Natural Language ToolKit

November 27, 2017

Overview

Natural Language ToolKit (NLTK)

- Python-based
- Comprehensive
- Useful for research and prototyping

NLTK Modules

Language processing task	NLTK modules	Functionality
Accessing corpora	corpus	standardized interfaces to corpora and lexicons
String processing	tokenize, stem	tokenizers, sentence tokenizers, stemmers
Collocation discovery	collocations	t-test, chi-squared, PMI
Part-of-speech tagging	tag	n-gram, backoff, Brill, HMM, TnT
Machine learning	classify, cluster, tbl	decision tree, max entropy, naive Bayes, EM, k-means
Chunking	chunk	regex, n-gram, NER
Parsing	parse, ccg	chart, feature-based, unification, probabilistic, dependency
Semantic interpretation	sem, inference	lambda calculus, first-order logic, model checking
Evaluation metrics	metrics	precision, recall, agreement coefficients
Probability and estimation	probability	frequency and smoothed probability distributions
Applications	app, chat	concordancer, parsers, WordNet browser, chatbots
Linguistic fieldwork	toolbox	manipulate data in SIL Toolbox format

NLTK Principles

- Simplicity
- Consistency
- Extensibility
- Modularity

NLTK Features

- Installation
- Theoretical background
- NLU
- Accessing corpora
- FreqDist, collocations, n-grams
- Stemming
- Lemmatization
- Segmentation, Tokenization
- POS-Taggers
- Chunking
- Information extraction
- Classifiers
- NER
- Parsers

Installation

Installation

- Python
  https://wiki.python.org/moin/BeginnersGuide/Download
- NLTK
  https://pypi.python.org/pypi/nltk
- PyCharm
  https://www.jetbrains.com/pycharm/download/

Text preprocessing

Text normalization

- tokenization
- stemming
- lemmatization
- stop word removal
- rare word removal

Segmentation

Tokenization

- Segmentation =
- Sentence tokenization =
- Sentence splitting =
- Sentence boundary detection, etc.

From splitting the string on (.) to a predictive classifier

Segmentation

Sample rules

1. Period?
2. The preceding token is in the list of abbreviations?
3. The next token is capitalized?
4. etc.

Better: maximum entropy models, NNs

Segmentation

On-demand segmentation

```
1 from nltk.tokenize import sent_tokenize
2 inputstring = 'This is an example sent.
    Will the sentence tokenizer split on
    sent markers?!'
3 all_sent = sent_tokenize(inputstring)
```

Segmentation

On-demand segmentation

```
1 from nltk.tokenize import sent_tokenize
2 inputstring = 'This is an example sent.
    Will the sentence tokenizer split on
    sent markers?!'
3 all_sent = sent_tokenize(inputstring)
```

[' This is an example sent', 'Will the sentence tokenizer split on sent markers?!']

Segmentation

PunktSentenceTokenizer instantiation

```
1 import nltk.tokenize.punkt
2 tokenizer = nltk.tokenize.punkt.
      PunktSentenceTokenizer()
```

Segmentation

PunktSentenceTokenizer instantiation

```
1 import nltk.tokenize.punkt
2 tokenizer = nltk.tokenize.punkt.
      PunktSentenceTokenizer()
```

[' This is an example sent', 'Will the sentence tokenizer split on sent markers?!']

Segmentation

Tokenizing sentences in other languages

```
1 import nltk.data
2 tokenizer = nltk.data.load('tokenizers/
    punkt/PY3/english.pickle')
3 tokenizer.tokenize(inputstring)
4 spanish_tokenizer = nltk.data.load('
    tokenizers/punkt/PY3/spanish.pickle')
5 spanish_tokenizer.tokenize('Hola amigo.
    Estoy bien.')
```

Segmentation

Tokenizing sentences in other languages

```
1 import nltk.data
2 tokenizer = nltk.data.load('tokenizers/
    punkt/PY3/english.pickle')
3 tokenizer.tokenize(inputstring)
4 spanish_tokenizer = nltk.data.load('
    tokenizers/punkt/PY3/spanish.pickle')
5 spanish_tokenizer.tokenize('Hola amigo.
    Estoy bien.')
```

['Hola amigo.', 'Estoy bien.']

You can see a list of all the available language tokenizers in /usr/share/nltk_data/ tokenizers/punkt/PY3 (or C:\nltk_data\tokenizers\punkt\PY3).

