

Dive Into NLTK, Part I: Getting Started with NLTK

<https://textminingonline.com/dive-into-nltk-part-i-getting-started-with-nltk>

Installing NLTK on windows

Windows ¶

These instructions assume that you do not already have Python installed on your machine.

32-bit binary installation

1. Install Python 3.6: <http://www.python.org/downloads/> (avoid the 64-bit versions)
2. Install Numpy (optional): <http://sourceforge.net/projects/numpy/files/NumPy/> (the version that specifies python3.5)
3. Install NLTK: <http://pypi.python.org/pypi/nltk>
4. Test installation: `start>Python35`, then type `import nltk`

在windows上首先先安裝python, Numpy, Nltk 都可以用下載的方式安裝

<https://www.nltk.org/install.html>

Installing NLTK on Mac/Unix

Install Setuptools: `http://pypi.python.org/pypi/setuptools`

Install Pip: `run sudo easy_install pip`

Install Numpy (optional): `run sudo pip install -U numpy`

Install PyYAML and NLTK: `run sudo pip install -U pyyaml nltk`

Test installation: `run python` then type `import nltk`

<https://textminingonline.com/dive-into-nltk-part-i-getting-started-with-nltk>

Import the whole module, or some specific attributes and functions

Python Import

When our program grows bigger, it is a good idea to break it into different modules.

A module is a file containing Python definitions and statements. [Python modules](#) have a filename and end with the extension `.py`.

Definitions inside a module can be imported to another module or the interactive interpreter in Python. We use the `import` keyword to do this.

For example, we can import the `math` module by typing in `import math`.

