

Natural Language Toolkit

prezentace do předmětu PA154

Nástroje pro korpusy

část 1 – možnosti NLTK

Stručná charakteristika

- NLTK je sada knihoven pro Python a programů pro symbolické a statistické zpracování přirozeného jazyka
- k dispozici jsou
 - zdrojové kódy
 - dokumentace
 - tutoriály

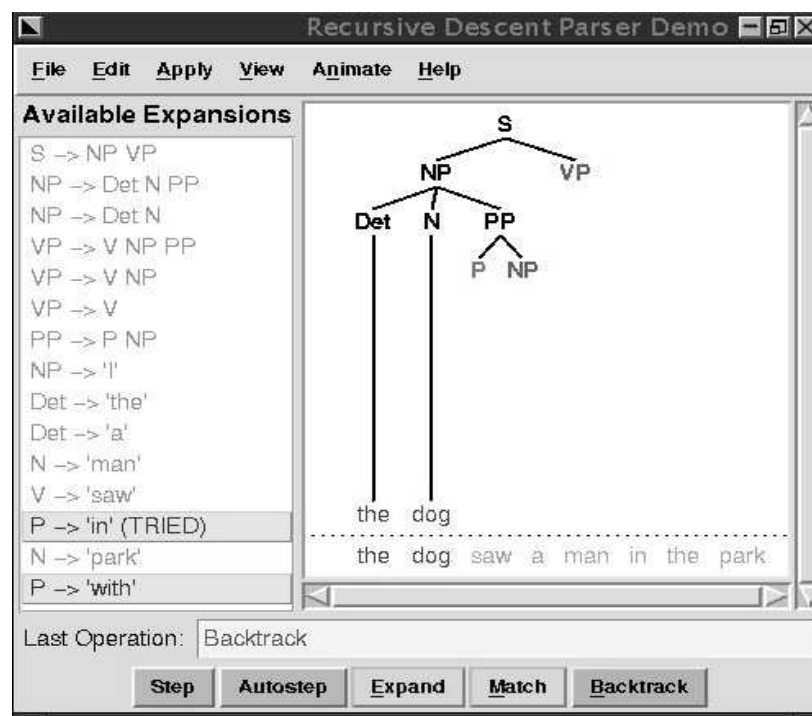
NLTK je určeno

- pro studenty zpracování přirozeného jazyka
- pro podporu výzkumu souvisejících oblastí, například:
 - empirická lingvistika (korpora)
 - kognitivní vědy
 - umělá inteligence, strojové učení
 - vyhledávání znalostí

Motivační příklad 1

nltk_lite.draw.rdparser

- Ukázka rekurzivní sestupné analýzy shora dolů



Motivační příklad 2

nlk_lite.draw.srparser

- Ukázka posuvně-redukční analýzy zdola nahoru

The screenshot shows a window titled "Shift Reduce Parser Demo" with a menu bar (File, Edit, Apply, View, Animate, Help). The main area is divided into three sections:

- Available Reductions:** A list of grammar rules. The rule "N -> 'man'" is highlighted.
- Stack:** A diagram showing the current state of the stack. The top element is "NP", which is expanded to "Det N". Below "Det" is "my" and below "N" is "dog". To the right of the stack are "V" (saw) and "Det" (a), followed by "man".
- Remaining Text:** The text "in the park with a statue" is shown.

At the bottom, a status bar indicates "Last Operation: Reduce: N -> 'man'", and there are buttons for "Step", "Shift", "Reduce", and "Undo".