Segmentation

Training a sentence tokenizer

White guy: So, do you have any plans for this evening?

Asian girl: Yeah, being angry!

White guy: Oh, that sounds good.

Segmentation

```
1 from nltk.tokenize import
      PunktSentenceTokenizer
2 from nltk.corpus import webtext
3 text = webtext.raw('overheard.txt')
4 sent_tokenizer = PunktSentenceTokenizer(
      text)
5 sents1 = sent_tokenizer.tokenize(text)
6 sents1[0]
```

Segmentation

```
1 from nltk.tokenize import
      PunktSentenceTokenizer
2 from nltk.corpus import webtext
3 text = webtext.raw('overheard.txt')
4 sent_tokenizer = PunktSentenceTokenizer(
      text)
5 sents1 = sent_tokenizer.tokenize(text)
6 sents1[0]
```

'White guy: So, do you have any plans for this evening?'

Segmentation

Compare:

```
1 from nltk.tokenize import sent_tokenize
2 sents2 = sent_tokenize(text)
3 sents1[678]
```

Segmentation

Compare:

```
1 from nltk.tokenize import sent_tokenize
2 sents2 = sent_tokenize(text)
3 sents1[678]
```

'Girl: But you already have a Big Mac...'

Segmentation

Compare:

```
1 from nltk.tokenize import sent_tokenize
2 sents2 = sent_tokenize(text)
3 sents1[678]
```

'Girl: But you already have a Big Mac...'

```
1 sents2[678]
```

Segmentation

Compare:

```
1 from nltk.tokenize import sent_tokenize
2 sents2 = sent_tokenize(text)
3 sents1[678]
```

'Girl: But you already have a Big Mac...'

```
1 sents2[678]
```

'Girl: But you already have a Big Mac...\nHobo: Oh, this is all theatrical.'

Segmentation

PunktSentenceTokenizer

Heuristics:

- Orthography
- Frequent sentence starters
- Collocations
- Initials

Tokenization

The simplest tokenizer: the split() method of Python strings.

```
1 s = "Hi Everyone ! hola gr8"
2 s.split()
```

Tokenization

The simplest tokenizer: the split() method of Python strings.

```
1 s = "Hi Everyone ! hola gr8"
2 s.split()
```

['Hi', 'Everyone', '!', 'hola', 'gr8']

Tokenization

The word_tokenize method

```
1 from nltk.tokenize import word_tokenize
2 word_tokenize(s)
```

Tokenization

The word_tokenize method

```
1 from nltk.tokenize import word_tokenize
2 word_tokenize(s)
```

['Hi', 'Everyone', '!', 'hola', 'gr8']

Tokenization

The regex_tokenize method

```
1 from nltk.tokenize import regexp_tokenize
2 regexp_tokenize(s, pattern='\w+')
```

Tokenization

The regex_tokenize method

```
1 from nltk.tokenize import regexp_tokenize
2 regexp_tokenize(s, pattern='\w+')
```

['Hi', 'Everyone', 'hola', 'gr8']

Tokenization

The regex_tokenize method

```
1 from nltk.tokenize import regexp_tokenize
2 regexp_tokenize(s, pattern='\w+')
```

['Hi', 'Everyone', 'hola', 'gr8']

```
1 regexp_tokenize(s, pattern='\d+')
```

Tokenization

The regex_tokenize method

```
1 from nltk.tokenize import regexp_tokenize
2 regexp_tokenize(s, pattern='\w+')
```

['Hi', 'Everyone', 'hola', 'gr8']

```
1 regexp_tokenize(s, pattern='\d+')
```

['8']

Tokenization

The wordpunct_tokenize and blankline_tokenize methods

```
1 from nltk.tokenize import
      wordpunct_tokenize, blankline_tokenize
2 wordpunct_tokenize(s)
```

Tokenization

The wordpunct_tokenize and blankline_tokenize methods

```
1 from nltk.tokenize import
      wordpunct_tokenize, blankline_tokenize
2 wordpunct_tokenize(s)
```

['Hi', ',', 'Everyone', '!!', 'hola', 'gr8']

Tokenization

The wordpunct_tokenize and blankline_tokenize methods

```
1 from nltk.tokenize import
      wordpunct_tokenize, blankline_tokenize
2 wordpunct_tokenize(s)
```

