Introduction to Human Language Technologies
2. Document structure

Natural Language Research Group

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Course 2018/19

Outline

Session requirements

Beautiful Soup 4:

- Linux (via shell)
  - > `pip3 install beautifulsoup4`
  - > `pip3 install lxml`
- Windows (via cmd)
  - > `pip install beautifulsoup4`
  - > `pip install lxml`

Tokenizers:

- Both Linux & Windows (via python shell)
  - > `import nltk`
  - > `nltk.download('punkt')`

Attached resources:

- `trial.tgz`
- `langId.zip`

Outline

Beautiful Soup:

Getting raw text from HTML:

```
In [1]: import urllib.request
        from bs4 import BeautifulSoup

        url = 'https://www.crummy.com/software/BeautifulSoup/bs4/doc/'
        with urllib.request.urlopen(url) as response:
           dt = response.read().decode('utf8')

        soup = BeautifulSoup(dt, 'html.parser')
        print(soup.get_text())

...
Beautiful Soup Documentation¶

Beautiful Soup is a
Python library for pulling data out of HTML and XML files. It works
with your favorite parser to provide idiomatic ways of navigating,
searching, and modifying the parse tree. It commonly saves programmers
hours or days of work.
...
```

Beautiful Soup:

- Beautiful Soup also allows to treat HTML in all forms (`https://www.crummy.com/software/BeautifulSoup/bs4/doc/`)
  - Output: raw text (`.get_text()`), pretty-printing...
  - Manipulating tags: name, attributes, content...
  - Navigating the tree: children, parent, siblings...
  - Searching the tree: string, regular expressions, functions...
  - Modifying the tree
  - Encoding
  - Parsing only a part of a document
  - ...

XML options:

- Beautiful Soup
  - Just changing the second argument of the constructor
  - > `soup = BeautifulSoup(dt, 'xml')`
- xml.etree.ElementTree
  - `https://docs.python.org/3.7/library/xml.etree.elementtree.html`
  - Standard python library
  - Builds a tree and provides methods for navigating, searching and modifying it
- xml.sax
  - `https://docs.python.org/3.7/library/xml.sax.html`
  - Standard python library
  - Processes the xml file without building the tree
  - It works based on events
  - It allows to process very big xml files such as big corpora

Sax example:

Sax example:

```
In [1]: import xml.sax

        class ChgHandler(xml.sax.ContentHandler):
            cnt = 1
            mn = (1.0, 'EUR')

            def startElement(self, name, attrs):
                if name == "Cube":
                    if 'rate' in attrs.keys():
                        # print(attrs.getValue('currency'), attrs.getValue('rate'))
                        ChgHandler.cnt += 1
                        ChgHandler.mn = min(ChgHandler.mn,
                                            (float(attrs.getValue('rate')),
                                             attrs.getValue('currency')))


        url = 'http://www.ecb.europa.eu/stats/eurofxref/eurofxref-daily.xml'

        parser = xml.sax.make_parser()
        parser.setContentHandler(ChgHandler())
        parser.parse(url)

        ChgHandler.mn

Out[1]: (0.8917, 'GBP')
```

Outline

Text level in nltk library

- Depending on the needs, text can be splitted into sentences before tokenizing or it can be directly tokenized. (http://www.nltk.org/_modules/nltk/tokenize.html)
- Standard functions (recommended by nltk):
  - > `s_list = nltk.sent_tokenize(T, [language='LANG'])`
  - > `t_list = nltk.word_tokenize(s, [language='LANG'])`

  `LANG` can be:

  czech, danish, ducth, english, estonian, finnish, french, german, greek, italian, norwegian, polish, portuguese, slovene, spanish, swedish or turkish

  Transform the text previously when it is a Unicode string:
  - > `T.decode('utf8')`

Example

Sentence splitting & tokenization:

```
In [1]: import nltk
        # sentence splitting
        source = 'Men want children. They get relaxed with kids.'
        sentences = nltk.sent_tokenize(source)
        sentences

Out[1]: ['Men want children.', 'They get relaxed with kids.']

In [2]: # tokenizer
        [nltk.word_tokenize(s) for s in sentences]

Out[2]: [['Men', 'want', 'children', '.'],
         ['They', 'get', 'relaxed', 'with', 'kids', '.']]
```

Similarities

Set-oriented methods: similarities between sets of words:

- Dice: $S_{\mathtt{dice}}(X, Y) = \frac{2 \cdot |X \cap Y|}{|X| + |Y|}$
- Jaccard: $S_{\mathtt{jaccard}}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$
- Overlap: $S_{\mathtt{overlap}}(X, Y) = \frac{|X \cap Y|}{\mathtt{min}(|X|, |Y|)}$
- Cosine: $S_{\mathtt{cosine}}(X, Y) = \frac{|X \cap Y|}{\sqrt{|X| \cdot |Y|}}$

Above similarities are in [0, 1] and can be used as distances simply subtracting: $D = 1 - S$.

