

POS-Tagging mit dem Natural Language Toolkit

Arne Neumann

Universität Potsdam

27. Mai 2008

NLTK: Überblick

- Sammlung von Open-Source-Paketten (Python) zur Verarbeitung natürlicher Sprache
- enthält zudem etliche Korpora, eine umfangreiche Dokumentation & viele Übungen
- Ausrichtung auf Forschung **und** Lehre

NLTK: Bestandteile

- **Funktionen:** Tokenizer, Stemmer, Tagger, Chunker, Parser, Wordnet-Anbindung
- **Korpora:** mehr als 30 annotierte Datensätze (ca. 300MB)
- **Dokumentation:** knapp 400-seitige Einführung, wissenschaftliche Artikel, API-Dokumentation

NLTK: Installation

- benötigt werden: Python \geq 2.4, NLTK und NLTK-Data (Korpora)

<http://nltk.org/index.php/Installation>

- **Windows:** nltk-0.9.2.win32.exe und nltk-data-0.9.2.zip
- **Mac OS X:** nltk-0.9.2.dmg (NLTK und NLTK-Data)
- **Debian/Ubuntu:** python-pywordnet, python-nltk und python-nltk-data aus dem Ubuntu NLP-Repository
<http://cl.naist.jp/~eric-n/ubuntu-nlp/>

NLTK: Installation auf Helios nltk.notlong.com

in eurem Homeverzeichnis folgende Zeilen der Datei
`.bash_profile` hinzufügen:

```
PYTHONPATH=/home/timo/tmp/nltk/lib/  
python2.5/site-packages/  
NLTK_DATA=/home/timo/tmp/nltk/data  
export PYTHONPATH  
export NLTK_DATA
```

Datei speichern, schließen und neu einlesen lassen mit:
`source .bash_profile`

NLTK: Erste Schritte

in der Python-Shell:

```
>>> import nltk
>>> nltk.corpus.brown.words()

['The', 'Fulton', 'County', 'Grand', 'Jury',
'said', ...]

>>> help(nltk.corpus.brown)
```

The screenshot shows the 'Shift Reduce Parser Demo' window. On the left, a list of 'Available Reductions' includes rules like 'S -> NP VP', 'NP -> Det N', and 'N -> 'park'', with 'N -> 'park'' highlighted. The main area is divided into 'Stack' and 'Remaining Text'. The 'Stack' contains a partial parse tree for 'my dog saw a man in the park' and the words 'in the'. The 'Remaining Text' contains 'with a statue'. The 'Last Operation' field shows 'Shift: 'park''. At the bottom, there are buttons for 'Step', 'Shift', 'Reduce', and 'Undo'.

Abbildung: Shift-Reduce-Parser: nltk.draw.srparser.demo()

Wordnet-Anbindung: nltk.wordnet.demo()

```
Enter a word: mice
```

```
Word is mice
```

```
Noun: mouse
```

```
Enter a word: clean
```

```
Word is clean
```

```
Noun: clean
```

```
Verb: clean
```

```
Adj: clean
```

```
Adv: clean
```

Korpora in NLTK: Übersicht

- **annotierte Korpora:** Brown (57340 Sätze), Penn Treebank (3914 Sätze), Switchboard (39 Anrufe)
- **unannotierte Korpora:** Gutenberg, Genesis
- **spezielle Korpora:** Stopwortlisten (11 Sprachen), Vornamen, Wordnet, Verbnets, Aussprachewörterbuch

Brown Corpus

```
>>> from nltk.corpus import brown #spart Tipparbeit  
  
>>> for i in brown.words()[0:100]: print i  
  
( 'The', 'AT' )  
( 'Fulton', 'NP-TL' )  
( 'County', 'NN-TL' )  
( 'Grand', 'JJ-TL' )  
( 'Jury', 'NN-TL' )  
( 'said', 'VBD' )  
...
```

Brown Corpus: Wortarten

```
# "alle" Adverbien ausgeben
>>> for (word,tag) in brown.tagged_words()[0:10000]:
        if tag == 'RB':
            print word
```

das komplette Brown Corpus Tagset gibt's hier:

[http://www.scs.leeds.ac.uk/ccalas/tagsets/
brown.html](http://www.scs.leeds.ac.uk/ccalas/tagsets/brown.html)

Unigramm-Tagger

Welches Tag t wurde in den bereits annotierten Trainingssätzen für das Wort w am häufigsten vergeben?

$$\text{tag}(w) = \arg \max_{t_i} P(t_i | w)$$

Dieser Tag t wird von nun an **jedem** Vorkommen des Wortes w zugeordnet. Position und Kontext des Wortes wird **nicht** berücksichtigt.

Unigramm-Tagger trainieren

- Tagger auf 5000 annotierten Sätzen trainieren, dann auf eigenen Satz anwenden
- **Problem:** unbekannte Wörter bekommen kein Tag zugewiesen

```
>>> tagger = nltk.UnigramTagger  
      (brown.tagged_sents()[0:5000])  
>>> tagger.tag  
      ('The horse crashed into George Bush'.split())  
  
[('The', 'AT'), ('horse', 'NN'), ('crashed', None),  
( 'into', 'IN'), ('George', 'NP'), ('Bush', 'NP')]
```

Ngramm-Tagger

- Bigramm: der Tag des vorhergehenden Wortes wird berücksichtigt
- Ngramm: die Tags der $n-1$ vorhergehenden Worte werden berücksichtigt

MaxEnt-Tagger

Maximum Entropy

Sprachmodell baut auf mehreren Iterationen über das Trainingskorpus auf. Hierbei werden nur tatsächliche vorkommende Wahrscheinlichkeiten berücksichtigt.

“model all that is known and assume nothing about that which is unknown” (Berger et al. 1996)

- MXPOST: Vorschlag von Ratnaparkhi (1996), implementiert in Java (Freeware, closed source)
- weitere Implementierungen: ACOPOST (C++), Maximum Entropy Modeling Toolkit (C++ mit Python-Bindings)
- mein Projekt: MaxEnt-Implementierung in Python für das NLTK-Projekt

Quellen

RATNAPARKHI, ADWAIT. *A Maximum Entropy Model for Part-Of-Speech Tagging*. 1996

BERGER AND DELLA PIETRA AND DELLA PIETRA. *A Maximum Entropy Approach To Natural Language Processing*. 1996

nltk.org/doc/guides/tag.html

nltk.org/doc/slides/tag.pdf

homepages.inf.ed.ac.uk/s0450736/maxent.html