In any Machine learning task, cleaning or preprocessing the data is as important as model building. Text data is one of the most unstructured forms of available data and when comes to deal with Human language then it's too complex. Have you ever wondered how Alexa, Siri, Google assistant can understand, process, and respond in Human language. NLP is a technology that works behind it where before any response lots of text preprocessing takes place. This tutorial will study the main text preprocessing techniques that you must know to work with any text data.

Text Preprocessing

Text preprocessing is a method to clean the text data and make it ready to feed data to the model. Text data contains noise in various forms like emotions, punctuation, text in a different case. When we talk about Human Language then, there are different ways to say the same thing, And this is only the main problem we have to deal with because machines will not understand words, they need numbers so we need to convert text to numbers in an efficient manner.

Libraries used to deal with NLP problems

There are many libraries and algorithms used to deal with NLP-based problems. A regular expression(re) is mostly used library for text cleaning. NLTK(Natural language toolkit) and spacy are the next level library used for performing Natural language tasks like removing stopwords, named entity recognition, part of speech tagging, phrase matching, etc.

```
# import and instantiate CountVectorizer (wi
th the default parameters)
from sklearn.feature_extraction.text import
CountVectorizer

vect = CountVectorizer()

# learn the 'vocabulary' of the training dat
a (occurs in-place)
vect.fit(simple_train)

# examine the fitted vocabulary
vect.get_feature_names_out()
```

Out[3]:

```
array(['cab', 'call', 'me', 'please', 'toni
ght', 'you'], dtype=object)
```

In [4]:

```
# transform training data into a 'document-t
erm matrix'
simple_train_dtm = vect.transform(simple_tr
ain)
simple_train_dtm
```

```
<3x6 sparse matrix of type '<class 'numpy.int64'>'
        with 9 stored elements in Compressed Sparse Row format>
```

In [5]:

```
# convert sparse matrix to a dense matrix
simple_train_dtm.toarray()
```

Out[5]:

```
array([[0, 1, 0, 0, 1, 1],
       [1, 1, 1, 0, 0, 0],
       [0, 1, 1, 2, 0, 0]])
```

In [6]:

```
# examine the vocabulary and document-term matrix together
pd.DataFrame(simple_train_dtm.toarray(), columns=vect.get_feature_names())
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

Out[6]:

	cab	call	me	please	tonight	you
0	0	1	0	0	1	1
1	1	1	1	0	0	0
2	0	1	1	2	0	0