Text Mining Online | Text Analysis Online | Text Processing Online

Text Mining | Text Analysis | Text Process | Natural Language Processing

Dive Into NLTK, Part IV: Stemming and Lemmatization

This is the fourth article in the series "Dive Into NLTK", here is an index of all the articles in the series that have been published to date:

Part I: Getting Started with NLTK
Part II: Sentence Tokenize and Word Tokenize
Part III: Part-Of-Speech Tagging and POS Tagger
Part IV: Stemming and Lemmatization (this article)
Part V: Using Stanford Text Analysis Tools in Python
Part VI: Add Stanford Word Segmenter Interface for Python NLTK
Part VII: A Preliminary Study on Text Classification
Part VIII: Using External Maximum Entropy Modeling Libraries for Text Classification

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. Here is the definition from wikipedia for stemming and lemmatization:

Stemming:

> *In linguistic morphology and information retrieval, stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form—generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. Algorithms for stemming have been studied in computer science since the 1960s. Many search engines treat words with the same stem as synonyms as a kind of query expansion, a process called conflation.*

Stemming programs are commonly referred to as stemming algorithms or stemmers.

Lemmatization:

Lemmatisation (or lemmatization) in linguistics, is the process of grouping together the different inflected forms of a word so they can be analysed as a single item.

In computational linguistics, lemmatisation is the algorithmic process of determining the lemma for a given word. Since the process may involve complex tasks such as understanding context and determining the part of speech of a word in a sentence (requiring, for example, knowledge of the grammar of a language) it can be a hard task to implement a lemmatiser for a new language.

In many languages, words appear in several inflected forms. For example, in English, the verb 'to walk' may appear as 'walk', 'walked', 'walks', 'walking'. The base form, 'walk', that one might look up in a dictionary, is called the lemma for the word. The combination of the base form with the part of speech is often called the lexeme of the word.

Lemmatisation is closely related to stemming. The difference is that a stemmer operates on a single word without knowledge of the context, and therefore cannot discriminate between words which have different meanings depending on part of speech. However, stemmers are typically easier to implement and run faster, and the reduced accuracy may not matter for some applications.

How to use Stemmer in NLTK

NLTK provides several famous stemmers interfaces, such as Porter stemmer, Lancaster Stemmer, Snowball Stemmer and etc. In NLTK, using those stemmers is very simple.

For Porter Stemmer, which is based on The Porter Stemming Algorithm, can be used like this:

```
>>> from nltk.stem.porter import PorterStemmer
>>> porter_stemmer = PorterStemmer()
>>> porter_stemmer.stem('maximum')
u'maximum'
>>> porter_stemmer.stem('presumably')
u'presum'
>>> porter_stemmer.stem('multiply')
u'multipli'
```

```
>>> porter_stemmer.stem('provision')
u'provis'
>>> porter_stemmer.stem('owed')
u'owe'
>>> porter_stemmer.stem('ear')
u'ear'
>>> porter_stemmer.stem('saying')
u'say'
>>> porter_stemmer.stem('crying')
u'cri'
>>> porter_stemmer.stem('string')
u'string'
>>> porter_stemmer.stem('meant')
u'meant'
>>> porter_stemmer.stem('cement')
u'cement'
>>>
```

For Lancaster Stemmer, which is based on The Lancaster Stemming Algorithm, can be used in NLTK like this:

```
>>> from nltk.stem.lancaster import LancasterStemmer
>>> lancaster_stemmer = LancasterStemmer()
>>> lancaster_stemmer.stem('maximum')
'maxim'
>>> lancaster_stemmer.stem('presumably')
'presum'
>>> lancaster_stemmer.stem('presumably')
'presum'
>>> lancaster_stemmer.stem('multiply')
'multiply'
>>> lancaster_stemmer.stem('provision')
u'provid'
>>> lancaster_stemmer.stem('owed')
'ow'
>>> lancaster_stemmer.stem('ear')
'ear'
>>> lancaster_stemmer.stem('saying')
'say'
>>> lancaster_stemmer.stem('crying')
'cry'
```

```
>>> lancaster_stemmer.stem('string')
'string'
>>> lancaster_stemmer.stem('meant')
'meant'
>>> lancaster_stemmer.stem('cement')
'cem'
>>>
```

For Snowball Stemmer, which is based on Snowball Stemming Algorithm, can be used in NLTK like this:

```
>>> from nltk.stem import SnowballStemmer
>>> snowball_stemmer = SnowballStemmer("english")
>>> snowball_stemmer.stem('maximum')
u'maximum'
>>> snowball_stemmer.stem('presumably')
u'presum'
>>> snowball_stemmer.stem('multiply')
u'multipli'
>>> snowball_stemmer.stem('provision')
u'provis'
>>> snowball_stemmer.stem('owed')
u'owe'
>>> snowball_stemmer.stem('ear')
u'ear'
>>> snowball_stemmer.stem('saying')
u'say'
>>> snowball_stemmer.stem('crying')
u'cri'
>>> snowball_stemmer.stem('string')
u'string'
>>> snowball_stemmer.stem('meant')
u'meant'
>>> snowball_stemmer.stem('cement')
u'cement'
>>>
```

How to use Lemmatizer in NLTK

The NLTK Lemmatization method is based on WordNet's built-in morphy function. Here is the introduction from WordNet official website:

> *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. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the browser. WordNet is also freely and publicly available for download. WordNet's structure makes it a useful tool for computational linguistics and natural language processing.*
>
> *WordNet superficially resembles a thesaurus, in that it groups words together based on their meanings. However, there are some important distinctions. First, WordNet interlinks not just word forms—strings of letters—but specific senses of words. As a result, words that are found in close proximity to one another in the network are semantically disambiguated. Second, WordNet labels the semantic relations among words, whereas the groupings of words in a thesaurus does not follow any explicit pattern other than meaning similarity*

In NLTK, you can use it as the following:

```
>>> from nltk.stem import WordNetLemmatizer
>>> wordnet_lemmatizer = WordNetLemmatizer()
>>> wordnet_lemmatizer.lemmatize('dogs')
u'dog'
>>> wordnet_lemmatizer.lemmatize('churches')
u'church'
>>> wordnet_lemmatizer.lemmatize('aardwolves')
u'aardwolf'
>>> wordnet_lemmatizer.lemmatize('abaci')
u'abacus'
>>> wordnet_lemmatizer.lemmatize('hardrock')
'hardrock'
>>> wordnet_lemmatizer.lemmatize('are')
'are'
>>> wordnet_lemmatizer.lemmatize('is')
'is'
```

You would note that the "are" and "is" lemmatize results are not "be", that's because the lemmatize method default pos argument is "n":

```
lemmatize(word, pos='n')
```

So you need specified the pos for the word like these:

```
>>> wordnet_lemmatizer.lemmatize('is', pos='v')
u'be'
>>> wordnet_lemmatizer.lemmatize('are', pos='v')
u'be'
>>>
```

We have use POS Tagging before word lemmatization, and implemented it in our Text Analysis API, you can test and use it without specified pos tagger by our Text Analysis API.

The Stem Module in NLTK

You can find the stem module in nltk-master/nltk/stem

```
YangtekiMacBook-Pro:stem textminer$ ls
total 456
-rwxr-xr-x@ 1 textminer staff 1270 7 22 2013 __init__.py
-rwxr-xr-x@ 1 textminer staff 798 7 22 2013 api.py
-rwxr-xr-x@ 1 textminer staff 17068 7 22 2013 isri.py
-rwxr-xr-x@ 1 textminer staff 11337 7 22 2013 lancaster.py
-rwxr-xr-x@ 1 textminer staff 24735 7 22 2013 porter.py
-rwxr-xr-x@ 1 textminer staff 1701 7 22 2013 regexp.py
-rwxr-xr-x@ 1 textminer staff 5563 7 22 2013 rslp.py
-rwxr-xr-x@ 1 textminer staff 146857 7 22 2013 snowball.py
-rwxr-xr-x@ 1 textminer staff 1513 7 22 2013 wordnet.py
```

```
# Natural Language Toolkit: Stemmers
#
# Copyright (C) 2001-2013 NLTK Project
# Author: Trevor Cohn <tacohn@cs.mu.oz.au>
#         Edward Loper <edloper@gradient.cis.upenn.edu>
#         Steven Bird <stevenbird1@gmail.com>
# URL: <http://www.nltk.org/>
# For license information, see LICENSE.TXT

"""
NLTK Stemmers

Interfaces used to remove morphological affixes from words, leaving
only the word stem.  Stemming algorithms aim to remove those affixes
required for eg. grammatical role, tense, derivational morphology
leaving only the stem of the word.  This is a difficult problem due to
irregular words (eg. common verbs in English), complicated
morphological rules, and part-of-speech and sense ambiguities
(eg. ``ceil-`` is not the stem of ``ceiling``).

```

```
21 StemmerI defines a standard interface for stemmers.
22 """
23
24 from nltk.stem.api import StemmerI
25 from nltk.stem.regexp import RegexpStemmer
26 from nltk.stem.lancaster import LancasterStemmer
27 from nltk.stem.isri import ISRIStemmer
28 from nltk.stem.porter import PorterStemmer
29 from nltk.stem.snowball import SnowballStemmer
30 from nltk.stem.wordnet import WordNetLemmatizer
31 from nltk.stem.rslp import RSLPStemmer
32
33
34 if __name__ == "__main__":
35     import doctest
36     doctest.testmod(optionflags=doctest.NORMALIZE_WHITESPACE)
```

Read the code, and change the World! Now it's your time!

Posted by TextMiner

Related Posts:

1. **We have launched the Text Analysis API on Mashape**
2. **Dive Into NLTK, Part I: Getting Started with NLTK**
3. **Dive Into NLTK, Part II: Sentence Tokenize and Word Tokenize**
4. **Dive Into NLTK, Part VII: A Preliminary Study on Text Classification**

This entry was posted in NLTK, Text Analysis, Text Mining, Uncategorized and tagged Lancaster Stemmer, Lemmatization, Lemmatization API, lemmatize, Lemmatizer, NLTK, Porter Stemmer, Snowball Stemmer, Stem, Stemmer, Stemming, Stemming API, Word Lemmatization API, word lemmatize, Word Stem, Word Stemming API, WordNet Lemmatization, WordNet Lemmatizer on July 18, 2014 [http://textminingonline.com/dive-into-nltk-part-iv-stemming-and-lemmatization] .