Imperial_S-CSIS312LA_FActivity4

December 7, 2024

De La Salle University – Dasmariñas
College of Information and Computer Studies
Computer Science Department

S–CSIS312LA — Natural Language Processing (Laboratory)
Instructor: Josephine T. Eduardo

1 Finals Activity 4

Name: Luis Anton P. Imperial
Section: BCS32

Type: Dropbox
Max score: 30
Category: Enabling Asssessment
Start: Nov 28, 3:00 pm
Due: Nov 29, 7:59 am

Max. attempts: 2
Allow late submissions:

1.1 Problem Description:

You are given two text strings. Your task is to compute how similar the two texts are after removing common stopwords and optional custom stopwords. The similarity score should be based on the number of common words between the two texts after stopword removal. Specifically, your function should calculate the Jaccard similarity index between the two cleaned texts, which is defined as:

Jaccard Similarity = Number of common words / Total number of unique words in both texts

1.1.1 Input:

- Two strings text1 and text2 (1 len(text1), len(text2) 10^4), which are the two text strings you need to compare.
- A list custom_stopwords (0 len(custom_stopwords) 100), which contains custom stopwords (optional).
  - Each custom stopword will be a string of lowercase letters (a-z), and the list will not contain any duplicates.

You can assume that the input text is in English, and punctuation marks are separated by spaces (e.g., “hello, world!” → “hello , world !”).

1.1.2 Output:

- A float representing the Jaccard similarity score of the two texts after stopword removal.

[13]:
```
import nltk, re
nltk.download('stopwords')
from nltk.corpus import stopwords

def remove_stopwords(text, custom_stopwords=None):
    stop_words = set(stopwords.words('english'))
    if custom_stopwords:
        stop_words.update(custom_stopwords)
    words = text.lower().split()

    # Use regex to remove punctuation and keep only words
    words = re.findall(r'\b\w+\b', text.lower())
    filtered_words = {word for word in words if word not in stop_words}
    return set(filtered_words)


def jaccard_similarity(text1, text2, custom_stopwords=None):
    set1 = remove_stopwords(text1, custom_stopwords)
    set2 = remove_stopwords(text2, custom_stopwords)

    intersection = len(set1.intersection(set2))
    union = len(set1.union(set2))

    if union == 0:
        return 0.0

    return intersection / union
```

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

[15]:
```
# Examples without custom stopwords
text1_1 = "This is the first document."
text2_1 = "This document is the second document."
similarity_1 = jaccard_similarity(text1_1, text2_1)
print("Your stopwords are:", remove_stopwords(text1_1),␣
 ↪remove_stopwords(text2_1))
print(f"Jaccard similarity (no custom stopwords): {similarity_1}", end="\n\n")

text1_2 = "The quick brown fox jumps over the lazy dog"
text2_2 = "The dog is lazy and the fox is quick"
similarity_2 = jaccard_similarity(text1_2, text2_2)
print("Your stopwords are:", remove_stopwords(text1_2),␣
 ↪remove_stopwords(text2_2))
```

```
print(f"Jaccard similarity (no custom stopwords): {similarity_2}", end="\n\n")

# Examples with custom stopwords
custom_stopwords = ["the", "is", "and"]
text1_3 = "This is the first document."
text2_3 = "This document is the second document."
similarity_3 = jaccard_similarity(text1_3, text2_3, custom_stopwords)
print("Your stopwords are:", remove_stopwords(text1_3),␣
 ↪remove_stopwords(text2_3))
print(f"Jaccard similarity (with custom stopwords): {similarity_3}", end="\n\n")

text1_4 = "The quick brown fox jumps over the lazy dog"
text2_4 = "The dog is lazy and the fox is quick"
similarity_4 = jaccard_similarity(text1_4, text2_4, custom_stopwords)
print("Your stopwords are:", remove_stopwords(text1_4),␣
 ↪remove_stopwords(text2_4))
print(f"Jaccard similarity (with custom stopwords): {similarity_4}")
```

```
Your stopwords are: {'document', 'first'} {'second', 'document'}
Jaccard similarity (no custom stopwords): 0.3333333333333333

Your stopwords are: {'dog', 'fox', 'jumps', 'lazy', 'brown', 'quick'} {'lazy',
'dog', 'quick', 'fox'}
Jaccard similarity (no custom stopwords): 0.6666666666666666

Your stopwords are: {'document', 'first'} {'second', 'document'}
Jaccard similarity (with custom stopwords): 0.3333333333333333

Your stopwords are: {'dog', 'fox', 'jumps', 'lazy', 'brown', 'quick'} {'lazy',
'dog', 'quick', 'fox'}
Jaccard similarity (with custom stopwords): 0.6666666666666666
```