script.py IPython Shell

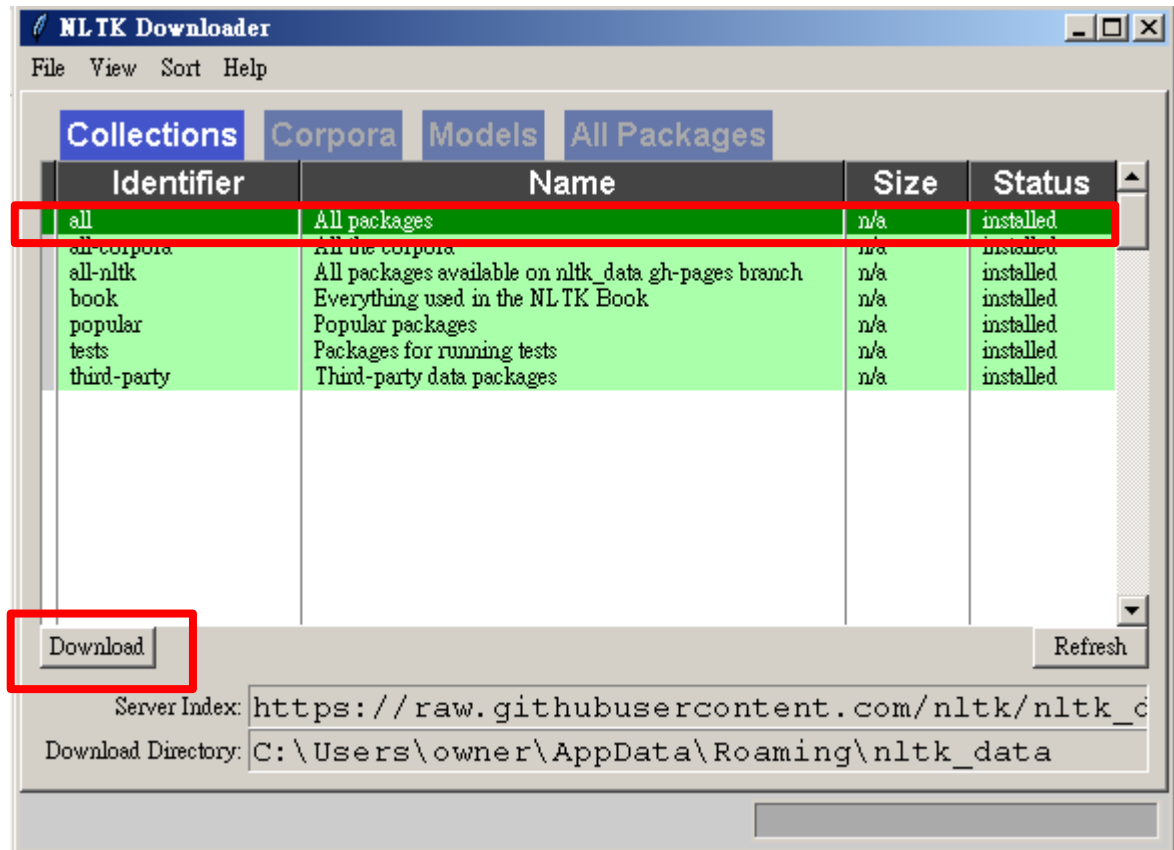
```
1 import math
2 print(math.pi)
```

Now all the definitions inside `math` module are available in our scope. We can also import some specific attributes and functions only, using the `from` keyword. For example:

```
>>> from math import pi
>>> pi
3.141592653589793
```

Installing NLTK Data

- `>>> import nltk`
- `>>> nltk.download()`



NLTK Corpora

NLTK has built-in support for dozens of corpora and trained models, as listed below. To use these within NLTK we recommend that you use the NLTK corpus downloader, >>> `nltk.download()`

Please consult the README file included with each corpus for further information.

1. *ACE Named Entity Chunker (Maximum entropy)* [[download](#)|[source](#)]

id: maxent_ne_chunker; size: 13404747; author: ; copyright: ; license: ;

2. *Australian Broadcasting Commission 2006* [[download](#)|[source](#)]

id: abc; size: 1487851; author: Australian Broadcasting Commission; copyright: ; license: ;

3. *Alpino Dutch Treebank* [[download](#)|[source](#)]

id: alpino; size: 2797255; author: ; copyright: ; license: Distributed with permission of Gertjan van Noord;

4. *BioCreative PE (Critical Assessment of Information Extraction Systems in Biology)* [[download](#)|[source](#)]

id: biocreative_ppi; size: 223566; author: ; copyright: Public Domain (not copyrighted); license: Public Domain;

5. *Brown Corpus* [[download](#)|[source](#)]

id: brown; size: 3314357; author: W. N. Francis and H. Kucera; copyright: ; license: May be used for non-commercial purposes.;

6. *Brown Corpus (TEI XML Version)* [[download](#)|[source](#)]

id: brown_tei; size: 8737738; author: W. N. Francis and H. Kucera; copyright: ; license: May be used for non-commercial purposes.;

http://www.nltk.org/nltk_data/

(1)Test NLTK - Brown Corpus

單詞，詞性，長度，...

```
>> from nltk.corpus import brown
```

```
>>> brown.words()[0:10]
```

```
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an',  
'investigation', 'of']
```

```
>>> brown.tagged_words()[0:10]
```

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

```
>>> len(brown.words())
```

```
1161192
```

```
>>> dir(brown)
```

```
['__class__', '__delattr__', '__dict__', '__doc__', '__format__',  
'__getattr__', '__hash__', '__init__', '__module__', '__new__', '__reduce__',  
'__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__',  
'__subclasshook__', '__weakref__', '_add', '_c2f', '_delimiter', '_encoding',  
'_f2c', '_file', '_fileids', '_get_root', '_init', '_map', '_para_block_reader',
```

(2) Test NLTK Book Resources

```
>>> from nltk.book import *
```

```
*** Introductory Examples for the NLTK Book ***
```

```
Loading text1, ..., text9 and sent1, ..., sent9
```

```
Type the name of the text or sentence to view it.
```

```
Type: 'texts()' or 'sents()' to list the materials.
```

```
text1: Moby Dick by Herman Melville 1851
```

```
text2: Sense and Sensibility by Jane Austen 1811 260819
```

```
text3: The Book of Genesis
```

```
text4: Inaugural Address Corpus
```

```
text5: Chat Corpus
```

```
text6: Monty Python and the Holy Grail
```

```
text7: Wall Street Journal
```

```
text8: Personals Corpus
```

```
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

