("send")
stop_words.append("please")
stop_words.append("come")
stop_words.append("help")
stop_words.append("n't")
stop_words.append("saw")
```

[7]:
```
#
train_data['Preprocessed_texts'] = train_data.apply(lambda row:␣
 ↪preprocess(row['Text'], punctuation_marks, stop_words), axis=1)
```

[8]:
```
#
words = Counter()
for txt in train_data['Preprocessed_texts']:
    words.update(txt)
```

[9]:
```
#     ,
word_to_index = dict()
#     ,
index_to_word = dict()
```

[10]:
```
#
for i, word in enumerate(words.most_common(max_words - 2)):
    word_to_index[word[0]] = i + 2
    index_to_word[i + 2] = word[0]
```

[11]:
```
#
def text_to_sequence(txt, word_to_index):
    seq = []
    for word in txt:
        index = word_to_index.get(word, 1)
        if index != 1:
            seq.append(index)
    return seq
```

[12]:
```
#
train_data['Sequences'] = train_data.apply(lambda row:␣
 ↪text_to_sequence(row['Preprocessed_texts'], word_to_index), axis=1)
```

[13]:
```
#
mapping = {
    'fire': 0,
    'medical': 1,
    'car_accident': 2,
    'natural_disasters': 3,
    'lost_man': 4,
    'airport_emergencies': 5,
    'violence': 6,
    'animals': 7,
}
```

[14]:
```
#
train, test = train_test_split(train_data, test_size=0.05)
```

[15]:
```
#
x_train_seq = train['Sequences']
y_train = train['Score']
x_test_seq = test['Sequences']
y_test = test['Score']
```

[16]:
```
#
def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        for index in sequence:
            results[i, index] += 1.
    return results
```

[17]:
```
#
x_train = vectorize_sequences(x_train_seq, max_words)
x_test = vectorize_sequences(x_test_seq, max_words)
```

[18]:
```
#
lr = LogisticRegression(random_state=random_state, max_iter=500)
```

[19]:
```
#
lr.fit(x_train, y_train)
```

[19]:
```
LogisticRegression(max_iter=500, random_state=42)
```

[20]:
```
#
print("Test accuracy:", lr.score(x_test, y_test))
```

```
Test accuracy: 0.9206349206349206
```

[21]:
```
#
print("Testing. enter -1 to exit")
```

```
print("Enter a sentence: ")
while True:
    text = input(">>> ")

    if (text == "-1"):
        break

    positive_preprocessed_text = preprocess(text, stop_words, punctuation_marks)
    positive_seq = text_to_sequence(positive_preprocessed_text, word_to_index)
    positive_bow = vectorize_sequences([positive_seq], max_words)

    result = lr.predict(positive_bow)
    print(result)
```

```
Testing. enter -1 to exit
Enter a sentence:
>>> Help! My house is on fire
[' fire']
>>> I've got a fever
[' medical']
>>> My dog is sick
[' animals']
>>> -1
```