['Hi', ',', 'Everyone', '!!', 'hola', 'gr8']

```
1 blankline_tokenize(s)
```

Tokenization

The wordpunct_tokenize and blankline_tokenize methods

```
1 from nltk.tokenize import
      wordpunct_tokenize, blankline_tokenize
2 wordpunct_tokenize(s)
```

['Hi', ',', 'Everyone', '!!', 'hola', 'gr8']

```
1 blankline_tokenize(s)
```

['Hi, Everyone !! hola gr8']

Tokenization

Experiment

```
1 from nltk.tokenize import word_tokenize,
    regexp_tokenize, wordpunct_tokenize,
    blankline_tokenize
```

Tokenization

Experiment

```
1 from nltk.tokenize import word_tokenize,
    regexp_tokenize, wordpunct_tokenize,
    blankline_tokenize
```

http://text-processing.com/demo

Tokenization

More on the word_tokenize method

```
1 from nltk.tokenize import
      TreebankWordTokenizer
2 tokenizer = TreebankWordTokenizer()
3 tokenizer.tokenize('Hello World.')
```

Tokenization

More on the word_tokenize method

```
1 from nltk.tokenize import
      TreebankWordTokenizer
2 tokenizer = TreebankWordTokenizer()
3 tokenizer.tokenize('Hello World.')
```

['Hello', 'World', '.']

Tokenization

Other subclasses of TokenizerI:

- TreebankWordTokenizer
- WhitespaceTokenizer
- SpaceTokenizer
- PunktWordTokenizer
- WordPunctTokenizer
- BlanklineTokenizer
- etc.

Tokenization

Separating contractions

```
1 from nltk.tokenize import
      TreebankWordTokenizer
2 tokenizer = TreebankWordTokenizer()
3 tokenizer.tokenize("can't")
```

Tokenization

Separating contractions

```
1 from nltk.tokenize import
      TreebankWordTokenizer
2 tokenizer = TreebankWordTokenizer()
3 tokenizer.tokenize("can't")
```

['ca', "n't"]

Tokenization

Separating contractions

```
1 from nltk.tokenize import
      PunktWordTokenizer
2 tokenizer = PunktWordTokenizer()
3 tokenizer.tokenize("Can't is a
      contraction.")
```

Tokenization

Separating contractions

```
1 from nltk.tokenize import
      PunktWordTokenizer
2 tokenizer = PunktWordTokenizer()
3 tokenizer.tokenize("Can't is a
      contraction.")
```

['Can', "'t", 'is', 'a', 'contraction.']

Tokenization

Separating contractions

```
1 from nltk.tokenize import
      WordPunctTokenizer
2 tokenizer = WordPunctTokenizer()
3 tokenizer.tokenize("Can't is a
      contraction.")
```

Tokenization

Separating contractions

```
1 from nltk.tokenize import
      WordPunctTokenizer
2 tokenizer = WordPunctTokenizer()
3 tokenizer.tokenize("Can't is a
      contraction.")
```

['Can', "'", 't', 'is', 'a', 'contraction', '.']

Tokenization

More on tokenization with regular expressions

Match on:

- tokens
- separators or gaps

Tokenization

Regex tokenization

```
from nltk.tokenize import RegexpTokenizer
tokenizer = RegexpTokenizer("[\w']+")
tokenizer.tokenize("Can't is a
    contraction.")
```

Equal to:

```
from nltk.tokenize import regexp_tokenize
regexp_tokenize("Can't is a contraction."
    , "[\w']+")
```

Tokenization

Regex tokenization

```
1 from nltk.tokenize import RegexpTokenizer
2 tokenizer = RegexpTokenizer("[\w']+")
3 tokenizer.tokenize("Can't is a
    contraction.")
```

Equal to:

```
1 from nltk.tokenize import regexp_tokenize
2 regexp_tokenize("Can't is a contraction."
    , "[\w']+")
```

["Can't", 'is', 'a', 'contraction']

Tokenization

Simple whitespace tokenizer

```
1 tokenizer = RegexpTokenizer('\s+', gaps=
    True)
2 tokenizer.tokenize("Can't is a
    contraction.")
```

Tokenization

Simple whitespace tokenizer

```
1 tokenizer = RegexpTokenizer('\s+', gaps=
    True)
2 tokenizer.tokenize("Can't is a
    contraction.")
```

["Can't", 'is', 'a', 'contraction.']