Motivační příklad 3

`nltk_lite.draw.chart`

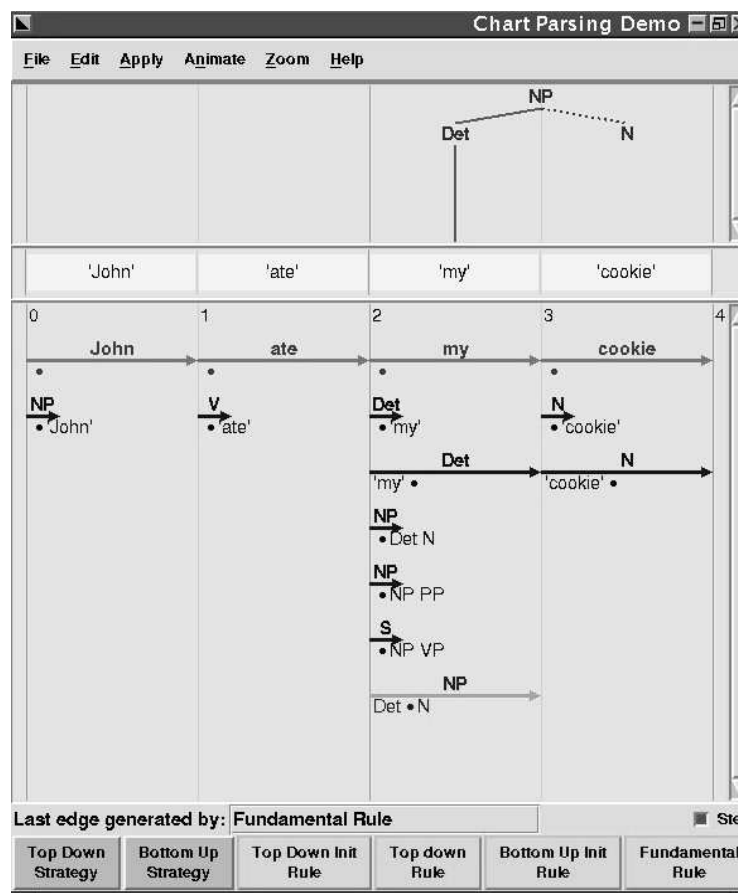
- Analýza zdola nahoru může najít jen jedno vyhodnocení, někdy nenalezne existující řešení
- Analýza shora dolů může být značně neefektivní (pro LR gramatiky může cyklit)
- Řešíme znovuužitím výpočtů (dynamické programování) -> chart parsing

Motivační příklad 3

nlk_lite.draw.chart

- Ukázka

tabulková
analýza



Motivační příklad 3

`nlTK_lite.draw.chart`

- Můžeme uložit jakoukoli hypotézu kompatibilní s gramatikou (ale nemusíme ji potom využít)
- Každá hypotéza reprezentována hranou

NLTK - lite

- vývoj: červen – prosinec 2005
- od prosince 2005 je to jediná podporovaná verze, proto se budeme zabývat právě jí
- stejná funkčnost jako klasické NLTK, avšak s nižšími nároky na programátora (používá standardní objekty Pythonu, atd.)

Autoři a licence

- autoři: Steven Bird, Edward Loper
- mnoho přispěvatelů
- licence:
 - projekt je open source bez záruky
 - GNU General Public License
 - dokumentace
 - Creative Commons Attribution-ShareAlike 2.5 License



OpenNLP



- organizační centrum projektů zabývajících se zpracování přirozeného jazyka
- soustřed'uje
 - projekty
 - užitečné odkazy
 - diskuzní fórum
- <http://opennlp.sourceforge.net/>

Instalace (1)

- Instalace vyžaduje Python 2.4 a vyšší
- Platformy
 - Linux
 - Mac
 - Windows

Instalace (2)

1) Python

<http://www.python.org/download/>

2) Numerical Python (Numarray)

http://sourceforge.net/project/showfiles.php?group_id=1369&package_id=32367

3) NLTK lite

http://sourceforge.net/project/showfiles.php?group_id=30982&package_id=156043

4) NLTK lite corpora

http://prdownloads.sourceforge.net/nltk/nltk_lite-corpora-0.6.3.zip

Python a NLP

- Python je vhodný nástroj pro NLP
 - jednoduchý
 - snadno “debugovatelný”
 - výjimky
 - interpretovaný jazyk
 - strukturovatelný
 - moduly, OOP
 - výkonná práce nad (znakovými) řetězci

Moduly a balíky

- moduly *modules* umožňují znovu použít kód
- balíky *packages* jsou hierarchické *moduly*
- *příkazy pro práci*
 - *import*
 - *from ... import*
 - *reload*

Moduly a balíky

import

•Příkaz *import* načítá modul:

```
# Load the regular expression module
```

```
>>> import re
```

