Text Data

LECTURE 4

Organization of Lecture 4

- Processing HTML Files
- Handling CSV Files
- Reading JSON Files
- Processing Text in Natural Languages

Processing Texts in Natural Languages

- 80% data is unstructured
- Includes audio, video, images, texts written in natural languages
- Text has no tags, no delimiters, no data types
- Still could be rich information
- May want to know if certain words are in text and their frequencies
  - Word and sentence tokenization
- If text is positive or negative in tone
  - Entity extraction
- Requires NLP

Python Natural Language Toolkit (`nltk`)

- Corpus
  - structured or unstructured collection of words or expressions
  - Stored in the module `nltk.corpus`
- Examples:
  - `gutenberg` – 18 English tests from Gutenberg Project, including Bible and Moby Dick
  - `names` – list of 8,000 male and female names
  - `words` – list of 235,000 most frequently used English words and forms
  - `stopwords` – list of words from 14 languages eliminated from analyses as they do not add anything; English list is in `stopwords.words("english")`
  - `cmudict` – pronunciation dictionary from Carnegie Mellon with 134,000 entries

Online Semantic WordNet

- nltk.corpus.wordnet is an interface object to the corpus WordNet
  - (internet access is required)
- WordNet is a collection of synsets (synonum sets)
  - Words tagged with a part-of-speech marker and a sequential number

```
wn = nltk.corpus.wordnet
wnsynsets("cat")        =>   [Synset('cat.n.01'), Synset('guy.n.01), … ]
```

  - Each synset, can look up definition

```
wn.syset("cat.n.02").definition() => 'an informal term for a youth or man'
```

Online Semantic WordNet

- Can calculate semantic similarity between two synsets.

```
import nltk
nltk.download('wordnet')
wn =nltk.corpus.wordnet
print (wn.synset("cat.n.01").path_similarity(wn.synset("lynx.n.01")))
```

```
ner/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
0.04
>
```

Online Semantic WordNet

- Can calculate semantic similarity between two synsets.

```
import nltk
nltk.download('wordnet')
wn =nltk.corpus.wordnet

#compute closest similarity between "cat" and "dog"
definitions = [simxy.definition() for simxy in max (
  (x.path_similarity(y), x, y)
    for x in wn.synsets('cat’)
    for y in wn.synsets('dog’)
    if x.path_similarity(y) #ensure they are related at all
) [1:]]

print (definitions)
```

```
[nltk_data] Downloading package wordnet to /home/run
[nltk_data] Downloading package wordnet to /home/runner/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
['an informal term for a youth or man', 'informal term for a man']
>
```

Normalization

- Procedure that prepares natural language text for further processing
- Steps (usually in this order)
  - Tokenization – breaking text into sentences or words (strings)
  - Conversion of words to all same-case characters
  - Elimination of stop words (use corpus stopwords-which are all lowercase)
  - Stemming – conversion of word forms to their stems
  - Lemmatization – slower, more conservative stemming (must have internet)

Tokenization

- NLTK provides two simple and two advanced tokenizzers.
  - nltk.word_tokenize(text)
  - nltk. sent_tokenize(text) – sentence tokenizer
  - nltk. regexp_tokenize(text. re) – re is a regular expression
  - WordPunctTokenizer.tokenize – for non-alphabetic characters, emoticons

Tokenizer Exampe

```
import nltk
from nltk.tokenize import WordPunctTokenizer
nltk.download('punkt')

#compare WordPunctTokenizer.tokenize() to word_tokenize()
text = "}Help! :))) :[ ..... :D{"
word_punct = WordPunctTokenizer()
print(word_punct.tokenize(text))
print (nltk.word_tokenize(text))
```

```
['}', 'Help', '!', ':)))', ':[', '.....', ':', 'D', '{']
['}', 'Help', '!', ':', ')', ')', ')', ':', '[', '.....', ':', 'D', '{']
```

Stemming

- NLTK supplies 2 basic stemmers applied to a single word
  - Porter stemmer – not as aggressive
  - Lancaster stemmer – more aggressive (produces homonymous stems, words that are spelled the same or pronounced the same with different meanings)
- Both stemmers have the function stem(word)

```
import nltk
pstemmer = nltk.PorterStemmer()
print(pstemmer.stem ("wonderful") )
print(pstemmer.stem ("feet"))
print (nltk.LancasterStemmer().stem ("wonderful"))
print (nltk.LancasterStemmer().stem ("feet"))
```

```
wonder
feet
wond
feet
>
```

Lemmatization

- WordNetLemmatizer looks up calculated stems in WordNet and accepts them ONLY if they exist as words or forms.

```
import nltk
nltk.download('wordnet')
lemmatizer = nltk.WordNetLemmatizer()
print("stem of wonderfully is",
lemmatizer.lemmatize("wonderfully"))
print("stem of bats is", lemmatizer.lemmatize("bats"))
print("stem of feet is", lemmatizer.lemmatize( "feet"))
```

```
stem of wonderfully wonderfully
stem of bats bat
stem of feet foot
>
```

Putting it altogether

- Display 10 most frequent non-stop word stems in an .html file

- https://replit.com/@CSREPLIT/TenMostCommonWords#main.py

```
import nltk
from bs4 import BeautifulSoup
from collections import Counter
from nltk.corpus import stopwords
#from nltk import LancasterStemmer

nltk.download('punkt')
nltk.download('stopwords')

#create a new stemmer
ls = nltk.LancasterStemmer()

#read the file and cook a soup
with open ("Tornado.html") as infile:
soup = BeautifulSoup(infile, features="html5lib")

#extract and tokenize the text
words = nltk.word_tokenize(soup.text)

#convert to lowercase
words = [w.lower() for w in words]

#eliminate stop words and stem the rest of the words
words = [ls.stem(w) for w in words if w not in stopwords.words("english") and
w.isalnum()]

#tally the words
freqs = Counter(words)
print (freqs.most_common(10))
```

```
[nltk_data] Downloading package punkt to /home/runner/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /home/runner/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[('tornado', 501), ('storm', 100), ('retriev', 89), ('weath', 84), ('nat', 70), ('cloud', 68)
('wind', 52), ('origin', 50), ('sev', 48), ('arch', 44)]
>
```

Other Stemming Mechanisms

- PorterStemmer

```
[('tornado', 499), ('storm', 100), ('retriev', 89), ('weather', 84), (
'cloud', 68), ('nation', 57), ('wind', 52), ('origin', 50), ('sever',
48), ('archiv', 44)]
>
```

- WordNetLemmatizer

```
[('tornado', 499), ('storm', 100), ('retrieved', 89), ('weather', 84), ('cloud', 68),
('national', 56), ('wind', 52), ('archived', 44), ('original', 44), ('area', 42)]
>
```