Removing stop words with NLTK in Python

In **natural language processing (NLP)**, **stopwords** are frequently filtered out to enhance text analysis and computational efficiency. Eliminating stopwords can improve the accuracy and relevance of NLP tasks by drawing attention to the more important words, or content words. The article aims to explore stopwords.

Table of Content

Certain words, like “the,” “and,” and “is,” are thought to be ineffective for communicating important information. The objective of eliminating words that add little or nothing to the comprehension of the text is to expedite text processing, even though the list of stopwords may differ.

What are Stop words?

A stop word is a commonly used word (such as “the”, “a”, “an”, or “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.

We would not want these words to take up space in our database or take up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to stop words. NLTK(Natural Language Toolkit) in Python has a list of stopwords stored in 16 different languages. You can find them in the nltk_data directory. **home/PratimaPython/nltk_data/corpora/stopwords** are the directory address. **or** (Do not forget to change your home directory name)

Sample text with Stop Words	Without Stop Words
GeeksforGeeks – A Computer Science Portal for Geeks	GeeksforGeeks , Computer Science, Portal ,Geeks
Can listening be exhausting?	Listening, Exhausting
I like reading, so I read	Like, Reading, read

Need to remove the Stopwords

The necessity of removing stopwords in NLP is contingent upon the specific task at hand. For text classification tasks, where the objective is to categorize text into distinct groups, excluding stopwords is common practice. This is done to channel more attention towards words that truly convey the essence of the text. As illustrated earlier, certain words like “there,” “book,” and “table” contribute significantly to the text’s meaning, unlike less informative words such as “is” and “on.”

Conversely, for tasks like machine translation and text summarization, the removal of stopwords is not recommended. In these scenarios, every word plays a pivotal role in preserving the original meaning of the content.

Types of Stopwords

Stopwords are frequently occurring words in a language that are frequently omitted from natural language processing (NLP) tasks due to their low significance for deciphering textual meaning. The particular list of stopwords can change based on the language being studied and the context. The following is a broad list of stopword categories:

- **Common Stopwords:** These are the most frequently occurring words in a language and are often removed during text preprocessing. Examples include “the,” “is,” “in,” “for,” “where,” “when,” “to,” “at,” etc.
- **Custom Stopwords:** Depending on the specific task or domain, additional words may be considered as stopwords. These could be domain-specific terms that don’t contribute much to the overall meaning. For example, in a medical context, words like “patient” or “treatment” might be considered as custom stopwords.
- **Numerical Stopwords:** Numbers and numeric characters may be treated as stopwords in certain cases, especially when the analysis is focused on the meaning of the text rather than specific numerical values.
- **Single-Character Stopwords:** Single characters, such as “a,” “I,” “s,” or “x,” may be considered stopwords, particularly in cases where they don’t convey much meaning on their own.

- **Contextual Stopwords:** Words that are stopwords in one context but meaningful in another may be considered as contextual stopwords. For instance, the word “will” might be a stopword in the context of general language processing but could be important in predicting future events.

Checking English Stopwords List

An English stopwords list typically includes common words that carry little semantic meaning and are often excluded during text analysis. Examples of these words are “the,” “and,” “is,” “in,” “for,” and “it.” These stopwords are frequently removed to focus on more meaningful terms when processing text data in natural language processing tasks such as text classification or sentiment analysis.

To check the list of stopwords you can type the following commands in the python shell.

Python3

```
import  nltk

from  nltk.corpus  import  stopwords

nltk.download(``'stopwords'``)

print``(stopwords.words(``'english'``))
```

Output:

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours',
'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself',
'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself',
'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves',
 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are
 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'd
 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'b
'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above'
'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', '
'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', '
'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm',
 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't",
'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
 "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't",
'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

Note: You can even modify the list by adding words of your choice in the English .txt. file in the stopwords directory.

Removing stop words with NLTK

The following program removes stop words from a piece of text:

Python3

```
from  nltk.corpus  import  stopwords

from  nltk.tokenize  import  word_tokenize

example_sent  =

stop_words  =  set``(stopwords.words(``'english'``))

word_tokens  =  word_tokenize(example_sent)

filtered_sentence  =  [w  for  w  in  word_tokens  if  not  w.lower()  in  stop_words]

filtered_sentence  =  []

for  w  in  word_tokens:

  if  w  not  in  stop_words:

    filtered_sentence.append(w)

print``(word_tokens)

print``(filtered_sentence)
```

Output:

```
['This', 'is', 'a', 'sample', 'sentence', ',', 'showing',
'off', 'the', 'stop', 'words', 'filtration', '.']
['This', 'sample', 'sentence', ',', 'showing', 'stop',
'words', 'filtration', '.']
```

The provided Python code demonstrates stopword removal using the Natural Language Toolkit (NLTK) library. In the first step, the sample sentence, which reads “This is a sample sentence, showing off the stop words filtration,” is tokenized into words using the `word_tokenize` function. The code then filters out stopwords by converting each word to lowercase and checking its presence in the set

of English stopwords obtained from NLTK. The resulting `filtered_sentence` is printed, showcasing both lowercased and original versions, providing a cleaned version of the sentence with common English stopwords removed.

