
Basic Text processing

- Sentence segmentation
- Tokenization
- Lemmatization/normalization

Sentence segmentation

The Standard & Poor's 500 (SPX) index, after being down 7.7% to its lowest point since Sept. 12, 2003, was down 3.9%. The Nasdaq composite (COMP) was down 4.3% after falling 8% to its lowest point since Aug. 28, 2003.

Sentence segmentation

The Standard & Poor's 500 (SPX) index, after being down 7.7% to its lowest point since Sept. 12, 2003, was down 3.9%. The Nasdaq composite (COMP) was down 4.3% after falling 8% to its lowest point since Aug. 28, 2003.

NLTK

- A Python natural language toolkit that has a lot of useful NL functions
- Before using it:
 - > cp ~/xuen/hw4.txt .
- To use NLTK
 - >>> import nltk, re, pprint
 - >>> infile = open("hw4.txt", "r")
 - >>> lines = infile.readlines()
 - >>> line = lines[0]

NLTK sentence segmentation

```
>>> s_segmenter = \
    nltk.data.load('tokenizers/punkt/english.pickle')
>>> sents = s_segmenter.tokenize(line)
>>> sents
>>> pprint.pprint(sents)
>>> sent = sents[3]
>>>
```

Tokenization with NLTK

```
>>> words = sent.split()
```

- Tokenization with regular expression

```
>>> pattern = r'\w+|[\^\w\s]+'
```

```
>>> nltk.tokenize.regexp_tokenize(sent,  
pattern)
```

Tokenization

- Lorrillard Inc., the unit of New York-based Loews Corp. that makes Kent cigarettes, stopped using crocidolite in its Micornite cigarette filters in 1956.

Tokenization

- Lorrillard Inc. , the unit of New York - based Loews Corp. that makes Kent cigarettes , stopped using crocidolite in its Micornite cigarette filters in 1956 .

Tokenization

- Typically, money-fund yields beat comparable short-term investments because portfolio managers can vary maturities and go after highest rates.

Tokenization

- Typically , money-fund yields beat comparable short-term investments because portfolio managers can vary maturities and go after highest rates .

Tokenization

- Sales of medium-sized cars, which benefited from price reductions arising from introduction of the consumption tax, more than doubled to 30,841 units from 13,056 in October 1988.

Tokenization

- Sales of medium-sized cars, which benefited from price reductions arising from introduction of the consumption tax, more than doubled to 30,841 units from 13,056 in October 1988.

Tokenization

- The document also said that although the 64-year-old Mr. Cray has been working on the project for more than six years, the Cray-3 machine is at least another year away from a fully operational prototype.

Tokenization

- The document also said that although the 64-year-old Mr. Cray has been working on the project for more than six years , the Cray-3 machine is at least another year away from a fully operational prototype .

Tokenization in Penn Treebank

■ English

- In the new position he will oversee Mazda 's U.S. sales , services , parts and marketing operations .
- We did n't have much of a choice .
- U.S. trade officials said the Philippines and Thailand would be the main beneficiaries of the president 's action .
- Anything 's possible -- how about the new Guinea Fund ?

Tokenization in Penn Treebank

■ English

- In the new position he will oversee Mazda 's U.S. sales , services , parts and marketing operations .
- We did n't have much of a choice .
- U.S. trade officials said the Philippines and Thailand would be the main beneficiaries of the president 's action .
- Anything 's possible -- how about the new Guinea Fund ?

Tokenization in Penn Treebank

- The federal government suspended sales of the U.S. savings bonds because Congress has n't lifted the ceiling on government debt .
- The Treasury said the U.S. will default on Nov. 9 if Congress does n't act by then .

Tokenization in Penn Treebank

- The federal government suspended sales of the U.S. savings bonds because Congress has **n't** lifted the ceiling on government debt .
- The Treasury said the U.S. will default on Nov. 9 if Congress does **n't** act by then .

Tokenization in Penn Treebank

- Assets of the 400 taxable funds grew by \$ 1.5 billion during the latest week .
- Exports in October stood \$ 5.29 billion , a mere 0.7 % increase from a year earlier , while imports increased sharply to \$ 5.39 billion , up 20 % from last year .
- Do you notice any ambiguity in tokenization?

Tokenization in Penn Treebank

- Assets of the 400 taxable funds grew by \$ 1.5 billion during the latest week .
- Exports in October stood \$ 5.29 billion , a mere 0.7 % increase from a year earlier , while imports increased sharply to \$ 5.39 billion , up 20 % from last year .
- Do you notice any ambiguity in tokenization ?

Big deal, you say

- The problem is pushed to the forefront for languages like Chinese, where there are no delimiting spaces between words

这句话里有几个词？

How many words are there in this sentence?

Big deal, you say

- The problem is pushed to the forefront for languages like Chinese, where there are no delimiting spaces between words

zhe ju hua li you ji ge ci
这 句 话 里 有 几 个 词 ？

this CL sentence inside have [how many] CL word ?

