Information Retrieval
CS 6900

Lecture 05

Razvan C. Bunescu

School of Electrical Engineering and Computer Science

bunescu@ohio.edu

NLTK & Python

- A suite of libraries for tokenization, stemming, tagging, syntactic parsing, and semantic processing of text:
  - Online book at http://nltk.org/book.
  - Package and documentation at http://nltk.org/.
  - Installed on california.cs.ohio.edu

- Implemented in Python:
  - a multi-paradigm programming language:
    - imperative, functional, and object-programming.
  - interpreted, interactive.
  - extensive standard library.

HTML Processing

```
>>> import nltk
>>> from urllib import urlopen

>>> url = "http://nltk.org/book/ch01.html"
>>> html = urlopen(url).read()
>>> len(html)
197573
>>> html[:75]
'<?xml version="1.0" encoding="ascii" ?>\n'
>>> text = nltk.clean_html(html)
>>> text[:40]
'1 Language Processing and Python \n It is'
```

Segmentation & Tokenization

```
>>> from nltk.tokenize import word_tokenize, wordpunct_tokenize,
                                                    sent_tokenize

>>> t = '''Good muffins cost $3.88\nin New York.  Please buy me two of
them. Thanks.'''
>>> sent_tokenize(s)
['Good muffins cost $3.88\nin New York.', 'Please buy me two of them.',
'Thanks.']
>>> [word_tokenize(s) for st in sent_tokenize(s)]
[['Good', 'muffins', 'cost', '$', '3.88', 'in', 'New', 'York', '.'], ['Please', 'buy',
'me', 'two', 'of', 'them', '.'], ['Thanks', '.']]
```

Segmentation & Tokenization

```
>>> sent_tokens = [word_tokenize(s) for s in sent_tokenize(text)]
>>> sent_tokens[1]
['What', 'can', 'we', 'do', 'with', 'it', ',', 'assuming', 'we', 'can', 'write', 'some', 'simple', 'programs', '?']

>>> tokens = [token for sentence in sent_tokens for token in sentence]
>>> tokens[:20]
['1', 'Language', 'Processing', 'and', 'Python', 'It', 'is', 'easy', 'to', 'get', 'our', 'hands', 'on', 'millions', 'of', 'words', 'of', 'text', '.', 'What']
```

http://nltk.org/api/nltk.tokenize.html

Segmentation and Tokenization

- There are multiple tokenizers in NLTK:
  - nltk.word_tokenize()
    - works with raw text.
  - nltk.tokenize.word_tokenize()
    - works with sentences, currently TreeBankWordTokenizer().
  - WhiteSpaceTokenizer(), SpaceTokenizer(), LineTokenizer(), ...
  - RegexpTokenizer():
    - WordPunctTokenizer(), BlanklineTokenizer(), ...

- Experiment with them, compare output for IR.

Vocabulary and Statistics

```
>>> words = [w.lower() for w in tokens]
>>> len(words)
19164
>>> vocab = sorted(set(words))
>>> len(vocab)
2632
>>> vocab[:10]
['!', '#', '%', '&', "'", "''", "'-", "'1851", "'29", "'61"]
>>> [w for w in vocab if w.isalpha()][:10]
['a', 'ability', 'able', 'about', 'abundant', 'accept', 'access', 'accesses',
'accessing', 'accidentally']
```

Computing Word Frequency: Python

```
word_counts = {}
for word in words:
    if word not in word_counts:
        count[word] = 0
    count[word] += 1
```

```
>>> len(word_counts)
2632
>>> print word_counts['language'], word_counts['python']
49 102
```

Ranking Words by Frequency: Python

```
# Solution 1
>>> ranked = []
>>> for (w, f) in word_freq:
>>> . . . ranked.append((f, w))
# Solution 2:
>>> ranked = [(f, w) for (w, f) in word_freq]

>>> ranked.sort()
>>> ranked.reverse()
>>> ranked[:10]
[(1136, ','), (874, ';'), (796, '&'), (772, 'the'), (589, 'gt'), (534, '.'), (511, '"'),
(405, 'of'), (355, '('), (350, ')')]
```

