Published in Towards Data Science · Follow

Chetna Khanna · Follow
Feb 10, 2021 · 12 min read

Text pre-processing: Stop words removal using different libraries

A handy guide about English stop words removal in Python

Image by Kai on Unsplash

We are well aware of the fact that computers can easily process numbers if programmed well. 🤵 However, a large portion of the information we have is in the form of text. 📗 We communicate with each other by directly talking with them or using text messages, social media posts, phone calls, video calls, etc. In order to create intelligent systems, we need to use this information that we have in abundance.

Natural Language Processing (NLP) is the branch of Artificial Intelligence that allows machines to interpret human language. 👍🏻 However, the same cannot be used directly by the machine, and we need to pre-process the same first.

Text pre-processing is the process of preparing text data so that machines can use the same to perform tasks like analysis, predictions, etc. There are many different steps in text pre-processing but in this article, we will only get familiar with stop words, why do we remove them, and the different libraries that can be used to remove them.

So, let's get started. 🏃🏻‍♀️

What are stop words? 🤔

The words which are generally filtered out before processing a natural language are called **stop words**. These are actually the most common words in any language (like articles, prepositions, pronouns, conjunctions, etc) and does not add much information to the

our text in order to give more focus to the important information. In order words, we can say that the removal of such words does not show any negative consequences on the model we train for our task.

Removal of stop words definitely reduces the dataset size and thus reduces the training time due to the fewer number of tokens involved in the training.

Do we always remove stop words? Are they always useless for us? 🙋

The answer is no! 🤷

We do not always remove the stop words. The removal of stop words is highly dependent on the task we are performing and the goal we want to achieve. For example, if we are training a model that can perform the sentiment analysis task, we might not remove the stop words.

Movie review: "The movie was not good at all."

Text after removal of stop words: "movie good"

We can clearly see that the review for the movie was negative. However, after the removal of stop words, the review became positive, which is not the reality. Thus, the removal of stop words can be problematic here.

Tasks like text classification do not generally need stop words as the other words present in the dataset are more important and give the general idea of the text. So, we generally remove stop words in such tasks.

In a nutshell, NLP has a lot of tasks that cannot be accomplished properly after the removal of stop words. So, think before performing this step. The catch here is that no rule is universal and no stop words list is universal. A list not conveying any important information to one task can convey a lot of information to the other task.

Word of caution: Before removing stop words, research a bit about your task and the problem you are trying to solve, and then make your decision.

What are the different libraries to remove stop words? 🤦

NLP is one of the most researched areas today and there have been many revolutionary developments in this field. NLP relies on advanced computational skills and developers across the world have created many different tools to handle human language. Out of so many libraries out there, a few are quite popular and help a lot in performing many different NLP tasks.

Some of the libraries used for the removal of English stop words, the stop words list along with the code are given below.

Natural Language Toolkit (NLTK):

NLTK is an amazing library to play with natural language. When you will start your NLP journey, this is the first library that you will use. The steps to import the library and the English stop words list is given below:

```
import nltk
from nltk.corpus import stopwords
sw_nltk = stopwords.words('english')
print(sw_nltk)
```

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', 'doing', 'a', 'an', 'the',
'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about',
'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from',
'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here',
```

```
'wouldn', "wouldn't"]
```

Let us check how many stop words this library has.

```
print(len(sw_nltk))
```

Output:

```
179
```

Let us remove stop words from a text.

```
text = "When I first met her she was very quiet. She remained quiet during the entire two hour long
journey from Stony Brook to New York."

words = [word for word in text.split() if word.lower() not in sw_nltk]
new_text = " ".join(words)

print(new_text)
print("Old length: ", len(text))
print("New length: ", len(new_text))
```

The above code is quite simple but I will still explain it for beginners. I have the text and I split this text into words as stop words is a list of words. I then changed the words to lowercase as all words in the list of stop words are in lowercase. Then I created a list of all words which are not in the stop words list. The resulting list is then joined to form the sentence again.

Output:

```
first met quiet. remained quiet entire two hour long journey Stony Brook New York.
Old length:  129
New length:  82
```

We can clearly see that the removal of stop words reduced the length of the sentence from 129 to 82.

Kindly note that I will be using similar code to explain stop words in each of the libraries.

spaCy:

spaCy is an open-source software library for advanced NLP. This library is quite popular now and NLP practitioners use this to get their work done in the best way.