Removing stop words with SpaCy

Python3

```
import spacy

nlp = spacy.load(``"en_core_web_sm"``)

text = "There is a pen on the table"

doc = nlp(text)

filtered_words = [token.text for token in doc if not token.is_stop]

clean_text = ' '``.join(filtered_words)

print``(``"Original Text:"``, text)

print``(``"Text after Stopword Removal:"``, clean_text)
```

Output:

```
Original Text: There is a pen on the table
Text after Stopword Removal: pen table
```

The provided Python code utilizes the spaCy library for natural language processing to remove stopwords from a sample text. Initially, the spaCy English model is loaded, and the sample text, “There is a pen on the table,” is processed using spaCy. Stopwords are then filtered out from the processed tokens, and the resulting non-stopword tokens are joined to create a clean version of the text.

Removing stop words with Genism

Python3

```
from gensim.parsing.preprocessing import remove_stopwords
```

```
new_text = "The majestic mountains provide a breathtaking view."

new_filtered_text = remove_stopwords(new_text)

print(("Original Text:", new_text)

print(("Text after Stopword Removal:", new_filtered_text)
```

Output:

```
Original Text: The majestic mountains provide a breathtaking view.
Text after Stopword Removal: The majestic mountains provide breathtaking view.
```

The provided Python code utilizes Gensim's `remove_stopwords` function to preprocess a sample text. In this specific example, the original text is "The majestic mountains provide a breathtaking view." The `remove_stopwords` function efficiently eliminates common English stopwords, resulting in a filtered version of the text, which is then printed alongside the original text.

Removing stop words with SkLearn

Python3

```
from sklearn.feature_extraction.text import CountVectorizer

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize

new_text = "The quick brown fox jumps over the lazy dog."

new_words = word_tokenize(new_text)

new_filtered_words = [

  word for word in new_words if word.lower() not in stopwords.words(('english'))]

new_clean_text = ' '(('.join(new_filtered_words)

print(("Original Text:", new_text)

print(("Text after Stopword Removal:", new_clean_text)
```

Output:

```
Original Text: The quick brown fox jumps over the lazy dog.
Text after Stopword Removal: quick brown fox jumps lazy dog .
```

The provided Python code combines scikit-learn and NLTK for stopword removal and text processing. First, the sample text, "The quick brown fox jumps over the lazy dog," is tokenized into words using NLTK's `word_tokenize` function. Subsequently, common English stopwords are removed by iterating through the tokenized words and checking their absence in the NLTK stopwords set. The final step involves joining the non-stopword tokens to create a clean version of the text. This approach integrates scikit-learn's `CountVectorizer` could be utilized for further text analysis, such as creating a bag-of-words representation.

Also Check:

- Natural Language Processing (NLP) Tutorial
- Python | Lemmatization with NLTK
- Introduction to Stemming

Frequently Asked Questions

1. What are stopwords in NLP?

> Stopwords are common words in a language that are often filtered out during NLP tasks because they are considered to carry little meaning or contribute minimally to the overall understanding of a text. Examples include "the," "is," "and," "in," etc.

2. Why do we remove stopwords in NLP?

> Stopwords are removed in NLP to focus on the more meaningful and informative words in a text. This is often done to reduce noise, improve efficiency in processing, and highlight keywords that carry the essential meaning of the text.

3. Can the list of stopwords vary between NLP libraries?

> Yes, the list of stopwords can vary between NLP libraries. Different libraries, such as NLTK, spaCy, Gensim, and scikit-learn, may provide their own sets of stopwords. Users can also customize the list based on their specific needs.

4. When should stopwords be retained?

> Stopwords should be retained when they contribute to the context, coherence, or specific linguistic features of the text. Tasks like machine translation, text summarization, and certain

> sentiment analysis scenarios may benefit from retaining stopwords.

5. What are some common English stopwords?

> Common English stopwords include “the,” “and,” “is,” “in,” “for,” “where,” “when,” “to,” “at,” etc. These words are frequently used but are often excluded during text analysis.