Lab_6_Andrade

February 18, 2024

21 Febuary 2024 #

Lab 6 Assignment - CS 4315

Doug Andrade

1. Load the SMS file into a pandas dataframe using tab delimiters.

```
[23]: # Import the Pandas module for data frame operations
import pandas as pd
# Import the Numpy module for
import numpy as np

# CSV Text file to load
csv_file = 'SMSSpamCollection.csv'

# Read-in the csv file as a Pandas data frame, as an object to be operated on␣
↪later
spam_df = pd.read_csv(filepath_or_buffer = csv_file,
                      sep = '\t',
                      header = 0)

spam_df.head()
```

```
[23]:   Label                                               SMS
0   ham  Go until jurong point, crazy.. Available only …
1   ham                       Ok lar… Joking wif u oni…
2  spam  Free entry in 2 a wkly comp to win FA Cup fina…
3   ham  U dun say so early hor… U c already then say…
4   ham  Nah I don't think he goes to usf, he lives aro…
```

```
[24]: # Create binary label version for follow-on correlation analysis (1 = spam)
spam_binary = spam_df.copy()
spam_binary['Label'] = spam_binary['Label'].replace({'ham': 0, 'spam': 1})

spam_binary.head()
```

```
[24]:    Label                                               SMS
0       0  Go until jurong point, crazy.. Available only …
```

```
1      0                       Ok lar... Joking wif u oni...
2      1  Free entry in 2 a wkly comp to win FA Cup fina...
3      0  U dun say so early hor... U c already then say...
4      0  Nah I don't think he goes to usf, he lives aro...
```

2. Use lemmatization to create count vectors for each SMS message.

[3]:
```python
# Import CountVectorizer for conversion of text to a token count matrix
from sklearn.feature_extraction.text import CountVectorizer
# Import word_tokenize for dividing strings to list of substrings
from nltk import word_tokenize
# Import WordNetLemmatizer for reducing words to base form
from nltk.stem import WordNetLemmatizer
# Import search() to search for specific regular expressions
from re import search

# Create a custom tokenizer with lemmatization
class LemmaTokenizer:
    def __init__(self):
        # Initialize the word reduction function
        self.wnl = WordNetLemmatizer()
    def __call__(self, doc):
        # Regular expressions filter for numeric characters and short words
        regex_num_punctuation = '(\d+)|([^\w\s])'
        regex_little_words = r'(\b\w{1,2}\b)'
        # Tokenize and lemmatize tokens not in the regular expression filter
        return [self.wnl.lemmatize(t) for t in word_tokenize(doc)
                if not search(regex_num_punctuation, t) and not
                search(regex_little_words, t)]

# Initialize the text to token matrix function with lemmatization
text2vec_lemma = CountVectorizer(tokenizer = LemmaTokenizer(),
                                 stop_words = ['english', 'ha', 'le', 'wa'],
                                 lowercase = True)

# Apply the text vectorizer and lemmatization to the data frame's "SMS" column
text2vec_lemma.fit(spam_binary['SMS'])
```

```
/home/drandrade/anaconda3/lib/python3.11/site-
packages/sklearn/feature_extraction/text.py:525: UserWarning: The parameter
'token_pattern' will not be used since 'tokenizer' is not None'
  warnings.warn(
```

[3]:
```
CountVectorizer(stop_words=['english', 'ha', 'le', 'wa'],
                tokenizer=<__main__.LemmaTokenizer object at 0x7f35881e2950>)
```

[4]:
```python
list(text2vec_lemma.vocabulary_.items())[:5]
```

```
[4]: [('until', 6106),
      ('jurong', 3012),
      ('point', 4273),
      ('crazy', 1274),
      ('available', 409)]
```

3. Calculate the correlation between each token count and the spam variable.

```
[6]: # Transforms the fitted count matrix to vector of total count of each token
     vecs_lemma = text2vec_lemma.transform(spam_binary['SMS'])

     # Get key (word) form the .vocabulary_ dictionary of key-value pairs
     keys = list(text2vec_lemma.vocabulary_.keys())
     # Sort the list of the keys alphabetically
     keys.sort()

     # Create a new DataFrame with count vectors and concatenate it with spam_df
     vecs_df = pd.DataFrame(vecs_lemma.toarray(),
                            columns = keys)

     # Combine the new tokenized vector data frame with the original
     spam_binary = pd.concat([spam_binary, vecs_df],
                             axis = 1)
     spam_binary.head(3)
```

```
[6]:    Label                                                SMS  ____  aah  \
     0      0  Go until jurong point, crazy.. Available only …     0    0
     1      0                      Ok lar… Joking wif u oni…     0    0
     2      1  Free entry in 2 a wkly comp to win FA Cup fina…     0    0

        aaniye  aaooooright  aathi  abbey  abdomen  abeg  …  zed  zero  zhong  \
     0       0            0      0      0        0     0  …    0     0      0
     1       0            0      0      0        0     0  …    0     0      0
     2       0            0      0      0        0     0  …    0     0      0

        zindgi  zoe  zogtorius  zoom  zouk  zyada  ud
     0       0    0          0     0     0      0   0
     1       0    0          0     0     0      0   0
     2       0    0          0     0     0      0   0

     [3 rows x 6625 columns]
```

```
[7]: # Calulate the label column with key word (key)
     corrs = spam_binary[['Label'] + keys].corr()

     corrs.head()
```

