Programmierkurs Python

Michaela Regneri & Stefan Thater
Universität des Saarlandes
FR 4.7 Allgemeine Linguistik (Computerlinguistik)

Winter 2010/11

Übersicht

- Das Programmierprojekt
  - Aufgabenstellung
  - Praktische Hinweise
- Beispiellösungen zum aktuellen Übungsblatt

Programmierprojekt

- Wir implementieren eine kleine Suchmaschine:
  - Es sollen Suchanfragen formuliert werden können, die mit den üblichen Boolschen Operatoren verknüpft sein können.
  - Das Ergebnis soll eine Liste von Dokumenten sein, die die entsprechende Suchanfrage erfüllen.
  - Nur relevante Wörter bzw. Dokumente
- Die Dokumentsammlung: Das Gigaword-Korpus (NYT).

Zwei Teilaufgaben

- Erstellen eines Index über der Dokumentsammlung.
  - Der Index soll für jedes Wort eine Liste von Dokumenten enthalten, in denen das entsprechende Wort vorkommt.
- Implementieren einer Funktion, mit der man Suchanfragen über dem Index formulieren kann.
  - Das Programm soll so aufgebaut sein, dass man Suche und Index-Erstellung getrennt voneinander benutzen kann

Relevanz

- Bei der Suche nach Dokumenten, die ein bestimmtes Wort enthalten, ist man normalerweise vor allem an „relevanten“ Dokumenten interessiert.
- Einfaches Relevanzmaß: Wort-Häufigkeit
  - Problem: es wird nicht zwischen Wörtern, die insgesamt häufig vorkommen („ein“, „der“, ...), und Wörtern, die nur in einzelnen Dokumenten häufig vorkommen, unterschieden.
- Besseres Relevanzmaß: tf-idf

tf-idf

- Der Index soll pro Dokument nur die n relevantesten Wörter im Dokument enthalten
- Zur Bestimmung der Relevanz verwenden wir tf-idf
  - $n_{i,j}$ = Anzahl der Vorkommen des Worts w_i im Dokument d_j
  - D = Menge der Dokumente
- $\text{tf-idf}_{i,j} = \text{tf}_{i,j} * \text{idf}_i$
  - Eine hoher td-idf Wert ergibt sich für Wörter mit hoher Termfrequenz und niedriger Dokumentfrequenz.

$$\text{idf}_i = \log \frac{|D|}{|\{d : t_i \in d\}|}$$

$$\text{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

Das Gigaword-Korpus

- Wir betrachten das NYT-Teilkorpus („New York Times“)

```
<DOC id="NYT19941101.0001" type="story" >
<HEADLINE>
SIMPSON JURY POOL 60 PERCENT AFRICAN-AMERICAN
</HEADLINE>
<DATELINE>
LOS ANGELES  (BC-SIMPSON-TRIAL-2Takes-LADN)
</DATELINE>
<TEXT>
<P>
After dismissing four more jurors Monday for
failing to heed the judge's order to avoid all media, [...]
</P>
[...]
</TEXT>
</DOC>
```


Das Gigaword-Korpus (DTD)

```
<!ELEMENT DOC (HEADLINE*, DATELINE*, TEXT*)>
<!ATTLIST DOC
  id ID #REQUIRED
  type (advis|multi|other|story) #REQUIRED >
<!ELEMENT HEADLINE (#PCDATA)>
<!ELEMENT DATELINE (#PCDATA)>
<!ELEMENT TEXT (#PCDATA | P)*>
<!ELEMENT P (#PCDATA)>
<!ENTITY AMP "&amp;" >
```

Das Gigaword-Korpus

- Wir betrachten das NYT-Teilkorpus („New York Times“)
- Dokumente sind nach Jahrgang und Monat in einzelnen komprimierten Dateien zusammengefasst.
  - Dateien können mit gzip.open(...) geöffnet werden.
- Das Format ist leider keine wohlgeformtes XML, man kann den SAX-Parser nicht direkt verwenden.
  - Wurzelelement fehlt
  - Nicht deklarierte Entity: &
  - Ein kleiner Trick hilft ... (⇒ nächste Folie)