•Použití přístupu k metodám (pomocí tečkové notace)

```
# Use the search method from the re module
```

```
>>> re.search('\w+', str)
```

•Zobrazení obsahu modulu pomocí *dir*:

```
>>> dir(re)
```

```
['DOTALL', 'I', 'IGNORECASE', ...]
```


Moduly a balíky

from .. import

•Příkaz *from...import* načítá jednotlivé funkce:

```
# Load the search function from the re module
>>> from re import search
>>> nltk_lite.draw.rdparser import *
```

•Poté již může být příkaz použit přímo:

```
# Use the search method from the re module
>>> search('\w+', str)
>>> demo()
```

Moduly NLTK-lite

- nltk_lite
- nltk_lite.chat
- nltk_lite.contrib
- nltk_lite.corpora
- nltk_lite.draw
- nltk_lite.misc
- nltk_lite.model
- nltk_lite.parse
- nltk_lite.tag
- nltk_lite.tokenize

Natural Language Toolkit

prezentace do předmětu PA154

Nástroje pro korpusy

část 2 – nástroje NLTK

Tokenizace

úvod

- Co je slovo?
 - Shluk znaků oddělený mezerou? NE
 - Konce řádků
 - Interpunkce
 - ...
- Rozdíl *Type* vs. *Token*
 - Type – to co je mnoha tokenům společné
 - Token - konkrétní realizace znaku

*"slovo" se vyskytuje dvakrát (dva tokeny),
ale jde jen o jedno slovo (jeden type)*

Tokenizace

text = sekvence tokenů

```
>>> from nltk_lite import tokenize
>>> text = 'Hello world. This is a test string.'
>>> list(tokenize.whitespace(text))
['Hello', 'world.', 'This', 'is', 'a', 'test', 'string.']

>>> text = 'That poster costs $22.40.'
>>> pattern = r'\w+|\$\d+\.\d+|[\^\w\s]+'
>>> list(tokenize.regexp(text, pattern))
['That', 'poster', 'costs', '$22.40', '.']
>>> list(tokenize.regexp(text, pattern=r'\s+', gaps=True))
['That', 'poster', 'costs', '$22.40.']

>>> from nltk_lite.corpora import brown, extract
>>> print extract(0, brown.raw('a'))
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an', 'investigation', 'of', 'Atlanta's', 'recent', 'primary', 'election', 'produced', 'no', 'evidence', 'that', 'any', 'irregularities', 'took', 'place', '.']
```

Tokenizace stemming (hledání kořene)

```
>>> porter = tokenize.PorterStemmer()  
>>> tokens = extract(0, brown.raw('a'))  
>>> for token in tokens:  
    print porter.stem(token)
```

The Fulton Counti Grand Juri said Friday an investig of Atlanta' recent primari elect produc `` no evid '' that ani irregular took place .

Tokenizace statistiky

Počet slov

```
>>> from nltk_lite.corpora import genesis
>>> len(list(genesis.raw('english-kjv')))
38240
>>> len(list(genesis.raw('finnish')))
26597
>>> |
```

Frekvenční distribuce

```
>>> from nltk_lite.probability import FreqDist
>>> fd = FreqDist()
>>> for token in genesis.raw():
>>>     fd.inc(token)

>>> fd.max()
'the'
```

Tokenizace statistiky

`nlTK_lite.probability.FreqDist`

Jméno	Příklad	Popis
Count	<code>fd.count('the')</code>	Kolikrát se daný vzorek vyskytl
Frequency	<code>fd.freq('the')</code>	Frekvence daného vzorku
N	<code>fd.N()</code>	Počet vzorků
Samples	<code>fd.samples()</code>	Seznam různých zaznamenaných vzorků
Max	<code>fd.max()</code>	Vzorek s nejvyšším počtem výskytů

Tokenizace

statistika pomocí tokenů

```
>>> def length_dist(text):
    fd = FreqDist()
    for token in genesis.raw(text):
        fd.inc(len(token))
    for i in range(15):
        print "%2d" % int(100*fd.freq(i)),
    print

>>> length_dist('english-kjv')
 0  2 14 28 21 13  7  5  2  2  0  0  0  0  0
>>> length_dist('finnish')
 0  0  9  6 10 16 16 12  9  6  3  2  2  1  0
```

