NLTK

Mahdi Hosseinzadeh

NLP or Why Analyze Text

- To get information out of textual data
- Sentiment analysis of messages
- Spam filtering
- Document similarity (plagiarism detection)
- Document categorization (topic detection)
- Social media data analysis

NLP and Machine Learning

How can we make a computer understand language?

- Based on language rules, patterns, or statistics
- Statistics are more accurate and popular

Natural Language Processing requires flexibility, which generally comes from machine learning.

Languages

Formal

- Strict, unchanging rules
- No ambiguity
- Inflexible: no new terms
- Parsable by regex
- Generally app-specific (math etc.)

Natural

- Flexible and evolving
- Ambigous: redundancy
- Very flexible
- Difficult to parse
- Used in many domains

What is NLTK

NLTK: Natural Language Toolkit

- Several text processing utilities and corpora
- Interface to over 50 corpora and lexical resources
- Focus on Machine Learning
- Free and Open Source
- Numpy and Scipy under the hood
- Fast and Formal
- Does not work well with Persian

  Other libraries like *spaCy* and *Hazm* can be used instead

NLTK Design Goals

Requirements

- Ease of use
- Consistency
- Extensibility
- Documentation
- Simplicity
- Modularity

Non-requirements

- Comprehensiveness
- Efficiency
- Performance
- Cleverness

NLTK Features

- Tokenization
- Stemming
- Tagging
- Chunking
- Parsing
- Classification
- Named-entity recognition

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 with recursive descent, shift-reduce etc.
- **cluster:** clustering algorithms
- **draw:** visualize NLP structures and processes
- **contrib:** various pieces of software from outside contributors

NLTK Good Points

- Trained models can be very fast
- Well known algorithms can be very accurate
- 3 Classification Algorithms
- 9 Part-of-Speech Tagging Algorithms
- Stemming Algorithms for 15 Languages
- 5 Word Tokenization Algorithms
- Sentence Tokenizers for 16 Languages
- 60 included corpora

NLTK Bad Points

- NLProc is hard
- Few out-of-the-box solutions (see Pattern)
- Not designed for big-data (see Mahout)
- Doesn’t have latest algorithms (see Scikits-Learn)
- No online or active learning algorithms
- Models can use a lot of memory

Install NLTK & Download Corpora

Installing NLTK: http://nltk.org/install

Training sets and corpora should be downloaded first
A corpus is a collection of documents to learn about

Resource *wordnet* not found

Please use the NLTK Downloader to obtain the resource

Python console:

```
>>> import nltk
>>> nltk.download('wordnet')
```

Tokenization

First step in NLP:

- Sentence segmentation
- Word tokenization

Token:

- A string of contiguous characters
- Space is the most common delimiter

POS Tagging

Part of speech:

- Whether it is a noun, a verb, etc.
- Helps to understand the sentence
- Used when parsing the text
- Needs training
  - So is based on statistics

CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential "there"
FW	Foreign word
IN	Prepostion or subordination conjunction
JJ	Adjective
JJR	Adjective- comparative
JJS	Adjective- superlative
LS	List item marker
MD	Modal
NN	Noun- singular or mass
NNS	Noun- plural
NP	Proper noun- singular
NPS	Proper noun- plural

Normalization

Stemming

- Works on affixes
- Just some predefined rules
- No additional knowledge required

Lemmatization

- More soffisticated
- Needs dictionary
- Also may need pos tags for homonyms

Stopwords

Common words with little value:

- Like is, and, are, a, the etc.
- May depend on the domain

Specifying the stopwords:

- Can define them manually
- Use idf or tf-idf to detect them

Chunking and Parsing

How do we get a machine to understand the text?

- We need structure

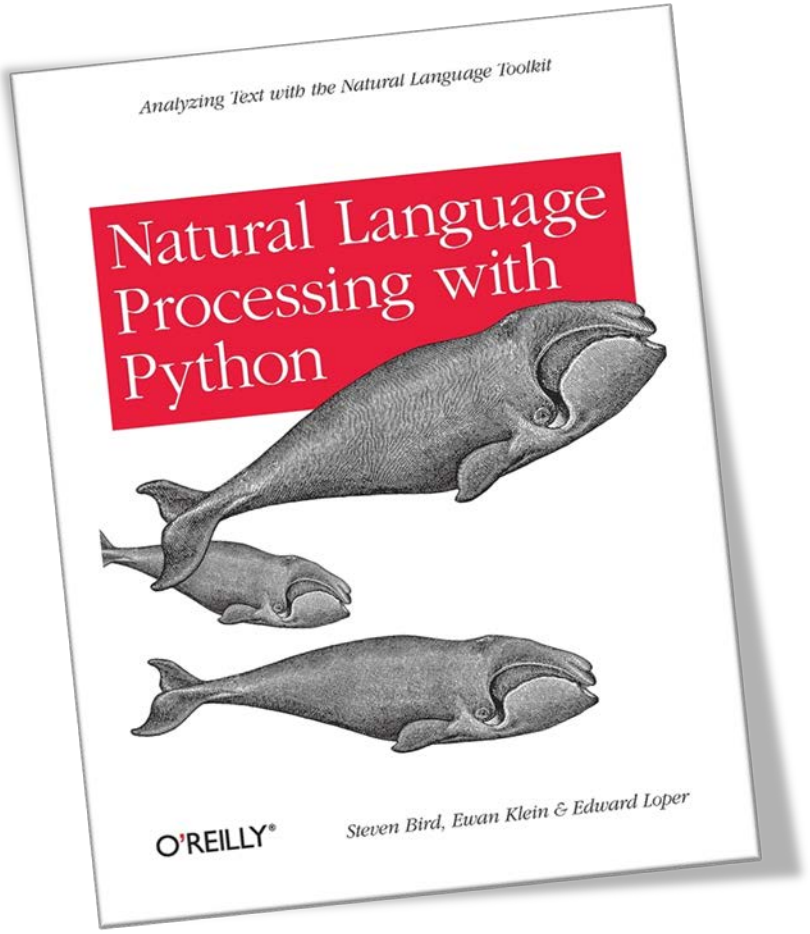
Chunking or shallow parsing:

- Group words that represent a single idea or thing

Generate parse tree:

- Root - the sentence
- Intermediate - noun phrase, verb phrase etc.
- Leaf - the words

NLTK Learning Resource



Thanks