NLTK Module

October 25, 2010

1 Module name

Natural Language Toolkit (NLTK)

2 Scope

A teaching and studying platform for prototyping and building research systems on natural language processing (NLP), related to linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning.

3 Learning objectives

These are the learning objectives for this module:

a. Be confident in the information retrieval related features of NLTK. These include the simple as well as some of the more advanced features.

b. Understand how NLTK relates to Python and be able to apply Python to utilize NLTK.

NLTK is a very huge package. We do not go very deep, to analyze sentences and try to understand information and concepts from text.

4 5S characteristics of the module

- Stream: NLTK handles different kinds of strings to represent information in different kinds of languages as input, and output analysis results of them in various formats.
- Structure: NLTK is a suite of libraries for the Python programming language. It makes use of data structures supported in Python such as arrays.
- Space: Since NLTK is a collection of language analysis tools, the spaces of it can be categorized into data spaces, which means information from different languages, and analysis space, which is the storage cost for analyzing it.
- Scenario: NLTK can be used to analyze the structure of words, sentences, and articles.
- Society: NLTK is a good tool for both researchers to research on natural language processing and for students to learn the knowledge of the processes.

5 Relationships with other modules

- Lemur: We can use the statistical functions inside NLTK to build simple language models. Lemur is a more mature application of such methods.
- PF/Tijah & SOLR: NLTK can do many basic tasks such as tokenization to support the more comprehensive search systems, such as SOLR and PF/Tijah.
- WordNet: WordNet is built into NLTK as a part of its word analysis. One can use Python with NLTK to e.g. a hypernym of a word quickly.
- WEKA: With WEKA, one can process some data mining tasks such as Naive Bayes classification which is built in NLTK.

6 Prerequisite knowledge/skills required

- Python: Since NLTK is a suite of Python libraries, Python knowledge is a prerequisite. Getting started with Python is simple. Some advanced usages such as class and exceptions are not needed in order to get familiar with NLTK.
  - Python: Basic concepts about Python programming language is required.
    - Block style
    - List
    - Dictionary
    - Function
    - Handling Stream
  - Some advanced usage: In order to improving our programming speed, and also vital to understanding examples from others.
    - Map, Reduce, Zip
    - Enumerate
    - List comprehension
    - Regular Expression (RE)
- Linux basis:
  - Script ABC: One needs to know how to write scripts and run them in Linux to make it easier to write Python programs. Instructions will be given in exercise part.

7 Introductory remedial instruction

None

8 Body of knowledge

NLTK (Natural Language ToolKit) is a Python package for natural language processing. The following subsections consist of exercises with NLTK which relate to chapters 1, 2, 3 and 13 of the book "An Introduction to Information Retrieval" by Manning et al.

8.1 Getting started with Python

We have set up the NLTK environment on the IBM cloud. Please access the server via ssh. When you log onto the server, just type *python* to start a Python interactive shell. To import the NLKT package, enter *import nltk*. This ensures that the NLTK package is ready to use. Samples of text should already be installed; we will use these in the following exercises. Type: *from nltk.book import ** to load the sample texts.

8.1.1 Python basics

You may wish to write short scripts as an alternative to using the interactive Python shell. Put it your scripts in the directory named with team number. You can choose the file name as you wish.

Here is a template for a Python script:

```
#!/usr/bin/python
print 'hello, world!'
```

You need to *chmod* to *755* to make it executable. For example, you save the file as *hello.py*.

```
$chmod 755 hello.py
```

Then run it:

```
$./hello.py
hello, world
$
```

Python is an interpreted language. The interactive Python shell is started by typing `python` from the commandline. The Python shell will evaluate every individual line you type and print the return value.

- Data structures The most basic Python structures are *lists* and *dictionaries*. Lists are ordered, indexable, dynamic arrays of elements of varying type. Dictionaries are key-value containers, also referred to as associative arrays etc. The following demonstrates the basic operations on the mentioned datatypes:

```
>>> list1 = ['a','b','c']
>>> list1[1]
'b'
>>> list1[3]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>> list1[:2] # return until index 2
['a', 'b']
>>> list1[2:] # return starting at index 2
['c']
>>> list1[1:2] # start at index 1, end at 2
['b']
>>> dict1 = {'a' : 1, 'b': 2, 'c' : 3}
>>> dict1['a']
1
>>> dict1['d']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'd'
>>> dict1.keys()
['a', 'c', 'b']
```

- Control flow

  Two important looping constructs in Python are the *for* and the *while* constructs. Here are examples:

```
>>> list1
['a', 'b', 'c']
>>> while len(list1) > 0:
...     print list1.pop()
...
c
b
a
>>>
```

  Two interesting notes:

  – There are no parentheses around the looping expression. The colon ':' marks the beginning of the loop body.