Tokenizace podmíněná pravděpodobnost

```
>>> from nltk_lite.probability import ConditionalFreqDist
>>> cfdist = ConditionalFreqDist()
>>> for text in genesis.items:
    for word in genesis.raw(text):
        cfdist[text].inc(len(word))
>>> for cond in cfdist.conditions():
    wordlens = cfdist[cond].samples()
    wordlens.sort()
    points = [(i, cfdist[cond].freq(i)) for i in wordlens]
```

```
>>> from nltk_lite.draw.plot import Plot
>>> Plot(points).mainloop()
```



Tokenizace

předpovídání slov (kolokace)

- Trénovací korpus

- ConditionalFreqDist a max()

```
>>> from nltk_lite.probability import ConditionalFreqDist
>>> cfdist = ConditionalFreqDist()
>>> prev = None
>>> for word in genesis.raw():
        cfdist[prev].inc(word)
        prev = word
```

```
>>> word = 'living'
>>> cfdist['living'].samples()
['creature,', 'substance', 'soul.', 'thing', 'thing,', 'creature']
```

- Generování

```
>>> word = 'better'
>>> for i in range(20):
        print word,
        word = cfdist[word].max()
```

```
better that he said, I will not be a wife of the land of the land
```

Tagování nástroje NLTK tag

```
>>> sent = """  
John/nn saw/vb the/at book/nn on/in the/at table/nn ./end He/nn s  
ighed/vb ./end  
"""  
  
>>> from nltk_lite.tag import tag2tuple  
>>> for t in tokenize.whitespace(sent):  
    print tag2tuple(t),  
  
( 'John', 'nn') ( 'saw', 'vb') ( 'the', 'at') ( 'book', 'nn') ( 'on', '  
in') ( 'the', 'at') ( 'table', 'nn') ( '.', 'end') ( 'He', 'nn') ( 'sig  
hed', 'vb') ( '.', 'end')
```

Word Class Label	Brown Tag	Word Class
Det	at	Article
N	nn	Noun
V	vb	Verb
Adj	jj	Adjective
P	in	Preposition
Card	cd	Number
	end	Sentence-ending punctuation

Tagování nástroje NLTK tag

- Jednoduchý tagger

```
>>> text = "John saw 3 polar bears ."
>>> tokens = list(tokenize.whitespace(text))
>>> ['John', 'saw', '3', 'polar', 'bears', '.']
['John', 'saw', '3', 'polar', 'bears', '.']
>>> my_tagger = tag.Default('nn')

>>> from nltk_lite import tag
>>> my_tagger = tag.Default('nn')
>>> list(my_tagger.tag(tokens))
[('John', 'nn'), ('saw', 'nn'), ('3', 'nn'), ('polar', 'nn'), ('bears', 'nn'), ('.', 'nn')]
```

- Přesnost cca. 20-30%
- Používá se jako *fallback solution*

Tagování nástroje NLTK tag

- Tagger s regulárními výrazy

```
>>> text = "John saw 3 polar bears ."  
>>> tokens = list(tokenize.whitespace(text))  
>>> ['John', 'saw', '3', 'polar', 'bears', '.']  
['_', 'John', 'saw', '3', 'polar', 'bears', '.']  
>>> patterns = [(r'^-?[0-9]+(.[0-9]+)?$', 'cd'), (r'.*', 'nn')]  
>>> nn_cd_tagger = tag.Regexp(patterns)  
>>> list(nn_cd_tagger.tag(tokens))  
[('John', 'nn'), ('saw', 'nn'), ('3', 'cd'), ('polar', 'nn'), ('be  
ars', 'nn'), ('.', 'nn')]
```

- Vhodný pro slovní tvary s charakteristickou příponou/předponou
- Vhodný pro čísla, mailové adresy, www stránky apod.
- Používá se jako *fallback solution*

Tagování nástroje NLTK tokenize

- Unigramový tagger

- Trénování:

```
>>> from nltk_lite.corpora import brown
>>> from itertools import islice
>>> train_sents = list(islice(brown.tagged(), 500))
>>> unigram_tagger = tag.Unigram()
>>> unigram_tagger.train(train_sents)
```

