Python für Linguisten

Dozentin: Wiebke Petersen & Valentin Heinz

8. Foliensatz

Zielsetzung für lawstats.py

- Korpus: verschiedene deutsche Gesetze im XML-Format, c.a. 400 000 Token
- Kookkurrenzen auflisten:
  - bigrammbasiert: Anfrage und nachfolgendes Token
  - satzbasiert: Anfrage und N umgebende Token
  - Häufigkeiten ausgeben
- Argumente entgegennehmen
- Ergebnisse formatiert ausgeben
- Ergebnisse in Datei speichern

Was müssen wir tun?

- Vorbereitung:
  - Dateien einlesen (XML, Encoding) und Korpus erstellen
  - Korpus säubern, Sätze segmentieren, Token segmentieren
  - Satz-ID vergeben und Sätze speichern
  - Paare finden und speichern (welche Datenstruktur?)
- Nutzung:
  - Anfrage entgegennehmen, Datensätze suchen
  - nach wortbasierter bzw. satzbasierter Ausgabe formatieren
  - Ergebnisse auf dem Terminal ausgeben und in eine Datei schreiben

Programmarchitektur

- modularisierte Architektur und DRY (Don't Repeat Yourself)
- Argumente einlesen: `make_options.py`
- häufig wiederkehrende, allgemeine Funktionen: `helper_functions.py`
- Korpus erstellen, säubern, segmentieren und als Objekt zurückgeben: `parse_corpus.py`
- Paare finden, speichern/lesen, formatieren, ausgeben: `lawstats.py`
- (Hinweis: es ist sauberer, die Ausgabe auch komplett von der Hauptdatei zu trennen)

Probleme: Geschwindigkeit

- welches Token wurde bereits abgelegt? dict.keys() bzw. dict.iterkeys() sind zu langsam, da das dictionary immer größer wird
- `had 20000 numbers and saw 12577 different numbers in 6.517426 seconds`
- Idee: `try: except:` verwenden. Annahme: der Wert ist schon da
- `had 20000 numbers and saw 12619 different numbers in 0.087225 seconds`
- Script siehe `https://python.inktrap.org/tryexceptperf.py`

Probleme: Speicherung, Ausgabe

- wie kann man die Ergebnisse (auf Anforderung: ohne Datenbank) so speichern, dass sie schnell durchsuchbar sind?
  - Objekte `picklen`: Die Einträge nach kleingeschriebenem ersten Buchstaben sortieren und in eine eigene Datei schreiben, dann selektiv laden.
- wie sollte eine flexible Ausgabe sein?
  - Eintragsanzahl begrenzen
  - Satznummern anzeigen bzw. nicht anzeigen
  - Dateiname wählbar
  - Kontext erweitern/begrenzen

Probleme: Satzsegmentierung

- Satzsegmentierung ist problematisch, **Satzteil**:
  (1a) Ordnungswidrig handelt, wer gegen die Verordnung (EG) Nr. 717/2007 des Europäischen Parlaments und des Rates vom 27. Juni 2007 über das Roaming in öffentlichen Mobilfunknetzen in der Gemeinschaft und zur Änderung der Richtlinie 2002/21/EG (ABl. L 171 vom 29.6.2007, S. 32), die durch die Verordnung (EG) Nr. 544/2009 (ABl. L 167 vom 29.6.2009, S. 12) geändert worden ist, verstößt, indem er vorsätzlich oder fahrlässig 1.als Betreiber eines besuchten Netzes dem Betreiber
- Lösungsansatz: negative look behind expression:

```
1 re.split(r'(?<!(
2 \) [a-zA-Z0-9]|ABl|. S|hst|chn|Abs|Art|..[0-9]|. d|vgl|. v| Nr)
3 ) [\.!?]', line
4 )
```