```
import spacy
#loading the english language small model of spacy
en = spacy.load('en_core_web_sm')
sw_spacy = en.Defaults.stop_words
print(sw_spacy)
```

Output:

```
{'those', 'on', 'own', '’ve', 'yourselves', 'around', 'between', 'four', 'been', 'alone', 'off', 'am',
'then', 'other', 'can', 'regarding', 'hereafter', 'front', 'too', 'used', 'wherein', '‘ll', 'doing',
'everything', 'up', 'onto', 'never', 'either', 'how', 'before', 'anyway', 'since', 'through', 'amount',
```




```
'third', 'anything', 'twelve', 'against', 'while', 'twenty', 'if', 'however', 'herself', 'when', 'may',
'ours', 'six', 'done', 'seems', 'else', 'call', 'perhaps', 'had', 'nevertheless', 'where', 'otherwise',
'still', 'within', 'its', 'for', 'together', 'elsewhere', 'throughout', 'of', 'others', 'show', '’s',
'anywhere', 'anyhow', 'as', 'are', 'the', 'hence', 'something', 'hereby', 'nowhere', 'latterly', 'say',
'does', 'neither', 'his', 'go', 'forty', 'put', 'their', 'by', 'namely', 'could', 'five', 'unless',
'itself', 'is', 'nine', 'whereafter', 'down', 'bottom', 'thereby', 'such', 'both', 'she', 'become',
'whole', 'who', 'yourself', 'every', 'thru', 'except', 'very', 'several', 'among', 'being', 'be', 'mine',
'further', 'n‘t', 'here', 'during', 'why', 'with', 'just', "'s", 'becomes', '’ll', 'about', 'a', 'using',
'seeming', "'d", "'ll", "'re", 'due', 'wherever', 'beforehand', 'fifty', 'becoming', 'might', 'amongst',
'my', 'empty', 'thence', 'thereafter', 'almost', 'least', 'someone', 'often', 'from', 'keep', 'him', 'or',
'‘m', 'top', 'her', 'nobody', 'sometime', 'across', '‘s', '’re', 'hundred', 'only', 'via', 'name',
'eight', 'three', 'back', 'to', 'all', 'became', 'move', 'me', 'we', 'formerly', 'so', 'i', 'whence',
'under', 'always', 'himself', 'in', 'herein', 'more', 'after', 'themselves', 'you', 'above', 'sixty',
'them', 'your', 'made', 'indeed', 'most', 'everywhere', 'fifteen', 'but', 'must', 'along', 'beside',
'hers', 'side', 'former', 'anyone', 'full', 'has', 'yours', 'whose', 'behind', 'please', 'ten', 'seemed',
'sometimes', 'should', 'over', 'take', 'each', 'same', 'rather', 'really', 'latter', 'and', 'ca',
'hereupon', 'part', 'per', 'eleven', 'ever', '‘re', 'enough', "n't", 'again', '‘d', 'us', 'yet',
'moreover', 'mostly', 'one', 'meanwhile', 'whither', 'there', 'toward', '’m', "'ve", '’d', 'give', 'do',
'an', 'quite', 'these', 'everyone', 'towards', 'this', 'cannot', 'afterwards', 'beyond', 'make', 'were',
'whether', 'well', 'another', 'below', 'first', 'upon', 'any', 'none', 'many', 'serious', 'various', 're',
'two', 'less', '‘ve'}
```

Quite a long list. Let us check how many stop words this library has.