- Použití:

```
>>> text = "John saw the book on the table"
>>> tokens = list(tokenize.whitespace(text))
>>> list(unigram_tagger.tag(tokens))
[('John', 'np'), ('saw', 'vbd'), ('the', 'at'), ('book', None), ('on', 'in'), ('the', 'at'), ('table', None)]
>>> unigram_tagger = tag.Unigram(backoff=nn_cd_tagger)
>>> unigram_tagger.train(train_sents)
>>> list(unigram_tagger.tag(tokens))
[('John', 'np'), ('saw', 'vbd'), ('the', 'at'), ('book', 'nn'), ('on', 'in'), ('the', 'at'), ('table', 'nn')]
```

Tagování nástroje NLTK tag

- Kombinace

```
>>> t0 = tag.Default('nn')
>>> t1 = tag.Unigram(backoff=t0)
>>> t2 = tag.Bigram(backoff=t1)
>>> t0 = tag.Default('nn')
>>> t1 = tag.Unigram(backoff=t0)
>>> t2 = tag.Bigram(cutoff=2, backoff=t1)
>>> t1.train(brown.tagged('a'))
>>> t2.train(brown.tagged('a'))
>>> accuracy2 = tag.accuracy(t2, brown.tagged('b'))

>>> print 'Bigram Accuracy = %4.1f%%' % (100 * accuracy2)
Bigram Accuracy = 79.3%
```

- Brillův tagger

```
>>> from nltk_lite.tag import brill
>>> brill.demo()
```


Chunk parsing nástroje NLTK parse

- Příklad

```
>>> from nltk_lite.parse import tree
>>> tree.chunk("[ the/DT little/JJ cat/NN ] sat/VBD on/IN [ the/DT mat/NN ]":
(S: (NP: ('the', 'DT') ('little', 'JJ') ('cat', 'NN')) ('sat', 'VBD') ('on',
'IN') (NP: ('the', 'DT') ('mat', 'NN'))))
```

- Shlukování tokenů do *chunks*

- Tvořeny vedoucím slovem (např. podst. jm)
a souvisejícími slovy (např. příd. jm.)

- *Chunks* a uplynulé tokeny
vytváří tzv. *chunk structure*

- Dvojúrovňový strom obsahující celý text a
obsahující jak *chunks* tak neparsované tokeny

Chunk parsing nástroje NLTK parse

- Chunk tree

```
>>> from nltk_lite.corpora import treebank, extract
>>> chunk_tree = extract(603, treebank.chunked())
>>> print chunk_tree
(S:
 ('In', 'IN')
 (NP: ('happier', 'JJR') ('news', 'NN'))
 (',', ',')
 (NP: ('South', 'NNP') ('Korea', 'NNP'))
 (',', ',')
 ('in', 'IN')
 ('establishing', 'VBG')
 (NP: ('diplomatic', 'JJ') ('ties', 'NNS'))
 ('with', 'IN')
 (NP: ('Poland', 'NNP') ('yesterday', 'NN'))
 (',', ',')
 ('announced', 'VBD')
 (NP: ('$ ', '$ ') ('450', 'CD') ('million', 'CD'))
 ('in', 'IN')
 (NP: ('loans', 'NNS'))
 ('to', 'TO')
 (NP: ('the', 'DT'))
 ('financially', 'RB')
 ('strapped', 'VBN')
 (NP: ('Warsaw', 'NNP') ('government', 'NN'))
 ('.', '.')
```

Shrnutí

- NLTK je vhodný nástroj pro NLP:
 - Umožňuje rychlou a pohodlnou práci s textem
 - Mnoho obsažených výrazů

Literatura

- **NLTK-Lite Tutorials**
Steven Bird, Ewan Klein, Edward Loper, 2001-2006
 - <http://nltk.sourceforge.net/lite/doc/en/>
- **Getting Started with NLTK**
 - http://nltk.sourceforge.net/getting_started.html
- **nltk_lite API**
 - <http://nltk.sourceforge.net/lite/doc/api/>