  – Blocks are denoted by indentation (tab character). There is no other explicit block notation.

  The following shows the *for* loop:

```
>>> list1 = ['a','b','c']
>>> for x in list1:
...     print x
...
a
b
c
>>>
```

  The following example shows how the *if* construct appears in Python:[1]

```
>>> x = int(raw_input("Please enter an integer: "))
Please enter an integer: 42
>>> if x < 0:
...      x = 0
...      print 'Negative changed to zero'
... elif x == 0:
...      print 'Zero'
... elif x == 1:
...      print 'Single'
... else:
...      print 'More'

>>> list1
['a', 'b', 'c']
>>> while len(list1) > 0:
...     print list1.pop()
...
```

[1] http://docs.python.org/tutorial/controlflow.html

```
c
b
a
>>>
```

Finally, Python has a list comprehension construct which will seem familiar to common set notation used in math: $S = \{2x | x \textbf{ in } \{0...9\}\}$

In Python, this is expressed in the following manner:

```
S = [ x*2 for x in range(0,9)]
```

Please see http://docs.python.org/tutorial/introduction.html for more information on Python.

8.2 NLTK features

- Getting Started With NLTK
  - Fetching Books From NLTK Core
    http://nltk.googlecode.com/svn/trunk/doc/book/ch01.html - "Getting Started with NLTK"
  - Basic Searching on Text
    An Introduction to Information Retrieval: Chapter 2
    http://nltk.googlecode.com/svn/trunk/doc/book/ch01.html - "Searching Text"
    - Sentence Segmentation
    - Tokenization
    - Normalizing Text
  - Counting Vocabulary
    An Introduction to Information Retrieval: Chapter 6.2
    http://nltk.googlecode.com/svn/trunk/doc/book/ch01.html - "Frequency Distributions"
    - Frequency Distributions
  - Automatic Natural Language Understanding
    http://nltk.googlecode.com/svn/trunk/doc/book/ch01.html - "1.5 Automatic Natural Language Understanding"
- Lexical Resources
  http://nltk.googlecode.com/svn/trunk/doc/book/ch02.html - "2 Accessing Text Corpora and Lexical Resources"
  - Accessing Text Corpora
- Categorizing and Tagging Words
  An Introduction to Information Retrieval: Chapter 2.4.1
  http://nltk.googlecode.com/svn/trunk/doc/book/ch06.html - "Part-of-Speech Tagging"
  - POS-tagger in NLTK
  - Tagged Corpora
    - Nouns, Verbs, Adjectives and Adverbs
    - Unsimplified Tags
    - Exploring Tagged Corpora
  - The Regular Expression Tagger
  - The Lookup Tagger

  - N-Gram Tagging
  - Transformation-Based Tagging

- Classification with NLTK
  An Introduction to Information Retrieval: Chapter 13
  http://nltk.googlecode.com/svn/trunk/doc/book/ch06.html - "6 Learning to Classify Text"
  - Supervised Classification
    * Feature Extractor
    * Data Set
    * Improvement and Choosing Features
  - Document Classification
  - Part-of-Speech Tagging
  - Sequence Classification
  - Evaluation
    * Accuracy
    * Precision and Recall
    * Confusion Matrices
    * Cross-Validation
  - Decision Trees
  - Naive Bayes Classifiers
  - Maximum Entropy Classifiers

9 Resources

- Natural Language Processing with Python
  by Steven Bird, Ewan Klein, and Edward Loper - O'Reilly Media, 2009
  http://www.nltk.org/book
- Official Python website: Introduction to Python http://docs.python.org/tutorial/introduction.html
- Auto-generated NLTK API Documentation http://nltk.googlecode.com/svn/trunk/doc/api/index.html

10 Exercises / Learning activities

Be sure to provide answers to every question appearing anywhere below, and to follow the instructions given, in order.