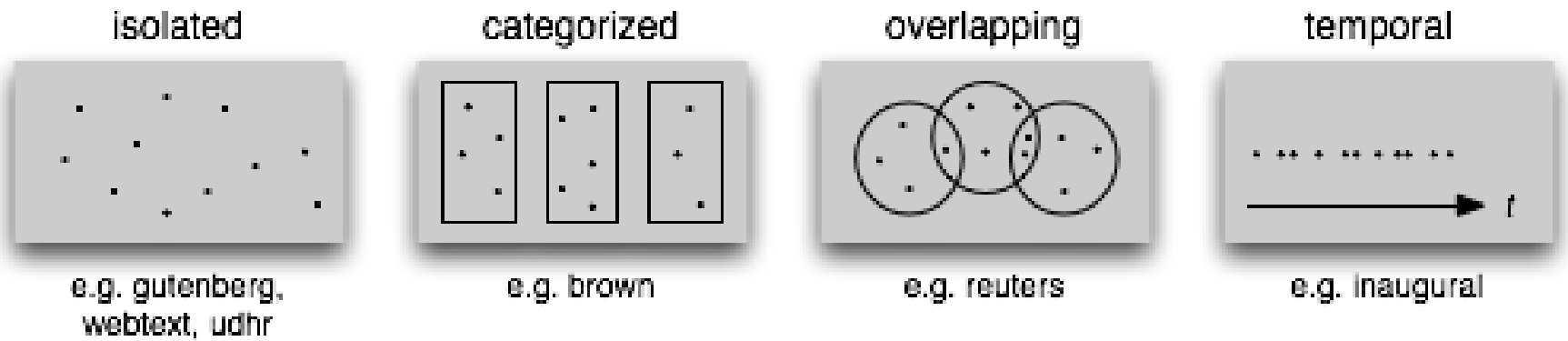
NLTK

- NLTK importieren: `>>> import nltk`
- einfache Texte nutzen, zB: `>>> from nltk.book import text1`
- Beispiel: Kookkurrenzen mit Kontext: `>>> text1.concordance('whale')`
- `>>> from nltk.book import text2`
- Kontext auswerten: `>>> text1.similar('monstrous')` *#bzw. text2*
- was fällt auf?

Plotting

- auch Plotting ist einfach möglich, zB Lexical Dispersion Plot:
- `>>> text1.dispersion_plot(['Moby', 'whale', 'sea', 'ship'])`
- zuhause: matplotlib und numpy installieren!

Korpustypen



Quelle: `http://nltk.org/book/ch02.html#fig-text-corpus-structure`

Korpora

- Beispiele: Brown Corpus, Gutenberg Corpus, Reuters Corpus
- Treebanks: Dependency Treebank, Penn Treebank (selections), Floresta Treebank
- multilingual: Genesis Corpus, Univ Decl of Human Rights
- monitoring/zeitlich: Inaugural Address Corpus
- gesprochene Sprache: Switchboard Corpus, TIMIT Corpus (selections)
- informelle Sprache: Chat-80-Corpus (Chatlogs), NPS Chat Corpus

- Nutzung: `>>> from nltk.corpus import brown`
- Raw: `>>> brown.raw()`
- Dateien: `>>> brown.fileids()`
- Kategorien: `>>> brown.categories`
- Worte: `>>> brown.words()`
- Sätze: `>>> brown.sents()`
- Typen: `>>> len(set([w.lower() for w in brown.words()]))`

- lade `http://python.inktrap.org/whole_corpus_raw.txt` herunter
- aktuelles Verzeichnis bestimmen: `>>> import os` und `os.path.abspath(os.curdir)`
- verschieben Sie die Datei in das Verzeichnis, in dem das Skript liegt
- PlaintextCorpusReader liest Textdateien:

```
1 from nltk.corpus import PlaintextCorpusReader
2 corpus_root = os.path.abspath(os.curdir)
3 corpus = PlaintextCorpusReader(corpus_root, 'whole_corpus_raw.txt')
```

Segmentierung

- Wie verhalten sich Satzsegmentierung und Tokenisierung?
- `>>> nltk.word_tokenize(corpus.raw()[:10000])`
- `>>> sent_tokenizer = nltk.data.load('tokenizers/punkt/german.pickle'`
- `>>> sent_tokenizer.tokenize(corpus.raw()[:10000])`
- vergleichen Sie dies mit `corpus/whole_corpus.txt` (Satz je Zeile, Token je Leerzeichen)
- es gibt natürlich noch andere Verfahren zur Satzsegmentierung/Tokenisierung

Frequency Distribution

- Zählen von Elementen:
- `>>> nltk.FreqDist(word_list)`
- nimmt eine Liste als Argument
- erstellt FreqDist-Objekt: ähnlich wie ein dictionary, nur sortiert.
- zählt Frequenzen, keine Wahrscheinlichkeiten!

Conditional Frequency Distribution

Condition: News

the	𝍸 𝍸 𝍸 ||||
cute	
Monday	𝍸 ||||
could	|
will	𝍸 |||

Condition: Romance

the	𝍸 𝍸 |||
cute	|||
Monday	|
could	𝍸 𝍸 𝍸
will	||||

http://nltk.org/images/tally2.png

- nimmt eine Liste von Tupeln als Argument.
  `[('la','le'),('lu','grml'),('grml','la')]`
- erstes Element: Bedingung, zweites Element: eintretendes Ereignis
- was kann man damit machen?

Beispiel: Conditional Frequency Distribution

- Bedingung: Kategorie.
- Ereignis: Modalverben

```
1 from nltk.corpus import brown
2 cfd = nltk.ConditionalFreqDist(
3     (genre, word)
4     for genre in brown.categories()
5     for word in brown.words(categories=genre)
6     )
7 genres = brown.categories()
8 modals = ['can', 'could', 'may', 'might', 'must', 'will']
9 cfd.tabulate(conditions=genres, samples=modals)
```

Quelle: `http://nltk.org/book/ch02.html#sec-conditional-frequency-distributions`