ΓΕ77
COMPUTATIONAL LINGUISTICS

Athanasios N. Karasimos

akarasimos@gmail.com

BA in Linguistics | National and Kapodistrian University of Athens

Lecture 11 | Wed 30 May 2018

Natural Language Analyses with NLTK

PYTHON AND NLTK

A short introduction to Natural Language ToolKit

PYTHON AND NLP

- **Python** is freely available programming language for many platforms from the Python Software Foundation:
  - http://www.python.org/
  - Named for the group Monty Python.
  - Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991.
  - An interpreted language, Python has a design philosophy that emphasizes code readability, and a syntax that allows programmers to express concepts in fewer lines of code

CHARACTERISTICS OF PYTHON

- Easy-to-learn scripting language, similar in many aspects to Perl
  - But with WYSIWYG block structure
- Object-oriented, with modules, classes, exceptions, high-level dynamic data types, similar to Java
- Strongly typed, but without type declarations (dynamic typing)
- Regular Expressions and other string processing features

NATURAL LANGUAGE TOOLKIT (NLTK)

- A suite of Python libraries for symbolic and statistical natural language programming
  - Developed at the University of Pennsylvania
- Developed to be a teaching tool and a platform for research NLP prototypes
  - Data types are packaged as classes
  - Goal of code is to be clear, rather than fastest performance
- Online book: http://www.nltk.org/book/
  - Authors: Edward Loper, Ewan Kline and Steven Bird

USING NLTK IN NLP

- NLTK provides libraries of many of the common NLP processes at various language levels
  - Leverage these libraries to process text
- Goal is to learn about and understand how NLP can be used to process text without programming all processes
  - However, some programming is required to
    - Call libraries
    - Process data
    - Customize NLP processes
  - Programming language is Python

GETTING STARTED IN PYTHON

- Python can be run as an interactive system
  - Type in expressions or small pieces of programs to try them out
- or as a command-line system.
  - Run stored python programs
- For both, it is recommended to use IDLE, the Python development environment
  - Especially good to edit Python programs in IDLE to keep track of the indentation for block structure

INTRODUCTION TO NLTK

- NLTK provides:
  - Basic classes for representing data relevant to Natural Language Processing.
  - Standard interfaces for performing NLP tasks such as tokenization, tagging and parsing
  - Standard implementation of each task which can be combined to solve complex problems

NLTK MODULES

- **corpora**: a package containing modules of example text
- **tokenize**: functions to separate text strings
- **probability**: for modeling frequency distributions and probabilistic systems
- **stem** – package of functions to stem words of text
- **wordnet** – interface to the WordNet lexical resource
- **chunk** – identify short non-nested phrases in text
- **etree**: for hierarchical structure over text
- **tag**: tagging each word with part-of-speech, sense, etc.
- **parse**: building trees over text - recursive descent, shift-reduce, probabilistic, etc.
- **cluster**: clustering algorithms
- **draw**: visualize NLP structures and processes
- **contrib**: various pieces of software from outside contributors

TUTORIALS FOR PYTHON AND NLTK

- Python
  - http://docs.python.org/tut/tut.html, the classic by Guido van Rossum
- NLTK is a SourceForge project at:
  - http://www.nltk.org
  - documentation: http://www.nltk.org/documentation,
  - including book: http://www.nltk.org/book
- API: http://nltk.googlecode.com/svn/trunk/doc/api/index.html

WHY TEXT PROCESSING?

- sentiment analysis
- spam filtering
- plagariasm detection / document similarity
- document categorization / topic detection
- phrase extraction, summarization
- smarter search
- simple keyword frequency analysis

SOME NLTK FEATURES

- part-of-speech tagging
- chunking & named entity recognition
- text classification
- many included corpora

SENTENCE TOKENIZATION

- >>> from nltk.tokenize import sent_tokenize
- >>> sent_tokenize("Hello SF Python. This is NLTK.")
  - ['Hello SF Python.', 'This is NLTK.']
- >>> sent_tokenize("Hello, Mr. Anderson. We missed you!")
  - ['Hello, Mr. Anderson.', 'We missed you!']

TEXT > SENTENCE TOKENIZER > **Sentence**

WORD TOKENIZATION

- >>> from nltk.tokenize import word_tokenize
- >>> word_tokenize('This is NLTK.')
  - ['This', 'is', 'NLTK', '.']
- >>> word_tokenize('Ας το δοκιμάσουμε και στα ελληνικά')
  - ['Ας', 'το', 'δοκιμάσουμε', 'και', 'στα', 'ελληνικά']

TEXT > SENTENCE TOKINIZER > WORD TOKINIZER > **Word**

WHAT'S A WORD?