```
print(len(sw_spacy))
```

Output:

```
326
```

Wow, 326! Let us remove stop words from our previous text.

```
words = [word for word in text.split() if word.lower() not in sw_spacy]
new_text = " ".join(words)
print(new_text)
print("Old length: ", len(text))
print("New length: ", len(new_text))
```

Output:

```
met quiet. remained quiet entire hour lomg journey Stony Brook New York.
Old length:  129
New length:  72
```

We can clearly see that the removal of stop words reduced the length of the sentence from 129 to 72, even shorter than NLTK because the spaCy library has more stop words than NLTK. The results, in this case, are quite similar though.

Gensim:

Gensim (Generate Similar) is an open-source software library that uses modern statistical machine learning. *According to Wikipedia, Gensim is designed to handle large text collections using data streaming and incremental online algorithms, which differentiates it from most other machine learning software packages that target only in-memory processing.*

```
import gensim
from gensim.parsing.preprocessing import remove_stopwords, STOPWORDS
print(STOPWORDS)
```


```
frozenset({'those', 'on', 'own', 'yourselves', 'ie', 'around', 'between', 'four', 'been', 'alone', 'off',
'am', 'then', 'other', 'can', 'cry', 'regarding', 'hereafter', 'front', 'too', 'used', 'wherein', 'doing',
'everything', 'up', 'never', 'onto', 'how', 'either', 'before', 'anyway', 'since', 'through', 'amount',
'now', 'he', 'cant', 'was', 'con', 'have', 'into', 'because', 'inc', 'not', 'therefore', 'they', 'even',
'whom', 'it', 'see', 'somewhere', 'interest', 'thereupon', 'thick', 'nothing', 'whereas', 'much',
'whenever', 'find', 'seem', 'until', 'whereby', 'at', 'ltd', 'fire', 'also', 'some', 'last', 'than',
'get', 'already', 'our', 'doesn', 'once', 'will', 'noone', 'that', 'what', 'thus', 'no', 'myself', 'out',
'next', 'whatever', 'although', 'though', 'etc', 'which', 'would', 'therein', 'nor', 'somehow',
'whereupon', 'besides', 'whoever', 'thin', 'ourselves', 'few', 'did', 'third', 'without', 'twelve',
'anything', 'against', 'while', 'twenty', 'if', 'however', 'found', 'herself', 'when', 'may', 'six',
'ours', 'done', 'seems', 'else', 'call', 'perhaps', 'had', 'nevertheless', 'fill', 'where', 'otherwise',
'still', 'within', 'its', 'for', 'together', 'elsewhere', 'throughout', 'of', 'eg', 'others', 'show',
'sincere', 'anywhere', 'anyhow', 'as', 'are', 'the', 'hence', 'something', 'hereby', 'nowhere',
'latterly', 'de', 'say', 'does', 'neither', 'his', 'go', 'forty', 'put', 'their', 'by', 'namely', 'km',
'could', 'five', 'unless', 'itself', 'is', 'nine', 'whereafter', 'down', 'bottom', 'thereby', 'such',
'both', 'she', 'become', 'whole', 'who', 'yourself', 'every', 'thru', 'except', 'very', 'several',
'among', 'being', 'be', 'mine', 'further', 'here', 'during', 'why', 'with', 'just', 'becomes', 'about',
'a', 'co', 'using', 'seeming', 'due', 'wherever', 'beforehand', 'detail', 'fifty', 'becoming', 'might',
'amongst', 'my', 'empty', 'thence', 'thereafter', 'almost', 'least', 'someone', 'often', 'from', 'keep',
'him', 'or', 'top', 'her', 'didn', 'nobody', 'sometime', 'across', 'hundred', 'only', 'via', 'name',
'eight', 'three', 'back', 'to', 'all', 'became', 'move', 'me', 'we', 'formerly', 'so', 'i', 'whence',
'describe', 'under', 'always', 'himself', 'more', 'herein', 'in', 'after', 'themselves', 'you', 'them',
'above', 'sixty', 'hasnt', 'your', 'made', 'everywhere', 'indeed', 'most', 'kg', 'fifteen', 'but', 'must',
'along', 'beside', 'hers', 'computer', 'side', 'former', 'full', 'anyone', 'has', 'yours', 'whose',
'behind', 'please', 'mill', 'amoungst', 'ten', 'seemed', 'sometimes', 'should', 'over', 'take', 'each',
'don', 'same', 'rather', 'really', 'latter', 'and', 'part', 'hereupon', 'per', 'eleven', 'ever', 'enough',
'again', 'us', 'yet', 'moreover', 'mostly', 'one', 'meanwhile', 'whither', 'there', 'toward', 'give',
'system', 'do', 'quite', 'an', 'these', 'everyone', 'towards', 'this', 'bill', 'cannot', 'un',
'afterwards', 'beyond', 'make', 'were', 'whether', 'well', 'another', 'below', 'first', 'upon', 'any',
'none', 'many', 'various', 'serious', 're', 'two', 'less', 'couldnt'})
```

Quite a long list again. Let us check how many stop words this library has.

