Korpuslinguistik

Literaturempfehlungen, Ressourcen, einschlägige Software

Literaturempfehlungen

- Scherer, Carmen. 2006. *Korpuslinguistik*. (Kurze Einführungen in Die Germanistische Linguistik 2). Heidelberg: Winter.
- Lemnitzer, Lothar & Heike Zinsmeister. 2015. *Korpuslinguistik. Eine Einführung*. 3rd ed. Tübingen: Narr.
- Stefanowitsch, Anatol. im Ersch. *Corpus linguistics. A guide to the methodology*. Berlin: Language Science Press.

Ressourcen und einschlägige Software

- für einfache Korpusrecherchen: AntConc
- für komplexere Korpusabfragen: CQP
- für Korpusannotation: GATE
- für Annotation und Auswertung von Konkordanzen: Tabellenkalkulationsprogramm (Excel, Calc)
- für alles mögliche: R und RStudio
- für Arbeit mit großen Datenmengen: Python und/oder Perl

- laurenceanthony.net/software

Ressourcen und einschlägige Software

- für Konvertierung der Ausgabedateien von Korpora ins KWIC-Format:

  github.com/hartmast/concordances

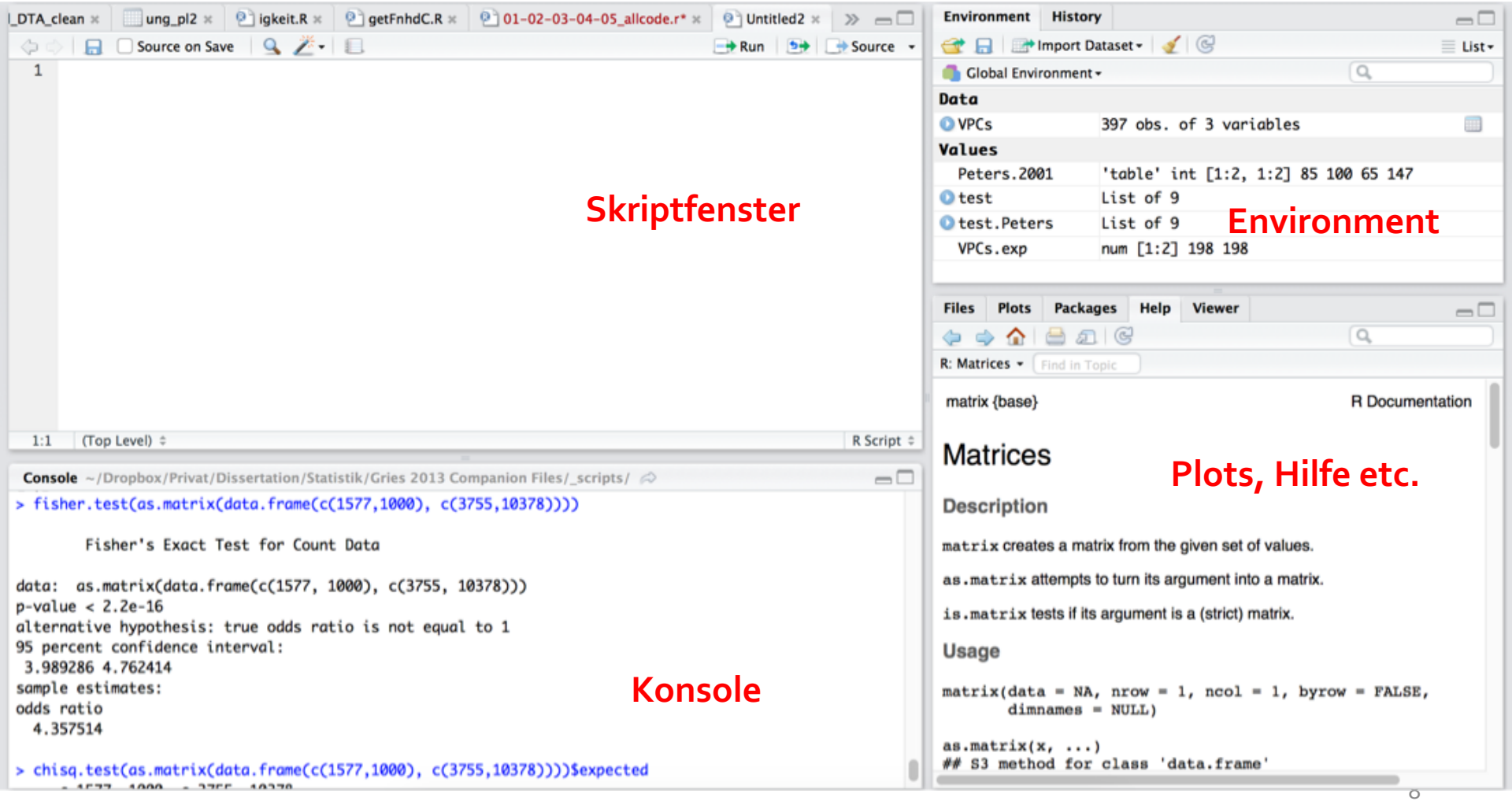
- Tutorials auf

  hartmast.github.io/sprachgeschichte

R

- Statistikprogramm und Programmiersprache
- kostenlos verfügbar unter www.r-project.org
- (geringfügig schnellere Variante: Revolution R – kann mehrere Prozessorkerne gleichzeitig benutzen)
- R ist ein Konsolenprogramm, eine gute Benutzeroberfläche bietet z.B. RStudio.

R Studio


Source on Save