1.2 Calculating Jaccard Similarity Score

The Jaccard similarity index is calculated as:

`Jaccard Similarity = Number of common words / Total number of unique words in both texts`

If there are no common words between the two texts after removing stopwords, the intersection will be empty, resulting in a similarity score of 0.0.

1.2.1 Examples without Custom Stopwords:

```
text1_1 = "This is the first document."
text2_1 = "This document is the second document."
similarity_1 = jaccard_similarity(text1_1, text2_1)
print("Your stopwords are:", remove_stopwords(text1_1), remove_stopwords(text2_1))
print(f"Jaccard similarity (no custom stopwords): {similarity_1}", end="\n\n")
```

```
text1_2 = "The quick brown fox jumps over the lazy dog"
text2_2 = "The dog is lazy and the fox is quick"
similarity_2 = jaccard_similarity(text1_2, text2_2)
print("Your stopwords are:", remove_stopwords(text1_2), remove_stopwords(text2_2))
print(f"Jaccard similarity (no custom stopwords): {similarity_2}", end="\n\n")
```

1. **Define Text Strings:** Two text strings, text1_1 and text2_1, are defined.
2. **Calculate Similarity:** The jaccard_similarity function is called with text1_1 and text2_1 as arguments to calculate their similarity. The result is stored in the variable similarity_1.
3. **Print Results:** The code then prints the stopwords removed from each text using the remove_stopwords function and the calculated Jaccard similarity score (similarity_1).
4. **Repeat:** The process is repeated with another set of text strings, text1_2 and text2_2, and the results are stored in similarity_2 and printed.

1.2.2 Examples with Custom Stopwords:

```
custom_stopwords = ["the", "is", "and"]
text1_3 = "This is the first document."
text2_3 = "This document is the second document."
similarity_3 = jaccard_similarity(text1_3, text2_3, custom_stopwords)
print("Your stopwords are:", remove_stopwords(text1_3), remove_stopwords(text2_3))
print(f"Jaccard similarity (with custom stopwords): {similarity_3}", end="\n\n")

text1_4 = "The quick brown fox jumps over the lazy dog"
text2_4 = "The dog is lazy and the fox is quick"
similarity_4 = jaccard_similarity(text1_4, text2_4, custom_stopwords)
print("Your stopwords are:", remove_stopwords(text1_4), remove_stopwords(text2_4))
print(f"Jaccard similarity (with custom stopwords): {similarity_4}")
```

1. **Define Custom Stopwords:** A list named custom_stopwords is created, containing words to be considered as stopwords in addition to the default English stopwords.
2. **Calculate Similarity:** Similar to the previous examples, the jaccard_similarity function is called, but this time with the custom_stopwords list as an additional argument. The results are stored in similarity_3 and similarity_4, respectively.
3. **Print Results:** The code prints the stopwords removed from each text and the calculated Jaccard similarity scores.

In essence, this section provides practical examples of how to use the defined functions to calculate the similarity between two texts, both with and without custom stopwords. It showcases how the jaccard_similarity function can be applied in different scenarios and how the custom_stopwords argument can be used to tailor the analysis.