Example:

```
In [1]: from nltk.metrics import jaccard_distance
        jaccard_distance(set(['The','eats','fish','.']),
                         set(['The','eats','blue','fish','.']))

Out[1]: 0.2
```

Mandatory exercise

Statement:

1. Read all pairs of sentences of the trial set within the evaluation framework of the project.
2. Compute their similarities by considering words and Jaccard distance.
3. Compare the previous results with gold standard by giving the pearson correlation between them.

> `from scipy.stats import pearsonr`

> `pearsonr(refs, tsts)[0]`

Notes:

- Read the file `00-readme.txt` of the trial data set to prepare the exercise.
- Justify the answer.

Outline

How to build a language identifier

- Preprocess:
  - Remove all the digits and punctuation from data
  - Convert all the texts to lower case
  - Replace continuous white spaces by a single one
  - Concatenate all sentences with a double space in between
- Use character trigrams as language model (section 4.2 of Jurafsky's book).
- Use *Laplace smoothing* technique to avoid zero counts (section 4.5.1 of Jurafsky's book).
- Remove all trigrams that occurs less than 5 times in training corpus.
- Note: string and regular expression python libraries are useful.

Trigrams with chars

Example:

```
In [1]: from nltk.collocations import TrigramCollocationFinder

        sq = 'the cat and the dog of the man are quite'

        # trigrams generation by chars
        finder = TrigramCollocationFinder.from_words(sq)
        # filtering: remove less than 2
        finder.apply_freq_filter(2)

        [tr for tr in finder.ngram_fd.items()]

Out[1]: [(('h', 'e', ' '), 3), ((' ', 't', 'h'), 2), (('t', 'h', 'e'), 3)]
```

Bigrams with words

Example:

```
In [2]: from nltk.collocations import BigramCollocationFinder
        from nltk import word_tokenize

        # bigrams generation by words
        finder = BigramCollocationFinder.from_words(word_tokenize(sq))

        [tr for tr in finder.ngram_fd.items()]

Out[2]: [(('and', 'the'), 1),
         (('the', 'man'), 1),
         (('the', 'dog'), 1),
         (('cat', 'and'), 1),
         (('of', 'the'), 1),
         (('dog', 'of'), 1),
         (('the', 'cat'), 1),
         (('are', 'quite'), 1),
         (('man', 'are'), 1)]
```

Accuracy & confusion matrix

Example:

```
In [1]: from nltk.metrics.scores import accuracy

        ref  = ['eng', 'spa', 'eng', 'nld']
        test = ['eng', 'spa', 'spa', 'nld']

        'accuracy: ' + str(accuracy(ref, test))

Out[1]: 'accuracy: 0.75'

In [2]: from nltk.metrics import ConfusionMatrix

        cm = ConfusionMatrix(ref, test)
        print(cm.pretty_format())

    | e n s |
    | n l p |
    | g d a |
----+-------+
eng |<1>. 1 |
nld | .<1>. |
spa | . .<1>|
----+-------+
(row = reference; col = test)
```

Optional exercise

Statement:

- Implement a language identifier for the European languages: English, Spanish, Dutch, German, Italian & French.
- Use *wortschatz leipzig corpora*:

  `http://wortschatz.uni-leipzig.de/en/download`

  It contains 30k sentences of each of the 6 languages as training set and 10k of each as test set.
- Use the guide of previous slide. Give the accuracy and confusion matrix on the test set as result.

Example of project output

Our implementation:

- learning time: 13:29.425
- test time: 2:31.334
- accuracy: 0.9989
- confusion matrix:

```
    |    d    e    f    i    n    s |
    |    e    n    r    t    l    p |
    |    u    g    a    a    d    a |
----+-------------------------------+
deu |<9981>   1    .    .    7    1 |
eng |    3<9979>   2    .    3    . |
fra |    .    1<9993>   2    1    3 |
ita |    1    4    4<9983>   .    8 |
nld |    4    7    .    2<9984>   3 |
spa |    .    5    1    2    .<9992>|
----+-------------------------------+
(row = reference; col = test)
```