Please use the NLTK book for additional instructions on how to use NLTK: http://www.nltk.org/book Specifially, chapter 6 of the NLTK book will be instrumental in solving Section 10.3 of these exercises. Additionally, the API documentation may be needed: http://nltk.googlecode.com/svn/trunk/doc/api/index.html

10.1 Introductory Exercises

10.1.1 Frequency distribution

```
>>> fdist1 = FreqDist(text1)
>>> fdist1
<FreqDist with 260819 outcomes>
```

What does "260819 outcomes" mean?

```
>>> vocabulary1 = fdist1.keys()
>>> vocabulary1[:50]
[',', 'the', '.', 'of', 'and', 'a', 'to', ';', 'in', 'that', "'", '-', 'his', 'it',
 'I', 's', 'is', 'he', 'with', 'was', 'as', '"', 'all', 'for', 'this', '!', 'at', 'by',
 'but', 'not', '--', 'him', 'from', 'be', 'on', 'so', 'whale', 'one', 'you', 'had',
 'have', 'there', 'But', 'or', 'were', 'now', 'which', '?', 'me', 'like']
```

`vocabulary1` is a list of elements. `vocabulary1[:50]` returns the first 50 elements of this list. What does it mean that ',' is first in the list? Which element is the last one in `vocabulary1` ?

```
>>> fdist1['whale']
906
```

What does 906 mean?
Explain why `fdist1.freq('whale')` returns what it does?

10.2 Terms and tokenization

To retrieve the first 200 characters from the Gutenberg collection, use the following:

```
first200 = gutenberg.raw()[:200]
```

1. Tokenize the first 200 characters of the gutenberg collection. Use any tokenizer included in NLTK.
2. Use NLTK to perform stemming on these tokens. Do this with two different stemmers.
Show the result.

10.3 Naive Bayes Classification

For a review of concepts related to this exercise, please see chapter 6 of the NLTK book which can be found at http://nltk.googlecode.com/svn/trunk/doc/book/ch06.html

We know that verbs can be classified as *General Form*, *Past Tense*, *Present Participle* and *Past Participle* in English.

Some terms used here:

- V verb
- VD past tense
- VG present participle
- VN past participle

We can fetch the Brown Corpus data set here:

```
>>> import nltk
>>> rdata_set = nltk.corpus.brown.tagged_words(simplify_tags=True)
>>> simplev_set = ['V', 'VD', 'VG', 'VN']
>>> data_set = [item for item in rdata_set if item[1] in simplev_set]
```

We can get a subset of it:

```
>>> sdset = data_set[:2000]
```

In *nltk*, we already have a tagger. It can be used to tag a word, but the performance is poor. We can use it and calculate the accuracy:

```
>>> test_list = [item[0] for item in sdset]
>>> nltkres = nltk.pos_tag(test_list)
>>> # you can print nltkres to see the result
>>> corcount = 0
>>> for item in zip(sdset, nltkres):
>>>     if item[0][1] == item[1][1]:
>>>         corcount += 1
>>> nltktagger_accuracy = float(corcount) / len(sdset)
>>> print nltktagger_accuracy
```

So, we need to use a Naive Bayes Classifier to replace *nltk.pos_tag()*.

There are some clues to classify verbs, such as *Present Participle* verbs are always ending with *ing*.

Please build a Naive Bayes Classifier to put verbs from a text into these four categories, and generate the classifying accuracy.

Please use the *data_set* as follows (you can divide the developing set from the training set as your wish):

```
>>> train_set = data_set[:10000]
>>> test_set = data_set[10000:100000]
```

You need to submit your code and the result for us to evaluate. You can follow the method mentioned at the beginning of this document to put your code in a script instead of running it in the interactive shell. In this way, it is easier to edit the code and copy it to submit.

11 Evaluation of learning objective achievement

In evaluation the learning objectives, you will be evaluated on your ability to comprehend the features of NLTK touched upon in the exercises, as well as their utility through Python. Your solution should try to explain how these features work, their application, and how to apply them from Python.

12 Glossary

None

13 Additional useful links

- Official Python Website *http://www.python.org*
- The Python Standard Library: Regular Expression in Python *http://docs.python.org/library/re.html*
- NLTK book chapter 4: How to Program Basic Python *http://nltk.googlecode.com/svn/trunk/doc/book/ch04.html*

14 Contributors

Prepared for class CS6504 at Virginia Tech

- Initial Authors: Team 3
  - Xiaokui Shu
  - Ron Cohen
- Reviewer
  - Dr. Edward A. Fox