```
>>> len(text1)
```

```
Sperm Whale; Moby Dick; White Whale; old man; Captain Ahab; sperm  
whale; Right Whale; Captain Peleg; New Bedford; Cape Horn; cried Ahab;  
years ago; lower jaw; never mind; Father Mapple; cried Stubb; chief  
mate; white whale; ivory leg; one hand
```

```
>>> dir(text1)
```

```
['_CONTEXT_RE', '_COPY_TOKENS', '__class__', '__delattr__', '__dict__', '__doc__',  
 '__format__', '__getattr__', '__getitem__', '__hash__', '__init__',  
 '__len__', '__module__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',  
 '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__',
```


(3-1) Sent Tokenize(sentence boundary detection, sentence segmentation), Word Tokenize and Pos Tagging

```
>>> from nltk import sent_tokenize, word_tokenize, pos_tag
```

```
>>> text = "Machine learning is the science of getting computers to act without being explicitly programmed. In the past decade, machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome. Machine learning is so pervasive today that you probably use it dozens of times a day without knowing it. Many researchers also think it is the best way to make progress towards human-level AI. In this class, you will learn about the most effective machine learning techniques, and gain practice implementing them and getting them to work for yourself. More importantly, you'll learn about not only the theoretical underpinnings of learning, but also gain the practical know-how needed to quickly and powerfully apply these techniques to new problems. Finally, you'll learn about some of Silicon Valley's best practices in innovation as it pertains to machine learning and AI."
```

```
>>> sents = sent_tokenize(text)
```

```
>>> sents
```

```
['Machine learning is the science of getting computers to act without being explicitly programmed.', 'In the past decade, machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome.', 'Machine learning is so pervasive
```

(3-2) Sent Tokenize(sentence boundary detection, sentence segmentation), Word Tokenize and Pos Tagging

```
>>> len(sents)
```

```
7
```

```
>>> tokens = word_tokenize(text)
```

```
>>> tokens
```

```
['Machine', 'learning', 'is', 'the', 'science', 'of', 'getting', 'computers',  
'to', 'act', 'without', 'being', 'explicitly', 'programmed.', 'In', 'the', 'past',  
'decade', ',', 'machine', 'learning', 'has', 'given', 'us', 'self-driving',  
'cars', ',', 'practical', 'speech', 'recognition', ',', 'effective', 'web',  
'search', ',', 'and', 'a', 'vastly', 'improved', 'understanding', 'of', 'the',  
'human', 'genome.', 'Machine', 'learning', 'is', 'so', 'pervasive', 'today',
```

```
>>> len(tokens)
```

```
161
```

```
>>> tagged_tokens = pos_tag(tokens)
```

```
>>> tagged_tokens
```

```
[('Machine', 'NN'), ('learning', 'NN'), ('is', 'VBZ'), ('the', 'DT'), ('science',  
'NN'), ('of', 'IN'), ('getting', 'VBG'), ('computers', 'NNS'), ('to', 'TO'),  
'act', 'VB'), ('without', 'IN'), ('being', 'VBG'), ('explicitly', 'RB'),  
'programmed.', 'NNP'), ('In', 'NNP'), ('the', 'DT'), ('past', 'JJ'), ('decade',  
'NN'), (',', ',', ','), ('machine', 'NN'), ('learning', 'NN'), ('has', 'VBZ'),
```

Dive Into NLTK, Part II: Sentence Tokenize and Word Tokenize

<https://textminingonline.com/dive-into-nltk-part-ii-sentence-tokenize-and-word-tokenize>

Tokenizers

- **Tokenizers** is used to divide strings into lists of substrings.
- **Sentence tokenizer** can be used to find the list of sentences
- **Word tokenizer** can be used to find the list of words in strings.

Tokenizing text into sentences

- Sentence Tokenize also known as **Sentence boundary disambiguation**, **Sentence boundary detection**, **Sentence segmentation**.
- Other NLP tools include the sentence tokenize function : OpenNLP , NLTK, TextBlob, MBSP, etc.

How to use sentence tokenize in NLTK?

After [installing nltk and nltk_data](#) , you can launch python and import sent_tokenize tool from nltk:

```
>>> text = "this's a sent tokenize test. this is sent two. is this sent three? sent 4 is cool! Now it's your turn."
```

```
>>> from nltk.tokenize import sent_tokenize
```

```
>>> sent_tokenize_list = sent_tokenize(text)
```

```
>>> len(sent_tokenize_list)
```

```
5
```

```
>>> sent_tokenize_list
```

```
['this's a sent tokenize test.', 'this is sent two.', 'is this sent three?', 'sent 4 is cool!', 'Now it's your turn.']
```

```
>>>
```

PunktSentenceTokenizer

`sent_tokenize` is one of instances of `PunktSentenceTokenizer` from the `nltk.tokenize.punkt` module.

There are total 17 european languages that NLTK support for sentence tokenize, and you can use them as the following steps:

```
>>> import nltk.data
>>> tokenizer = nltk.data.load('tokenizers/punkt/english.pickle')
>>> tokenizer.tokenize(text)
['this's a sent tokenize test.', 'this is sent two.', 'is this sent three?', 'sent 4 is cool!', "Now it's your turn."]
```

Here is a spanish sentence tokenize example:

```
>>> spanish_tokenizer = nltk.data.load('tokenizers/punkt/spanish.pickle')
>>> spanish_tokenizer.tokenize('Hola amigo. Estoy bien.')
['Hola amigo.', 'Estoy bien.']
>>>
```

Tokenizing text into words

Tokenizing text into words in NLTK is very simple, just called `word_tokenize` from `nltk.tokenize` module:

```
>>> from nltk.tokenize import word_tokenize
>>> word_tokenize('Hello World.')
['Hello', 'World', '.']
>>> word_tokenize("this's a test")
['this', "'s", 'a', 'test']
```

Actually, `word_tokenize` is a wrapper function that calls `tokenize` by the `TreebankWordTokenizer`, here is the code in NLTK:

```
# Standard word tokenizer.
_word_tokenize = TreebankWordTokenizer().tokenize
def word_tokenize(text):
    """
    Return a tokenized copy of *text*,
    using NLTK's recommended word tokenizer
    (currently :class:`TreebankWordTokenizer`).
    This tokenizer is designed to work on a sentence at a time.
    """
```


Alternative Word Tokenizers

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

`PunktTokenizer` splits on punctuation, but keeps it with the word:

```
>>> from nltk.tokenize import PunktWordTokenizer
>>> punkt_word_tokenizer = PunktWordTokenizer()
>>> punkt_word_tokenizer.tokenize("this's a test")
['this', "'s", 'a', 'test']
```

`WordPunktTokenizer` splits all punctuations into separate tokens:

```
>>> from nltk.tokenize import WordPunktTokenizer
>>> word_punct_tokenizer = WordPunktTokenizer()
>>> word_punct_tokenizer.tokenize("This's a test")
['This', "'", 's', 'a', 'test']
```


You can choose any word tokenizer in `nltk` for your using purpose.


Example - EVA Airlines discussion



Mark V
Puyallup, Washington


Level  Contributor

 152 reviews

 65 helpful votes


“Not all that great”

NEW


 Reviewed yesterday via mobile


I had heard EVA premium economy was like first class domestic. While you do get more space, the service is very economy. Service is very scripted, as as an English speaker you are a second class passenger. It seemed like we were ignored.


Traveled March 2018

 Seat comfort

 Legroom

 Customer service (e.g. attitude, care, helpfulness)

 In-flight entertainment (WiFi, TV, movies)

 Value for money

International

Premium Economy

Seattle - Macau

Helpful?

 Thank Mark V

 Report

[Ask Mark V about EVA Air](#)

This review is the subjective opinion of a TripAdvisor member and not of TripAdvisor LLC.

Example - EVA Airlines discussion

Tokenizing text into sentences

In [4]:

```
1 sentence_dicussion = 'I had heard EVA premium economy was like first class domestic. While you do get more space, the service  
2  
3 print(sentence_dicussion, "\n")  
4 print(sent_tokenize(sentence_dicussion), "\n")
```

I had heard EVA premium economy was like first class domestic. While you do get more space, the service is very economy. Service is very scripted, as as an English speaker you are a second class passenger. It seemed like we were ignored.

```
['I had heard EVA premium economy was like first class domestic.', 'While you do get more space, the service is very economy.', 'Service is very scripted, as as an English speaker you are a second class passenger.', 'It seemed like we were ignored.']
```

Example - EVA Airlines discussion

Tokenizing text into words

```
In [6]: 1 word_dicussion = 'I had heard EVA premium economy was like first class domestic.'  
        2  
        3 print(word_tokenize(word_dicussion))
```

```
['I', 'had', 'heard', 'EVA', 'premium', 'economy', 'was', 'like', 'first', 'class', 'domestic', '.']
```

Dive Into NLTK, Part III: Part-Of-Speech Tagging and POS Tagger

<https://textminingonline.com/dive-into-nltk-part-iii-part-of-speech-tagging-and-pos-tagger>

Part-of-speech tagging-1

- one of the most important text analysis tasks
- classify words into their part-of-speech and label them according the tagset (collection of tags).
- Part-of-speech tagging also known as word classes or lexical categories.

Part-of-speech tagging-2

- It is a term in In corpus linguistics
- also called **grammatical tagging or word-category disambiguation**
- is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech,
- **based on both its definition, as well as its context**—i.e. relationship with adjacent and related words in a phrase, sentence, or paragraph.
- POS-tagging algorithms fall into two distinctive groups: **rule-based** and **stochastic**.

How to use POS Tagging in NLTK

```
>>> import nltk
>>> text = nltk.word_tokenize("Dive into NLTK: Part-of-speech tagging and POS Tagger")
>>> text
['Dive', 'into', 'NLTK', ':', 'Part-of-speech', 'tagging', 'and', 'POS', 'Tagger']
>>> nltk.pos_tag(text)
[('Dive', 'JJ'), ('into', 'IN'), ('NLTK', 'NNP'), (':', ':'), ('Part-of-speech', 'JJ'), ('tagging', 'NN'), ('and', 'CC'), ('POS', 'NNP'), ('Tagger', 'NNP')]
```

NLTK batch pos tagging method

```
>>> nltk.batch_pos_tag([['this', 'is', 'batch', 'tag', 'test'], ['nltk', 'is', 'text', 'analysis', 'tool']])
[('this', 'DT'), ('is', 'VBZ'), ('batch', 'NN'), ('tag', 'NN'), ('test', 'NN'), [('nltk', 'NN'), ('is', 'VBZ'), ('text', 'JJ'), ('analysis', 'NN'), ('tool', 'NN')]]
>>>
```


* **warning:** 要先做word_tokenize 再做 pos_tag , 否則會直接對單字進行詞性分析

```
In [10]: # 直接pos_tag
test_text = "Dive into NLTK: Part-of-speech tagging and POS Tagger"
nlk.pos_tag(test_text)
```

```
Out[10]: [('D', 'NNP'),
          ('i', 'NN'),
          ('v', 'VBP'),
          ('e', 'NN'),
          (' ', 'NN'),
          ('i', 'NN'),
          ('n', 'VBP'),
          ('t', 'NN'),
          ('o', 'NN'),
          (' ', 'NNP'),
          ('N', 'NNP'),
          ('L', 'NNP'),
          ('T', 'NNP'),
          ('K', 'NNP'),
          (':', ':'),
          (' ', 'NN'),
          ('P', 'NNP'),
          ('a', 'DT'),
          ('r', 'NN'),
          ('t', 'SYM')]
```

各個詞性說明文檔 nltk.help.upenn_tagset(...)