Stemming

Example: *eat*

eating, *eaten*, *eats*

Stemming

Stemming strategies

- Lookup table
- Rule-based affix stripping
- Hybrid approaches
- Suffix substitution
- Recursive rules

Stemming

```
1 from nltk.stem import PorterStemmer
2 from nltk.stem.lancaster import
      LancasterStemmer
3 pst = PorterStemmer()
4 lst = LancasterStemmer()
5 lst.stem("eating")
```

Stemming

```
1 from nltk.stem import PorterStemmer
2 from nltk.stem.lancaster import
      LancasterStemmer
3 pst = PorterStemmer()
4 lst = LancasterStemmer()
5 lst.stem("eating")
```

'eat'

Stemming

```
1 from nltk.stem import PorterStemmer
2 from nltk.stem.lancaster import
      LancasterStemmer
3 pst = PorterStemmer()
4 lst = LancasterStemmer()
5 lst.stem("eating")
```

'eat'

```
1 pst.stem("hopping")
```

Stemming

```
1 from nltk.stem import PorterStemmer
2 from nltk.stem.lancaster import
      LancasterStemmer
3 pst = PorterStemmer()
4 lst = LancasterStemmer()
5 lst.stem("eating")
```

'eat'

```
1 pst.stem("hopping")
```

'hop'

Stemming

Subclasses of StemmerI:

- PorterStemmer
- LancasterStemmer
- RegexStemmer
- SnowballStemmer

Stemming

```
1 from nltk.stem import RegexpStemmer
2 stemmer = RegexpStemmer('ing')
3 stemmer.stem('cooking')
```

Stemming

```
1 from nltk.stem import RegexpStemmer
2 stemmer = RegexpStemmer('ing')
3 stemmer.stem('cooking')
```

'cook'

Stemming

```
1 from nltk.stem import RegexpStemmer
2 stemmer = RegexpStemmer('ing')
3 stemmer.stem('cooking')
```

'cook'

```
1 stemmer.stem('ingenious')
```

Stemming

```
1 from nltk.stem import RegexpStemmer
2 stemmer = RegexpStemmer('ing')
3 stemmer.stem('cooking')
```

'cook'

```
1 stemmer.stem('ingenious')
```

'enious'

Stemming

```
1 from nltk.stem import SnowballStemmer
2 SnowballStemmer.languages('danish', '
    dutch', 'english', 'finnish', 'french'
    , 'german', 'hungarian', 'italian', '
    norwegian', 'porter', 'portuguese', '
    romanian', 'russian', 'spanish', '
    swedish')
3 spanish_stemmer = SnowballStemmer('
    spanish')
4 spanish_stemmer.stem('hola')
```

Stemming

```
1 from nltk.stem import SnowballStemmer
2 SnowballStemmer.languages('danish', '
    dutch', 'english', 'finnish', 'french'
    , 'german', 'hungarian', 'italian', '
    norwegian', 'porter', 'portuguese', '
    romanian', 'russian', 'spanish', '
    swedish')
3 spanish_stemmer = SnowballStemmer('
    spanish')
4 spanish_stemmer.stem('hola')
```

'hol'

Stemming

Experiment

```
1 from nltk.stem import PorterStemmer,
      SnowballStemmer, RegexpStemmer
2 from nltk.stem.lancaster import
      LancasterStemmer
3 g_snst = SnowballStemmer('german')
4 pst = PorterStemmer()
5 lst = LancasterStemmer()
6 rst = RegexpStemmer()
7
8 stemer.stem()
```

Stemming

Experiment

```
1 from nltk.stem import PorterStemmer,
      SnowballStemmer, RegexpStemmer
2 from nltk.stem.lancaster import
      LancasterStemmer
3 g_snst = SnowballStemmer('german')
4 pst = PorterStemmer()
5 lst = LancasterStemmer()
6 rst = RegexpStemmer()
7
8 stemer.stem()
```

http://text-processing.com/demo

Lemmatization

Lemmatisation algorithms

- Stochastic algorithms
- n-gram analysis
- Hybrid approaches

Lemmatization

```
1 from nltk.stem import WordNetLemmatizer
2 wlem = WordNetLemmatizer()
3 wlem.lemmatize("ate")
```