[7]:
```
                Label      ____       aah    aaniye  aaooooright     aathi  \
Label        1.000000 -0.007456 -0.009132 -0.005272    -0.005272 -0.012919
____        -0.007456  1.000000 -0.000440 -0.000254    -0.000254 -0.000622
aah         -0.009132 -0.000440  1.000000 -0.000311    -0.000311 -0.000762
aaniye      -0.005272 -0.000254 -0.000311  1.000000    -0.000180 -0.000440
aaooooright -0.005272 -0.000254 -0.000311 -0.000180     1.000000 -0.000440

                abbey   abdomen      abeg      abel  …       zed      zero  \
Label       -0.005272 -0.005272 -0.005272 -0.005272  …  0.083443 -0.005272
____        -0.000254 -0.000254 -0.000254 -0.000254  … -0.000622 -0.000254
aah         -0.000311 -0.000311 -0.000311 -0.000311  … -0.000762 -0.000311
aaniye      -0.000180 -0.000180 -0.000180 -0.000180  … -0.000440 -0.000180
aaooooright -0.000180 -0.000180 -0.000180 -0.000180  … -0.000440 -0.000180

                zhong    zindgi       zoe  zogtorius      zoom      zouk  \
Label       -0.005272 -0.005272  0.020351  -0.005272 -0.005272  0.034050
____        -0.000254 -0.000254 -0.000359  -0.000254 -0.000254 -0.000254
aah         -0.000311 -0.000311 -0.000440  -0.000311 -0.000311 -0.000311
aaniye      -0.000180 -0.000180 -0.000254  -0.000180 -0.000180 -0.000180
aaooooright -0.000180 -0.000180 -0.000254  -0.000180 -0.000180 -0.000180

                zyada        ud
Label       -0.005272 -0.005272
____        -0.000254 -0.000254
aah         -0.000311 -0.000311
aaniye      -0.000180 -0.000180
aaooooright -0.000180 -0.000180

[5 rows x 6624 columns]
```

4. **Find the nearest neighbor to "I know that!" using Euclidean distance.**

[8]:
```python
# New data frame for nearest neighbor analysis, dropping the "SMS" column
nn_spam_df = spam_binary.drop(labels = ['SMS'],
                              axis = 1,
                              inplace = False)

# Convert the count vector to a Numpy array
vectors = nn_spam_df[nn_spam_df.columns[1:]].to_numpy()
# Vectorize the text to analyze, using CountVectorizer
example_vec = text2vec_lemma.transform(["I know that!"]).toarray()

print('The shape of the new Pandas dataframe is: %s\n\
The shape of the vectorized Numpy array is: %s\n\
The shape of the example vector is: %s' % (str(nn_spam_df.shape),
                                           str(vectors.shape),
                                           str(example_vec.shape)))
```

```
The shape of the new Pandas dataframe is: (5572, 6624)
The shape of the vectorized Numpy array is: (5572, 6623)
The shape of the example vector is: (1, 6623)
```

[26]:
```python
# Calculate Euclidean distances
euclid_dist = np.sqrt(np.sum((vectors - example_vec) ** 2,
                             axis = 1))

# Find the index of the nearest neighbor
euclid_idx = np.argmin(euclid_dist)

# Find the label of the nearest neighbor
euclid_label = spam_df.loc[euclid_idx, 'Label']

# Get the nearest neighbor SMS message
euclid_SMS = spam_df.loc[euclid_idx, 'SMS']

print('The nearest Euclidean neighbor to \"%s\":\n\
%s: \"%s\"' % (str('I know that!'),
               euclid_label,
               euclid_SMS))
```

```
The nearest Euclidean neighbor to "I know that!":
ham: "Ok.."
```

5. Finding the nearest neighbor to “I know that!” using cosine distance.

[27]:
```python
from numpy.linalg import norm

cos_dist = (
    (vectors / (norm(vectors, axis = 1) + 1e-10).reshape([-1, 1])) *
    (example_vec / norm(example_vec, axis = 1 ))
).sum(axis = 1)

# Find the index of the nearest neighbor
cos_idx = np.argmax(cos_dist)

# Find the label of the nearest neighbor
cos_label = spam_df.loc[cos_idx, 'Label']

# Get the nearest neighbor SMS message
cos_SMS = spam_df.loc[cos_idx, 'SMS']

print('The nearest cosine neighbor to \"%s\":\n\
%s: \"%s\"' % (str('I know that!'),
               cos_label,
               cos_SMS))
```

```
The nearest cosine neighbor to "I know that!":
ham: "I know that my friend already told that."
```