```
print(len(STOPWORDS))
```

Output:

```
337
```

Ummm! Similar count as spaCy. Let us remove stop words from our text.

```
new_text = remove_stopwords(text)
print(new_text)

print("Old length: ", len(text))
print("New length: ", len(new_text))
```

We can see that it is quite simple to remove stop words using the Gensim library.

Output:

```
When I met quiet. She remained quiet entire hour long journey Stony Brook New York.
Old length:  129
New length:  83
```

Removal of stop words reduced the length of the sentence from 129 to 83. We can see that even though the length of stop words in spaCy and Gensim is similar, the resulting text is quite different.




```
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
print(ENGLISH_STOP_WORDS)
```

Output:

```
frozenset({'those', 'on', 'own', 'yourselves', 'ie', 'around', 'between', 'four', 'been', 'alone', 'off',
'am', 'then', 'other', 'can', 'cry', 'hereafter', 'front', 'too', 'wherein', 'everything', 'up', 'onto',
'never', 'either', 'how', 'before', 'anyway', 'since', 'through', 'amount', 'now', 'he', 'cant', 'was',
'con', 'have', 'into', 'because', 'inc', 'not', 'therefore', 'they', 'even', 'whom', 'it', 'see',
'somewhere', 'interest', 'thereupon', 'nothing', 'thick', 'whereas', 'much', 'whenever', 'find', 'seem',
'until', 'whereby', 'at', 'ltd', 'fire', 'also', 'some', 'last', 'than', 'get', 'already', 'our', 'once',
'will', 'noone', 'that', 'what', 'thus', 'no', 'myself', 'out', 'next', 'whatever', 'although', 'though',
'etc', 'which', 'would', 'therein', 'nor', 'somehow', 'whereupon', 'besides', 'whoever', 'thin',
'ourselves', 'few', 'third', 'without', 'anything', 'twelve', 'against', 'while', 'twenty', 'if',
'however', 'found', 'herself', 'when', 'may', 'ours', 'six', 'done', 'seems', 'else', 'call', 'perhaps',
'had', 'nevertheless', 'fill', 'where', 'otherwise', 'still', 'within', 'its', 'for', 'together',
'elsewhere', 'throughout', 'of', 'eg', 'others', 'show', 'sincere', 'anywhere', 'anyhow', 'as', 'are',
'the', 'hence', 'something', 'hereby', 'nowhere', 'de', 'latterly', 'neither', 'his', 'go', 'forty',
'put', 'their', 'by', 'namely', 'could', 'five', 'itself', 'is', 'nine', 'whereafter', 'down', 'bottom',
'thereby', 'such', 'both', 'she', 'become', 'whole', 'who', 'yourself', 'every', 'thru', 'except', 'very',
'several', 'among', 'being', 'be', 'mine', 'further', 'here', 'during', 'why', 'with', 'becomes', 'about',
'a', 'co', 'seeming', 'due', 'wherever', 'beforehand', 'detail', 'fifty', 'becoming', 'might', 'amongst',
'my', 'empty', 'thence', 'thereafter', 'almost', 'least', 'someone', 'often', 'from', 'keep', 'him', 'or',
'top', 'her', 'nobody', 'sometime', 'across', 'hundred', 'only', 'via', 'name', 'eight', 'three', 'back',
'to', 'all', 'became', 'move', 'me', 'we', 'formerly', 'so', 'i', 'whence', 'describe', 'under', 'always',
'himself', 'in', 'herein', 'more', 'after', 'themselves', 'you', 'above', 'sixty', 'them', 'hasnt',
'your', 'made', 'indeed', 'most', 'everywhere', 'fifteen', 'but', 'must', 'along', 'beside', 'hers',
'side', 'former', 'anyone', 'full', 'has', 'yours', 'whose', 'behind', 'please', 'amoungst', 'mill',
'ten', 'seemed', 'sometimes', 'should', 'over', 'take', 'each', 'same', 'rather', 'latter', 'and',
'hereupon', 'part', 'per', 'eleven', 'ever', 'enough', 'again', 'us', 'yet', 'moreover', 'mostly', 'one',
'meanwhile', 'whither', 'there', 'toward', 'give', 'system', 'do', 'an', 'these', 'everyone', 'towards',
'this', 'bill', 'cannot', 'un', 'afterwards', 'beyond', 'were', 'whether', 'well', 'another', 'below',
'first', 'upon', 'any', 'none', 'many', 'serious', 're', 'two', 'couldnt', 'less'})
```

Quite a long list again. Let us check how many stop words this library has.