Parsen von Gigaword-Dateien

```
parser = xml.sax.make_parser()
parser.setContentHandler(MyHandler(docs))
parser.feed('<!DOCTYPE DOCS [<!ENTITY AMP "&amp;" >]>')
parser.feed('<DOCS>')

with closing(gzip.open(filename)) as f:
    for line in f:
        parser.feed(line)

parser.feed('</DOCS>')
parser.close()
```

Große Wörterbücher

- Häufig speichert man seine Daten in Wörterbüchern (dict).
- Wenn sehr viele Daten gespeichert werden müssen, bietet es sich an, stattdessen mit „`shelve`“ Objekten zu arbeiten
- Ein `shelve` ist gewissermaßen ein persistentes Wörterbuch, das auf der Festplatte gespeichert ist.
- Einschränkungen: nur Strings als Schlüssel
  - Werte könne beliebige Python-Objekte sein

Wörter zählen: `dict`

```
import sys
freqs = dict()
with open(sys.argv[1]) as f:
    for line in f:
        for word in line.split():
            try:
                freqs[word] += 1
            except KeyError:
                freqs[word] = 1

for word, freq in freqs.items():
    print('%d\t%s' % (freq, word))
```

Wörter zählen: shelve

```
import sys, shelve
freqs = shelve.open('tmp')
with open(sys.argv[1]) as f:
    for line in f:
        for word in line.split():
            try:
                freqs[word] += 1
            except KeyError:
                freqs[word] = 1

for word, freq in freqs.items():
    print('%d\t%s' % (freq, word))
```


shelve: Achtung

- Nachträgliches Ändern von veränderbaren (mutable) Werten ist nicht ohne Weiteres möglich:

```
>>> s = shelve.open('tmp')
>>> s['4'] = []
>>> s['4']
[]
>>> s['4'].append(1)
>>> s['4']
[]
```

- Mehr zu shelves:
  - http://docs.python.org/py3k/library/shelve.html

with expr [as variable]

- Dateien sollten immer geschlossen werden.
- Bequeme Syntax: das `with`-Statement

```
with open(filename) as f:
  for line in f:
    # ...
```

- Hier wird nach dem `with`-block die Datei automatisch geschlossen.
- `expr` muss zu einem Objekt auswerten, das die beiden Methoden `__enter__` und `__exit__` implementiert.

with closing(expr) as var

- Leider implementieren nicht alle (Datei-) Objekte die beiden Methoden `__enter__` und `__exit__`
  - können also nicht (direkt) mit with verwendet werden.
- Das contextlib-Modul implementiert einige nützliche Funktionen, die die fehlende Funktionalität „nachrüstet“.

```
from contextlib import closing
import gzip
with closing(gzip.open(filename)) as f:
  for line in f:
    # ...
```

Nützliche Unix-Werkzeuge: ssh

- Mit ssh kann man sich auf einen entfernten Rechner im Netzwerk anmelden:
  - ssh stth@login.coli.uni-saarland.de
- Kopieren von Daten:
  - scp <datei> stth@login.coli.uni-saarlande:<pfad>
  - scp stth@login.coli.uni-saarlande:<pfad> <pfad>

Nützliche Unix-Werkzeuge: screen

- Screen erlaubt es, mehrere virtuelle Terminal-Sitzungen in einem Terminal auszuführen.
- Mit „ctrl-a d“ kann man eine Screen-Sitzung „abtrennen“
- Mit „screen -r“ holt man sich eine abgetrennte Screen-Sitzung zurück.
- Nützlich bei zeitaufwändigen Berechnungen

Nützliche Unix-Werkzeuge: gzip

- Mit gzip (gunzip) kann man komprimierte Dateien dekomprimieren.
- gzip -dc <Datei> | less
  - dekomprimiert die Datei und zeigt sie mit „less“ an