

Handling Text

<code>text='Some words'</code>	assign string
<code>list(text)</code>	Split text into character tokens
<code>set(text)</code>	Unique tokens
<code>len(text)</code>	Number of characters

Accessing corpora and lexical resources

<code>from nltk.corpus import brown</code>	import CorpusReader object
<code>brown.words(text_id)</code>	Returns pretokenised document as list of words
<code>brown.fileids()</code>	Lists docs in Brown corpus
<code>brown.categories()</code>	Lists categories in Brown corpus

Tokenization

<code>text.split(" ")</code>	Split by space
<code>nltk.word_tokenize(text)</code>	nltk in-built word tokenizer
<code>nltk.sent_tokenize(doc)</code>	nltk in-built sentence tokenizer

Lemmatization & Stemming

<code>input= "List listed lists listing listings"</code>	Different suffixes
<code>words= input.lower().split(' ')</code>	Normalize (lower-case) words
<code>porter = nltk.PorterStemmer</code>	Initialise Stemmer
<code>[porter.stem(t) for t in words]</code>	Create list of stems
<code>WNL= nltk.WordNetLemmatizer()</code>	Initialise WordNet lemmatizer
<code>[WNL.lemmatize(t) for t in words]</code>	Use the lemmatizer

Part of Speech (POS) Tagging

<code>nltk.help.upenn_tagset('MD')</code>	Lookup definition for a POS tag
<code>nltk.pos_tag(words)</code>	nltk in-built POS tagger
<use an alternative tagger to illustrate ambiguity>	

Sentence Parsing

<code>g=nltk.data.load('grammar.cfg')</code>	Load a grammar from a file
<code>g=nltk CFG.fromstring("""...""")</code>	Manually define grammar
<code>parser = nltk.ChartParser(g)</code>	Create a parser out of the grammar
<code>trees= parser.parse_all(text)</code>	
<code>for tree in trees: ... print tree</code>	
<code>from nltk.corpus import treebank</code>	
<code>treebank.parsed_sents('wsj_0001.mrg')</code>	Treebank parsed sentences

Text Classification

<code>from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer</code>	
<code>vect=CountVecorizer().fit(X_train)</code>	Fit bag of words model to data
<code>vect.get_feature_names()</code>	Get features
<code>vect.transform(X_train)</code>	Convert to doc-term matrix

Entity Recognition (Chunking/Chinking)

<code>g="NP: {<D T>? <JJ >* < NN> }"</code>	Regex chunk grammar
<code>cp=nltk.RegexpParser(g)</code>	Parse grammar
<code>ch=cp.parse(pos_tagged_sents)</code>	Parse tagged sent. using grammar
<code>print(ch)</code>	Show chunks
<code>ch.draw()</code>	Show chunks in IOB tree
<code>cp.evaluate(test_sents)</code>	Evaluate against test docs
<code>sents= nltk.corpus.treebank.tagged_sents()</code>	
<code>print(nltk.ne_chunk(sents))</code>	Print chunk tree



RegEx with Pandas & Named Groups

```
df=pd.DataFrame({'time_sents', columns=['text']})

df['text'].str.split().str.len()

df['text'].str.contains('word')

df['text'].str.count(r'\d')

df['text'].str.findall(r'\d')

df['text'].str.replace(r'\w+day\b', '???')

df['text'].str.replace(r'(\w)', lambda x:
x.group(0)[:3])

df['text'].str.extract(r'(\d? \d): (\d -
\d)')

df['text'].str.extractall(r'((\d? \d) :
(\d\d) ?([ap] m))')

df['text'].str.extractall(r'(?P<digits> \d)')
```



By **RJ Murray** (murenei)
[cheatography.com/murenei/
tutify.com.au](https://cheatography.com/murenei/tutify.com.au)

Published 28th May, 2018.
Last updated 29th May, 2018.
Page 2 of 2.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>