```
# tag的說明文檔
print(nltk.help.upenn_tagset("RB"))
print("=====")
print(nltk.help.upenn_tagset("NN.*"))
```

RB: adverb

occasionally unabatingly maddeningly adventurously professedly
stirringly prominently technologically magisterially predominately
swiftly fiscally pitilessly ...

None

=====
NN: noun, common, singular or mass

common-carrier cabbage knuckle-duster Casino afghan shed thermostat
investment slide humour falloff slick wind hyena override subhumanity
machinist ...

NNP: noun, proper, singular

Motown Venneboerger Czestochwa Ranzer Conchita Trumplane Christos
Oceanside Escobar Kreisler Sawyer Cougar Yvette Ervin ODI Darryl CTCA
Shannon A.K.C. Meltex Liverpool ...

NNPS: noun, proper, plural

Americans Americas Amharas Amityvilles Amusements Anarcho-Syndicalists
Andalusians Andes Andruses Angels Animals Anthony Antilles Antiques
Apache Apaches Apocrypha ...

NNS: noun, common, plural

undergraduates scotches bric-a-brac products bodyguards facets coasts
divestitures storehouses designs clubs fragrances averages
subjectivists apprehensions muses factory-jobs ...

NLTK documentation for POS Tagging

```
>>> nltk.help.upenn_tagset('JJ')
```

```
JJ: adjective or numeral, ordinal
```

```
third ill-mannered pre-war regrettable oiled calamitous first separable  
ectoplasmic battery-powered participatory fourth still-to-be-named  
multilingual multi-disciplinary ...
```

```
>>> nltk.help.upenn_tagset('IN')
```

```
IN: preposition or conjunction, subordinating
```

```
astride among upon whether out inside pro despite on by throughout  
below within for towards near behind atop around if like until below  
next into if beside ...
```

```
>>> nltk.help.upenn_tagset('NNP')
```

```
NNP: noun, proper, singular
```

```
Motown Venneboerger Czestochwa Ranzer Conchita Trumplane Christos  
Oceanside Escobar Kreisler Sawyer Cougar Yvette Ervin ODI Darryl CTCA  
Shannon A.K.C. Meltex Liverpool ...
```

```
>>>
```

The pre-trained POS Tagger Model in NLTK

You can find the pre-trained POS Tagging Model in `nltk_data/taggers`:

```
YangtekiMacBook-Pro:taggers textminer$ pwd
/Users/textminer/nltk_data/taggers
YangtekiMacBook-Pro:taggers textminer$ ls
total 11304
drwxr-xr-x  3 textminer  staff  102 7  9  14:06 hmm_treebank_pos_tagger
-rw-r--r-   1 textminer  staff 750857 5 26 2013 hmm_treebank_pos_tagger.zip
drwxr-xr-x  3 textminer  staff  102 7 24 2013 maxent_treebank_pos_tagger
-rw-r--r-   1 textminer  staff 5031883 5 26 2013 maxent_treebank_pos_tagger.zip
```

How to train a POS Tagging Model or POS Tagger in NLTK

- used the **maxent treebank pos tagging model in NLTK by default**
- NLTK provides not only the maxent pos tagger,
- other pos taggers like crf, hmm, brill, **tnt** and interfaces with stanford pos tagger, hunpos pos tagger and senna postaggers
- Next show **how to train a TnT POS Tagger Model**

train data and test data

First you need the train data and test data, we use the treebank data from nltk.corpus:

```
>>> from nltk.corpus import treebank
>>> len(treebank.tagged_sents())  first 3000 treebank tagged sentences as the train_data
3914                               last 914 tagged sentences as the test_data,
>>> train_data = treebank.tagged_sents()[:3000]
>>> test_data = treebank.tagged_sents()[3000:]
>>> train_data[0]
[(u'Pierre', u'NNP'), (u'Vinken', u'NNP'), (u',', u','), (u'61', u'CD'), (u'years', u'NNS'), (u'old',
u'JJ'), (u',', u','), (u'will', u'MD'), (u'join', u'VB'), (u'the', u'DT'), (u'board', u'NN'), (u'as', u'IN'),
(u'a', u'DT'), (u'nonexecutive', u'JJ'), (u'director', u'NN'), (u'Nov.', u'NNP'), (u'29', u'CD'), (u',',
u'.')]
>>> test_data[0]
[(u'At', u'IN'), (u'Tokyo', u'NNP'), (u',', u','), (u'the', u'DT'), (u'Nikkei', u'NNP'), (u'index', u'NN'),
(u'of', u'IN'), (u'225', u'CD'), (u'selected', u'VBN'), (u'issues', u'NNS'), (u',', u','), (u'which',
u'WDT'), (u'*T*-1', u'-NONE-'), (u'gained', u'VBD'), (u'132', u'CD'), (u'points', u'NNS'),
(u'Tuesday', u'NNP'), (u',', u','), (u'added', u'VBD'), (u'14.99', u'CD'), (u'points', u'NNS'), (u'to',
u'TO'), (u'35564.43', u'CD'), (u',', u'.')]
>>>
```

train TnT POS Tagger and evaluate it

```
>>> from nltk.tag import tnt
>>> tnt_pos_tagger = tnt.TnT()
>>> tnt_pos_tagger.train(train_data)
>>> tnt_pos_tagger.evaluate(test_data)
0.8755881718109216
```

```
from nltk.tag import tnt
# tnt: 詞性分類

tnt_pos_tagger = tnt.TnT()
```

```
# train
tnt_pos_tagger.train(train_data)
# evaluate
tnt_pos_tagger.evaluate(test_data) # 可以用自己的資料進行testing
```

0.875545003237643 精準度

save this pos tagger model as a pickle file

```
>>> import pickle
>>> f = open('tnt_treebank_pos_tagger.pickle', 'w')
>>> pickle.dump(tnt_pos_tagger, f)
>>> f.close()
```

線上資料可能要修改一下

```
import pickle
```

pickle須以binary方式寫入

```
# save model
f = open("tnt_treebank_pos_tagger.pickle", "wb") # 二進位輸入
pickle.dump(tnt_pos_tagger, f)
f.close()
           model name
```


use the model at any time

```
>>> tnt_tagger.tag(nltk.word_tokenize("this is a tnt treebank tnt tagger"))  
[('this', u'DT'), ('is', u'VBZ'), ('a', u'DT'), ('tnt', 'Unk'), ('treebank', 'Unk'), ('tnt', 'Unk'), ('tagger',  
'Unk')]  
>>>
```

Do you want to train your POS Tagging Model?



alisonj217
西班牙

👍10 🗨️2

🌟🌟🌟🌟🌟 評論過2018年1月27日

不可思議的美麗地方

從我們完美的泳池露台別墅可以看到極好的熱帶雨林風景。這裡有完美且友善的專業服務、華麗的食物。我們在這裏住了12晚，但依然沒有時間去嘗試完所有菜單上的選擇。你眼睛所到的地方都是很漂亮的。離市區有點遠，但是有免費贈送的出租車服務，烏布市有兩個上車點。如果你難以決定住在哪裡，不要錯過這裏！

👍 感謝 alisonj217

```
comment = '''Fabulous rain forest view from our immaculate pool terrace villa.  
Perfect friendly professional service. Gorgeous food.  
We were here for 12 nights but still had no time to get through all the menu choices.  
Every where your eye rests it is beautiful.  
A little way from town but there is a complimentary taxi  
service with 2 pickup points in Ubud.  
if you're having trouble deciding where to stay,  
look no further!  
'''
```

```
# use model: tnt_pos_tagger  
tnt_pos_tagger.tag(word_tokenize(comment))  
  
# use pos_tag  
pos_tag(word_tokenize(comment))
```

* 使用model: tnt_pos_tagger來對comment分詞性

* 使用nltk的pos_tag來對comment分詞性