Ranking Words by Frequency: NLTK

```
>>> fdist = nltk.FreqDist(words)
>>> len(fdist)
2632
>>> print fdist['language'], fdist['python']
49 102
>>> word_freq = fdist.items()
>>> word_freq[:10]
[(',', 1136), (';', 874), ('&', 796), ('the', 772), ('gt', 589), ('.', 534), ("""", 511),
('of', 405), ('(', 355), (')', 350)]
```

Stemming

[http://nltk.org/book/ch03.html]

- With regular expressions:

```
>>> import re

>>> re.findall(r'^(.*?)(ing|ly|ed|ious|ies|ive|es|s|ment)?$', 'processing')
[('process', 'ing')]

>>> re.findall(r'^(.*?)(ing|ly|ed|ious|ies|ive|es|s|ment)?$', 'processes')
[('process', 'es')]

>>> re.findall(r'^(.*?)(ing|ly|ed|ious|ies|ive|es|s|ment)?$', 'processes')
[('language', '')]
```

Stemming

[http://nltk.org/book/ch03.html]

- With **Porter** stemmer or **Lancaster** stemmer:

```
>>> raw = '''DENNIS: Listen, strange women lying in ponds
distributing swords is no basis for a system of government.'''
>>> tokens = = [w.lower() for w in nltk.word_tokenize(raw)]
>>> porter = nltk.PorterStemmer()
>>> [porter.stem(t) for t in tokens]
['denni', ':', 'listen', ',', 'strang', 'women', 'lie', 'in', 'pond', 'distribut',
'sword', 'is', 'no', 'basi', 'for', 'a', 'system', 'of', 'govern', '.']
>>> lancaster = nltk.LancasterStemmer()
>>> [lancaster.stem(t) for t in tokens]
['den', ':', 'list', ',', 'strange', 'wom', 'lying', 'in', 'pond', 'distribut',
'sword', 'is', 'no', 'bas', 'for', 'a', 'system', 'of', 'govern', '.']
```

Lemmatization

[http://nltk.org/book/ch03.html]

- The WordNet lemmatizer removes affixes if the resulting word is in its dictionary.
  - slower than the stemmers.

```
>>> wnl = nltk.WordNetLemmatizer()
>>> [wnl.lemmatize(t) for t in tokens]
['dennis', ':', 'listen', ',', 'strange', 'woman', 'lying', 'in', 'pond',
'distributing', 'sword', 'is', 'no', 'basis', 'for', 'a', 'system', 'of',
'government', '.']
```

Part of Speech Tagging

[http://nltk.org/api/nltk.tag.html]

- Using NLTK's default POS tagger:
  - with the Penn Treebank tagset.

```
>>> from nltk.tag import pos_tag
>>> from nltk.tokenize import word_tokenize

>>> pos_tag(word_tokenize("John's big idea isn't all that bad."))
[('John', 'NNP'), ("'s", 'POS'), ('big', 'JJ'), ('idea', 'NN'), ('is', 'VBZ'),
("n't", 'RB'), ('all', 'DT'), ('that', 'DT'), ('bad', 'JJ'), ('.', '.')]
```

Part of Speech Tagging

[http://nltk.org/api/nltk.tag.html]

- Using Stanford's POS tagger:

```
>>> from nltk.tag.stanford import POSTagger

>>> st = POSTagger('/usr/share/stanford-postagger/models/english-bidirectional-distsim.tagger',
                   '/usr/share/stanford-postagger/stanford-postagger.jar')

>>> st.tag('What is the airspeed of an unladen swallow?'.split())
[('What', 'WP'), ('is', 'VBZ'), ('the', 'DT'), ('airspeed', 'NN'), ('of', 'IN'),
('an', 'DT'), ('unladen', 'JJ'), ('swallow', 'VB'), ('?', '.')]
```