```
print(len(ENGLISH_STOP_WORDS))
```

Output:

```
318
```

Let us remove stop words from our text.

```
words = [word for word in text.split() if word.lower() not in ENGLISH_STOP_WORDS]
new_text = " ".join(words)
print(new_text)
print("Old length: ", len(text))
print("New length: ", len(new_text))
```

Output:

```
met quiet. remained quiet entire hour long journey Stony Brook New York.
Old length:  129
New length:  72
```




Can I add my own stop words to the list? ✍️

Yes, we can also add custom stop words to the list of stop words available in these libraries to serve our purpose.

Here is the code to add some custom stop words to NLTK's stop words list:

```
sw_nltk.extend(['first', 'second', 'third', 'me'])
print(len(sw_nltk))
```

Output:

```
183
```

We can see that the length of NLTK stop words is 183 now instead of 179. And, we can now use the same code to remove stop words from our text.

Can I remove stop words from the premade list? 👋

Yes, if we want we can also remove stop words from the list available in these libraries.

Here is the code using the NLTK library:

```
sw_nltk.remove('not')
```

The stop word 'not' is now removed from the stop words list.

Depending on the library you are using, you can perform the relevant operations to add or remove stop words from the premade list. I am pointing this because NLTK returns a list of stop words while the other libraries return a set of stop words.

If we do not want to use any of these libraries, we can also create our own custom stop words list and use it in our task. This is usually done when we have domain expertise in our field and when we know which words we should avoid while performing our task.

Look at the below code to see how simple this is.

```
#create your custom stop words list
my_stop_words = ['her','me','i','she','it']

words = [word for word in text.split() if word.lower() not in my_stop_words]
new_text = " ".join(words)
print(new_text)
print("Old length: ", len(text))
print("New length: ", len(new_text))
```

Output:

```
When first met was very quiet. remained quiet during the entire two hour long journey from Stony Brook to
New York.
Old length:  129
New length:  115
```

In a similar way, you can create your list of stop words according to your task and use it. 🤟

We have observed in this article that different libraries have a different collection of stop words and we can clearly say that stop words

Thank you, everyone, for reading this. Do share your valuable feedback or suggestion regarding this post! Happy reading! 📗 🖌

LinkedIn