- >>> word_tokenize("What's up?")
  - ['What', "'s", 'up', '?']
- >>> from nltk.tokenize import wordpunct_tokenize
- >>> wordpunct_tokenize("What's up?")
  - ['What', "'", 's', 'up', '?']

- Learn More: http://text-processing.com/demo/tokenize/

TEXT > SENTENCE TOKENIZER > WORD TOKENIZER > **Word**

PART-OF-SPEECH TAGGING

- >>> words = word_tokenize("And now for something completely different")
- >>> from nltk.tag import pos_tag
- >>> pos_tag(words)
  - [('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something', 'NN'), ('completely', 'RB'), ('different', 'JJ')]

TEXT > SENTENCE TOKENIZER > WORD TOKENIZER > POS TAGGER > **Word, Tag**

WHY PART-OF-SPEECH TAG?

- word definition lookup (WordNet, WordNik)
- fine-grained text analytics
- part-of-speech specific keyword analysis
- chunking & named entity recognition (NER)

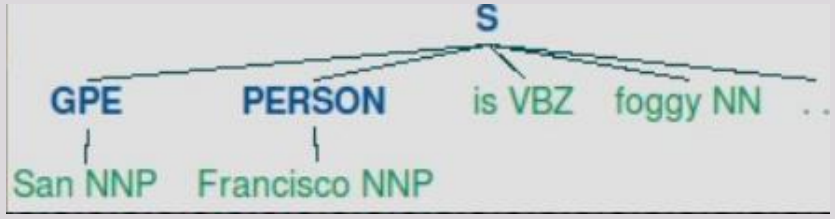
CHUNKING & NER

- >>> from nltk.chunk import ne_chunk
- >>> ne_chunk(pos_tag(word_tokenize('My name is Jacob Perkins.')))
  - Tree('S', [('My', 'PRP$'), ('name', 'NN'), ('is', 'VBZ'),
  - Tree('PERSON', [('Jacob', 'NNP'), ('Perkins', 'NNP')]), ('.', '.')])

TEXT > SENTENCE TOKENIZER > WORD TOKENIZER > POS TAGGER > CHUNKER > **Syntactic Tree**

NER NOT PERFECT

- >>> ne_chunk(pos_tag(word_tokenize('San Francisco is foggy.')))
  - Tree('S', [
    - Tree('GPE', [('San', 'NNP')]),
    - Tree('PERSON', [('Francisco', 'NNP')]),
    - ('is', 'VBZ'), ('foggy', 'NN'), ('.', '.')])



TEXT CLASSIFICATION

- def bag_of_words(words):
  - return dict([(word, True) for word in words])
- >>> feats = bag_of_words(word_tokenize("great movie"))
- >>> import nltk.data
- >>> classifier = nltk.data.load('classifiers/ movie_reviews_NaiveBayes.pickle')
- >>> classifier.classify(feats) 'pos

TEXT >>> WORD TOKENIZER > TEXT CLASSIFIER > **Label**

CLASSIFICATION ALGORITHMS IN NLTK

- Naive Bayes
- Maximum Entropy / Logistic Regression
- Decision Tree
- SVM (coming soon)

TEXT >> WORD TOKENIZER > TEXT CLASSIFIER > **Label**

NLTK-TRAINER

- https://github.com/japerk/nltk-trainer
- command line scripts
- train custom models
- analyze corpora
- analyze models against corpora

TRAIN A SENTIMENT CLASSIFIER

$ ./train_classifier.py movie_reviews --instances paras loading movie_reviews

2 labels: ['neg', 'pos']

2000 training feats, 2000 testing feats training

NaiveBayes classifier

accuracy: 0.967000 neg

precision: 1.000000 neg

recall: 0.934000 neg

f-measure: 0.965874 pos

precision: 0.938086 pos

recall: 1.000000 pos

f-measure: 0.968054

dumping NaiveBayesClassifier to ~/nltk_data/classifiers/ movie_reviews_NaiveBayes.pickle

OTHER PYTHON NLP LIBRARIES

- pattern: http://www.clips.ua.ac.be/pages/pattern
- scikits.learn: http://scikit-learn.sourceforge.net/stable/
- fuzzywuzzy: https://github.com/seatgeek/fuzzywuzzy
- TextProcessing: http://text-processing.com/

- http://www.nltk.org/book/
- http://www.nltk.org/book_1ed/