Lemmatization

```
1 from nltk.stem import WordNetLemmatizer
2 wlem = WordNetLemmatizer()
3 wlem.lemmatize("ate")
```

'eat'

Lemmatization

```
1 from nltk.stem import WordNetLemmatizer
2 wlem = WordNetLemmatizer()
3 wlem.lemmatize('cooking')
```

Lemmatization

```
1 from nltk.stem import WordNetLemmatizer
2 wlem = WordNetLemmatizer()
3 wlem.lemmatize('cooking')
```

'cooking'

Lemmatization

```
1 from nltk.stem import WordNetLemmatizer
2 wlem = WordNetLemmatizer()
3 wlem.lemmatize('cooking')
```

'cooking'

```
1 wlem.lemmatize('cooking', pos='v')
```

Lemmatization

```
1 from nltk.stem import WordNetLemmatizer
2 wlem = WordNetLemmatizer()
3 wlem.lemmatize('cooking')
```

'cooking'

```
1 wlem.lemmatize('cooking', pos='v')
```

'cook'

Lemmatization

```
1 from nltk.stem import WordNetLemmatizer
2 wlem = WordNetLemmatizer()
3 wlem.lemmatize('cooking')
```

'cooking'

```
1 wlem.lemmatize('cooking', pos='v')
```

'cook'

```
1 wlem.lemmatize('cookbooks')
```

Lemmatization

```
1 from nltk.stem import WordNetLemmatizer
2 wlem = WordNetLemmatizer()
3 wlem.lemmatize('cooking')
```

'cooking'

```
1 wlem.lemmatize('cooking', pos='v')
```

'cook'

```
1 wlem.lemmatize('cookbooks')
```

'cookbook'

Lemmatization

```
1 from nltk.stem import PorterStemmer
2 from nltk.stem import WordNetLemmatizer
3 stemmer = PorterStemmer()
4 stemmer.stem('believes')
```

Lemmatization

```
1 from nltk.stem import PorterStemmer
2 from nltk.stem import WordNetLemmatizer
3 stemmer = PorterStemmer()
4 stemmer.stem('believes')
```

'believ'

Lemmatization

```
1 from nltk.stem import PorterStemmer
2 from nltk.stem import WordNetLemmatizer
3 stemmer = PorterStemmer()
4 stemmer.stem('believes')
```

'believ'

```
1 lemmatizer = WordNetLemmatizer()
2 lemmatizer.lemmatize('believes')
```

Lemmatization

```
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
stemmer = PorterStemmer()
stemmer.stem('believes')
```

'believ'

```
lemmatizer = WordNetLemmatizer()
lemmatizer.lemmatize('believes')
```

'belief'

Stop word removal

```
1 from nltk.corpus import stopwords
2 stoplist = stopwords.words('english')
3 text = "This is just a test"
4 cleanwordlist = [word for word in text.
      split() if word not in stoplist]
```

Stop word removal

```
1 from nltk.corpus import stopwords
2 stoplist = stopwords.words('english')
3 text = "This is just a test"
4 cleanwordlist = [word for word in text.
    split() if word not in stoplist]
```

['test']

Stop word removal

```
1 from nltk.corpus import stopwords
2 stoplist = stopwords.words('english')
3 text = "This is just a test"
4 cleanwordlist = [word for word in text.
      split() if word not in stoplist]
```

['test']

```
1 stopwords.fileids()
```

Stop word removal


```
from nltk.corpus import stopwords
stoplist = stopwords.words('english')
text = "This is just a test"
cleanwordlist = [word for word in text.
    split() if word not in stoplist]
```

['test']

```
stopwords.fileids()
```

['danish', 'dutch', 'english', 'finnish', 'french', 'german', 'hungarian', 'italian', 'norwegian', 'portuguese', 'russian', 'spanish', 'swedish', 'turkish']

Rare word removal

```
1 freq_dist = nltk.FreqDist(tokens)
2 rarewords = freq_dist.keys()[-50:]
3 nonrarewords = [ word for word in token
      not in rarewords]
```

References

Bird, Steven, Ewan Klein, and Edward Loper (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc."

Perkins, Jacob (2014). *Python 3 Text Processing with NLTK 3 Cookbook*. Packt Publishing Ltd.