Run
Source
Skriptfenster
Environment
History
Import Dataset
List
Global Environment
Data
VPCs
397 obs. of 3 variables
Values
Peters.2001
'table' int [1:2, 1:2] 85 100 65 147
test
List of 9
test.Peters
List of 9
VPCs.exp
num [1:2] 198 198
Environment
Files
Plots
Packages
Help
Viewer
R: Matrices
Find in Topic
matrix {base}
R Documentation
Matrices
Description
matrix creates a matrix from the given set of values.
as.matrix attempts to turn its argument into a matrix.
is.matrix tests if its argument is a (strict) matrix.
Usage
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,
dimnames = NULL)
as.matrix(x, ...)
S3 method for class 'data.frame'
Plots, Hilfe etc.
Console ~/Dropbox/Privat/Dissertation/Statistik/Gries 2013 Companion Files/_scripts/
> fisher.test(as.matrix(data.frame(c(1577,1000), c(3755,10378))))
Fisher's Exact Test for Count Data
data: as.matrix(data.frame(c(1577, 1000), c(3755, 10378)))
p-value < 2.2e-16
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
3.989286 4.762414
sample estimates:
odds ratio
4.357514
> chisq.test(as.matrix(data.frame(c(1577,1000), c(3755,10378))))$expected
Konsole

R

- R-Paket *concordances* lässt sich weitgehend **ohne** jegliche R-Kenntnisse benutzen.
- Wichtig ist nur, dass die in den Tutorials dargestellten Schritte richtig ausgeführt werden.
- Fehler sind natürlich dennoch möglich…