How many words are there in this sentence ?

A much harder problem than it first appears...

- Well, what if we just create a list of words (a dictionary) and compare the sentence against this list?

- 日文章鱼怎么说？

Dictionary entries: 日 “Sun”, 日文 “Japanese”, 文章 “article”, 章鱼 “octopus”, 鱼 “fish” 怎么 “how” 说 “say”

A much harder problem than it first appears...

- Well, what if we just create a list of words (a dictionary) and compare the sentence against this list?
- 日文 章鱼 怎么说?
Japanese Octopus how say
How do you say octopus in Japanese?
- 日 文章 鱼 怎么说?
Sun article fish how say
???

Computer problem vs human problem

- Well that may be a problem for the computer because the computer is dumb...
- Segmentation is difficult for humans as well
 - What is a word?
 - Different criteria do not coincide

What if we let native speakers follow their intuitions?

- Inadequate level of inter-annotator agreement
 - Sproat, 1996: 70%
 - Xue et al, 2005: 90%
- Conclusion: need a linguistic definition of wordhood to develop segmentation standards

HW4

- Compile a dictionary out of a corpus. In a real dictionary, there are words and definitions. Our dictionary only has words and their frequencies. You want to make sure that words are lemmatized, i.e., 'worked', 'working', 'works', 'work' are one word, not four. You are also asked to sort the dictionary by frequency, with most frequent words listed first. Make sure you output both the words and their frequencies. The corpus will be provided to you.

Regular expressions in Python

```
>>> import re
>>> p = re.compile(" +")
>>> line = "this is a test  a  test a  test"
>>> line.split(' ')
>>> ?
>>> p.split(line)
>>> ?
>>> line.replace(' ', '=')
>>> ?
>>> p.sub('=', line)
>>> ?
>>> p.subn('=', line)
```

More regular expression functions

- `p.match(string)`
 - Matches `p` from the beginning of `string`
- `p.search(string)`
 - Matches `p` anywhere in `string`, stops at first match
- `p.findall(string)`
 - Return all matched strings in a list

An example

```
>>> line = "These are test1 test2 test3"
>>> p2 = re.compile(r"test\d")
>>> if p2.search(line):
        print p2.search(line).group()
>>> ?
>>> p2.findall(line)
>>> ?
```

Escape! Escape!

```
>>> string1 = "\\section"  
>>> p3 = re.compile("\\\\section")  
>>> p3.search(string1).group()  
>>> ?  
>>> p3 = re.compile("\\\\\\\\section")  
>>> p3.search(string1).group()  
>>> ?  
>>> p3.search(string1).group()  
>>> p3 = re.compile(r'\\section')  
>>> ?
```

Grouping

```
>>> line = "the 10-12 class"
>>> p=re.compile(r"((\d+)-(\d+))")
>>> m=p.search(line)
>>> m.group()
>>> ?
>>> m.group(0)
>>> ?
>>> m.group(1)
>>> ?
```

Grouping

```
>>> m.group(2)
```

```
>>> ?
```

```
>>> m.group(3)
```

```
>>> ?
```

```
>>> m.group(4)
```

```
>>> ?
```

```
>>> m.group(3, 2)
```

```
>>> ?
```

More methods on Match Object

```
>>> m.groups()
```

```
>>> ?
```

```
>>> m.start(3)
```

```
>>> ?
```

```
>>> m.end(3)
```

```
>>> ?
```

```
>>> m.span(3)
```

```
>>> ?
```

Back references

```
>>> p.sub(r'\3-\2', line)
>>> ?
>>> line1 = "this is a test test"
>>> p1 = re.compile(r'(\b\w+)\s+\1')
>>> m1 = p1.search(line1)
>>> m1.group()
>>> ?
>>>
```

Give names to groups

```
>>> line3 = "Python is great!"
>>> p3 = re.compile(r'(?P<word>\b\w+)\w\b')
>>> m3 = p3.search(line3)
>>> m3.group(1)
>>> ?
>>> m3.group('word')
>>> ?
```

Greedy and non-greedy match

```
>>> line4 = "my Sentence One. My Sentence Two."  
>>> p4 = re.compile(r' .+\.' )  
>>> m4 = p4.search(line4)  
>>> m4.group()  
>>> ?  
>>> p5 = re.compile(r' .+?\.' )  
>>> ?  
>>> m5 = p5.search(line4)  
>>> m5.group()  
>>> ?
```

Case-sensitive and -insensitive match

```
>>> line5 = "Python is not python"
```

```
>>> p6 = re.compile('Python')
```

```
>>> p6.findall(line5)
```

```
>>> ?
```

```
>>> p7 = re.compile('Python', re.I)
```

```
>>> p7.findall(line5)
```

```
>>> ?
```

Special characters

- |
- ^
- \$
- \A: matches only at the beginning of the string
- \z: matches only at the end of a string
- \b: matches only at beginning or end of a word
- \B: matches only not at beginning or end of a word