Dive Into NLTK

Tutorial Source Code:

https://github.com/ridzuan05/NLTK-Tutorial

NLTK is a platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning.

http://nltk.org/nltk_data/

Part I: Getting Started with NLTK (Platform: Anaconda Python with Jupyter Notebook)

```
# install NLTK package using PC command prompt
```

conda install -c anaconda nltk

install using pip (other option)

pip install nltk

import NLTK using Jupyter Notebook

import nltk

update and install specific corpora/models

nltk.download()

import specific corpora

from nltk.corpus import brown
brown.tagged_words()[0:10]

```
[(u'The', u'AT'),
 (u'Fulton', u'NP-TL'),
 (u'County', u'NN-TL'),
 (u'Grand', u'JJ-TL'),
 (u'Jury', u'NN-TL'),
 (u'said', u'VBD'),
 (u'Friday', u'NR'),
 (u'an', u'AT'),
 (u'investigation', u'NN'),
 (u'of', u'IN')]
```

Part II: Sentence Tokenize and Word Tokenize

Tokenizers is used to divide strings into lists of substrings. For example, Sentence tokenizer can be used to find the list of sentences and Word tokenizer can be used to find the list of words in strings.

Tag	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun

Tag	Description
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Whdeterminer
WP	Whpronoun
WP$	Possessive whpronoun
WRB	Whadverb

```
# import nltk models

from nltk import sent_tokenize, word_tokenize, pos_tag

# output number of sentences

sents = sent_tokenize(text)

len(sents)
```

sent_tokenize uses an instance of PunktSentenceTokenizer from the nltk. tokenize.punkt module. It works well for many European languages. So it knows what punctuation and characters mark the end of a sentence and the beginning of a new sentence.

```
# output number of words

tokens = word_tokenize(text)

len(tokens)
```

Except the TreebankWordTokenizer, there are other alternative word tokenizers, such as PunktWordTokenizer and WordPunktTokenizer.

Part III: Part-Of-Speech Tagging and POS Tagger

Part-of-speech tagging is one of the most important text analysis tasks used to classify words into their part-of-speech and label them according the tagset which is a collection of tags used for the pos tagging.

```
# text pos tagging

from nltk import pos_tag

tagged = nltk.pos_tag(tokens)

tagged
```

Filtering stopwords in a tokenized sentence

Stopwords are common words that generally do not contribute to the meaning of a sentence, at least for the purposes of information retrieval and natural language processing.

```
# Import the stop word list

from nltk.corpus import stopwords

print stopwords.words("english")

# Remove stop words from "words"

tokens = [w for w in tokens if not w in stopwords.words("english")]

# list all stopwords for other languages

stopwords.fileids()
```

Part IV: Stemming and Lemmatization

Stemming and Lemmatization are the basic text processing methods for English text. The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. Major issue is overstemming (managed become manag).. etc.

NLTK provides several famous stemmers interfaces, such as Porter stemmer, Lancaster Stemmer, Snowball Stemmer and etc.

```
from nltk.stem.porter import PorterStemmer

stemmer = PorterStemmer()

stemmer.stem("cooking")
```

The NLTK Lemmatization method is based on WordNet's built-in morphy function. WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept.

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

lemmatizer.lemmatize('cooking')

lemmatizer.lemmatize('cooking', pos='v')

from nltk.corpus import wordnet

syn = wordnet.synsets('cooking')
```

Part V: TextBlob NLP API

TextBlob is a Python (2 and 3) library for processing textual data. It provides a consistent API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, and more.

http://textblob.readthedocs.io

```
# install TextBlob package using PC command prompt

pip install textblob

# import TextBlob using Jupyter Notebook

from textblob import TextBlob
```

Tokenization

You can break TextBlobs into words or sentences.

```
text = TextBlob("Textblob is amazingly simple to use. What great fun!")

text.words

text.sentences
```

Part-of-speech Tagging

```
text.tags
```

Translation and Language Detection

TextBlobs can be translated between languages.

```
b = TextBlob(u"좋은 아침")

b.detect_language()
```

Translation

```
en_blob = TextBlob(u'Simple is better than complex.')

en_blob.translate(to='ko')
```

Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials.

The sentiment property returns a namedtuple of the form Sentiment(polarity, subjectivity). The polarity score is a float within the range [-1.0, 1.0]. The subjectivity is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective.

```
text.sentiment

testimonial.sentiment.polarity
```

Print sentiment score for each sentence

```
for sentence in text.sentences:
        print(sentence.sentiment.polarity)
```