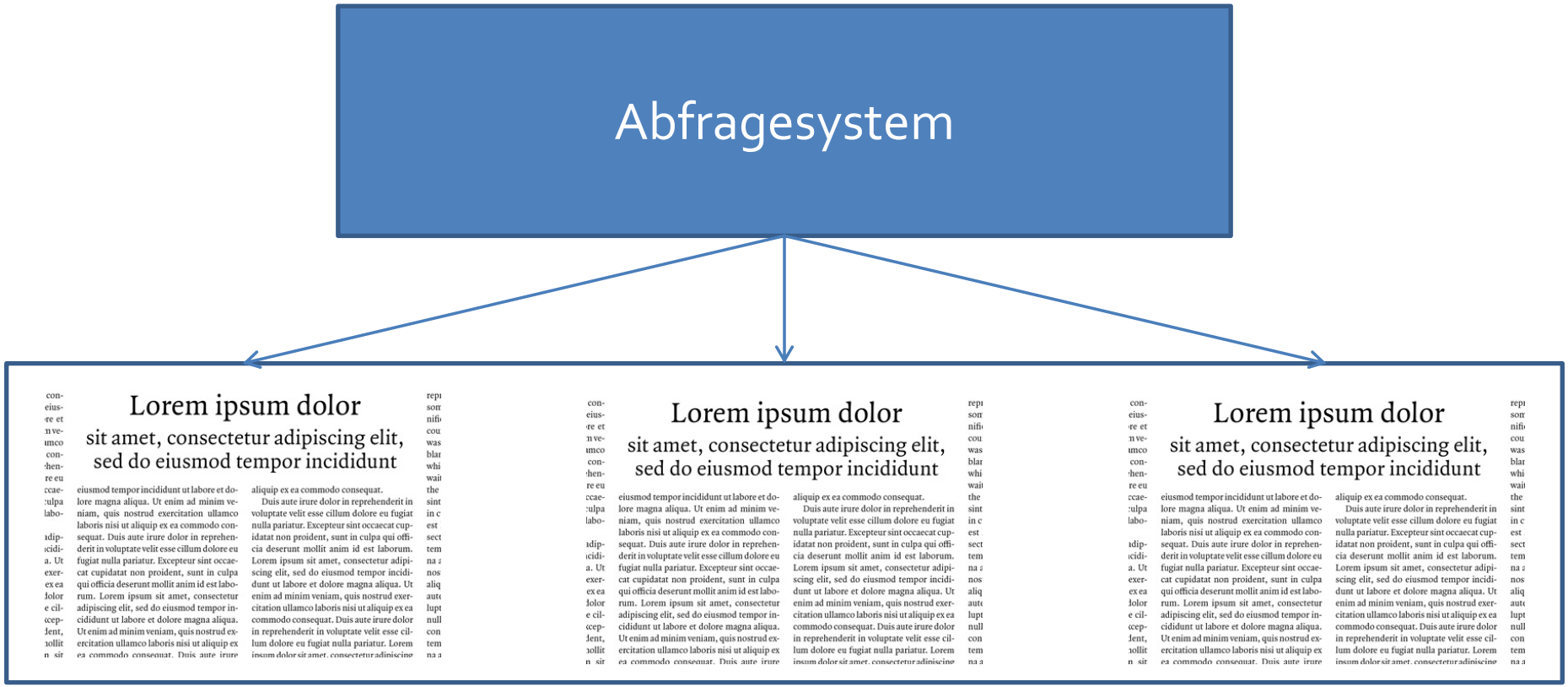
Ressourcen und einschlägige Software

- Texteditor: Notepad++ (für Windows), für Mac z.B. TextWrangler

- zur Arbeit mit Konkordanzen: Spreadsheet-Programme wie Excel oder LibreOffice Calc

Abfragesysteme und Abfragesyntax

Korpusabfragesysteme



Korpus

Korpusabfragesysteme

CWB
The IMS Open Corpus Workbench

ANNIS
A web browser-based search and visualization architecture for complex multilayer linguistic corpora with diverse types of annotation.

NoSketch Engine

Sketch Engine

Korpusabfragesysteme

- Einige Abfragesysteme sind desktop-basiert, andere web-basiert
- Von den meisten desktop-basierten Systemen sind web-basierte Versionen verfügbar
- In die web-basierten Versionen lassen sich i.d.R. jedoch keine eigenen Korpora einspeisen.

Abfragesyntax

- Korpusabfragesysteme haben oft eine eigene **Syntax**.
- Korpusabfrage-Syntax ist ein bisschen wie Fremdsprachensyntax…

➢ Man kann eine Fremdsprache benutzen, obwohl man die Syntax nur rudimentär beherrscht…

➢ …aber ihre Ausdrucksmöglichkeiten kann man nur mit guten Syntaxkenntnissen ausnutzen.

Abfragesyntax

Welche Optionen benötigen wir überhaupt, wenn wir ein Korpus durchsuchen?

Beispiel 1:

Wir suchen alle Belege für das Verb *legen* in einem **nicht getaggten** und **nicht lemmatisierten** Korpus.

Abfragesyntax

Welche Optionen benötigen wir überhaupt, wenn wir ein Korpus durchsuchen?

Beispiel 2:

Wir suchen alle Belege für die Verben *setzen*, *stellen*, *legen* in einem **lemmatisierten** Korpus.

Abfragesyntax

Welche Optionen benötigen wir überhaupt, wenn wir ein Korpus durchsuchen?

Beispiel 3:

Wir suchen Belege für die Wendung *je* X-*er desto* Y-*er* in einem getaggten Korpus.

Abfragesyntax

Welche Optionen benötigen wir überhaupt, wenn wir ein Korpus durchsuchen?

Beispiel 4:

Wir suchen Belege für *weil* + Substantiv und *weil* + Adjektiv in einem **getaggten** Korpus:

- Ich kann heute nicht ins Kino, weil Seminar.
- Ich will heute nicht ins Kino, weil müde.
- aber **nicht:** …weil Seminar ist.

Abfragesyntax

Welche Optionen benötigen wir überhaupt, wenn wir ein Korpus durchsuchen?

Beispiel 5:

Wir suchen Belege für V-*en gehen* in einem getaggten Korpus.

- Ich gehe heute schwimmen.
- Ich will heute schwimmen gehen.
- Ich gehe heute mit meinem Freund schwimmen.

Was brauchen wir also?

Logische Operatoren

- UND
- ODER
- NICHT

Wildcards

- leg*, setz*, stell*

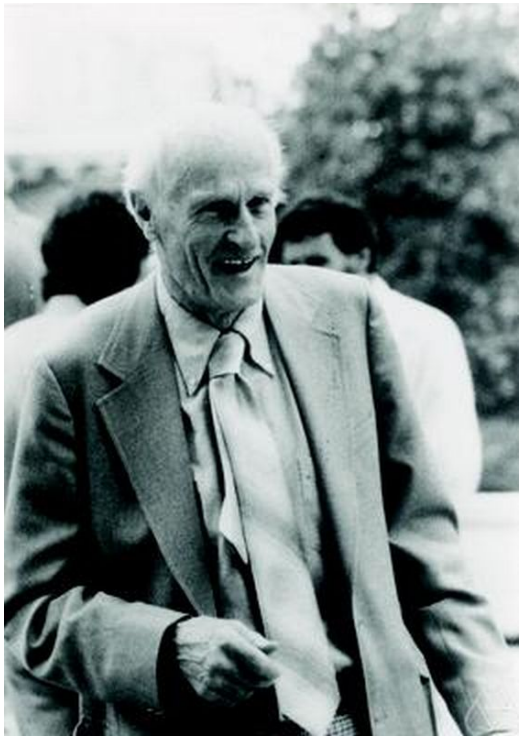
Wortabstandsoperatoren

- Ich gehe {0-5} schwimmen

Wie finde ich, was ich suche?

Reguläre Ausdrücke

- Zeichensequenz, die ein Suchmuster definiert
- Ursprung: Mathematik und Informatik



Stephen Kleene

Reguläre Ausdrücke

- sind Ihnen ggf. aus Internet-Suchmaschinen bekannt
- am bekanntesten wohl: * als Platzhalterzeichen
- Reguläre Ausdrücke können aber viel mehr!

Die wichtigsten regulären Ausdrücke…

Gruppierung durch Klammern

- ( ) Runde Klammern definieren eine **Erfassungsgruppe** (*capturing group*)
- [ ] Eckige Klammern definieren eine **Zeichenklasse** (*character class*), z.B. [abc] = irgendein Zeichen aus dem Inventar a,b,c, [asdf] irgendein Zeichen aus dem Inventar a,s,d,f.
- [^] quasi das negative "Gegenstück" zu [ ]: irgendein Zeichen, das **nicht** in dem Inventar an Zeichen enthalten ist, das in den eckigen Klammern definiert wird, z.B. [^abc]: irgendein Zeichen, das nicht a, b oder c ist.

- (Wichtig: In anderen Kontexten bedeutet ^ etwas anderes!)

Die wichtigsten regulären Ausdrücke

Wildcards und Wiederholungsoperatoren

- . irgendein Zeichen
- ? das Zeichen unmittelbar davor tritt 0- oder 1-mal auf.
- * das Zeichen unmittelbar davor tritt 0- oder x-mal (in unmittelbarer Folge) auf.
- + das Zeichen unmittelbar davor tritt 1- oder x-mal (in unmittelbarer Folge) auf.
- {n} das Zeichen unmittelbar davor tritt genau n-mal (in unmittelbarer Folge) auf.
- {x,} das Zeichen unmittelbar davor tritt mindestens x-mal (in unmittelbarer Folge) auf.
- {x,y} das Zeichen unmittelbar davor tritt mindestens x-, maximal y-mal (in unmittelbarer Folge) auf.

Die wichtigsten regulären Ausdrücke

Weitere Operatoren

- | oder-Operator
- \ Escape-String, z.B. um "echte" Fragezeichen zu finden
- ^ Anfangsposition
- $ Endposition

Zum Gebrauch regulärer Ausdrücke

- Kleinere und größere Unterschiede je nach Korpusabfragesprache
- z.B. in DWDS teilweise doppelter ODER-Operator erforderlich
- in einigen Korpora benutzt man statt oder-Operator das Wort oder (oder OR, oder ODER, ….)
- Ergo: Jede Korpusabfragesprache will gelernt sein (wie echte Sprachen auch…)

Das nächste Level:

Bracket expressions

- [:alnum:] alphanumerisch (a, b, 1, 2)
- [:alpha:] alphabetisch (a, b, c, nicht 1, 2)
- [:digit:] Ziffern (1, 2, 3, … nicht a, b, c)
- [:blank:] Leerzeichen, Tabstopps
- [:punct:] Interpunktion

Lookaround / Lookahead

(?=foo)	Lookahead	Der String, der der gesuchten Position unmittelbar folgt, ist *foo*
(?<=foo)	Lookbehind	Der String, der der gesuchten Position unmittelbar vorausgeht, ist *foo*
(?!foo)	Negative Lookahead	Der String, der der gesuchten Position unmittelbar folgt, ist nicht *foo*
(?<!foo)	Negative Lookbehind	Der String, der der gesuchten Position unmittelbar vorausgeht, ist nicht *foo*

Lookaround / Lookahead

(?=foo)	Lookahead	Der String, der der gesuchten Position unmittelbar folgt, ist *foo*
(?<=foo)	Lookbehind	Der String, der der gesuchten Position unmittelbar vorausgeht, ist *foo*
(?!foo)	Negative Lookahead	Der String, der der gesuchten Position unmittelbar folgt, ist nicht *foo*
(?<!foo)	Negative Lookbehind	Der String, der der gesuchten Position unmittelbar vorausgeht, ist nicht *foo*

Lookaround / Lookahead

(?=foo)	Lookahead	Der String, der der gesuchten Position unmittelbar folgt, ist *foo*
(?<=foo)	Lookbehind	Der String, der der gesuchten Position unmittelbar vorausgeht, ist *foo*
(?!foo)	Negative Lookahead	Der String, der der gesuchten Position unmittelbar folgt, ist nicht *foo*
(?<!foo)	Negative Lookbehind	Der String, der der gesuchten Position unmittelbar vorausgeht, ist nicht *foo*

Lookaround / Lookahead

(?=foo)	Lookahead	Der String, der der gesuchten Position unmittelbar folgt, ist *foo*
(?<=foo)	Lookbehind	Der String, der der gesuchten Position unmittelbar vorausgeht, ist *foo*
(?!foo)	Negative Lookahead	Der String, der der gesuchten Position unmittelbar folgt, ist nicht *foo*
(?<!foo)	Negative Lookbehind	Der String, der der gesuchten Position unmittelbar vorausgeht, ist nicht *foo*

Übung zu lookaround / lookahead

- Bitte schreiben Sie in den Texteditor:

```
Dies ist ein Test, der testen soll, wie
Lookahead und Lookbehind funktionieren.
```

- Benutzen Sie die Suchfunktion und Lookahed, um das Wort *schöner* vor *Test* einzufügen.
- Benutzen Sie Lookahead, um einen Tabstopp (`\t`) vor jedem Satzzeichen (`[[:punct:]]`) einzufügen.

Encoding Hell

Encöding h≈ll

- Use Unicode
- Use Unicode
- Use Unicode

(Quelle: http://pt.slideshare.net/MapRTechnologies/data-breaking-bad/19?smtNoRedir=1)

Encoding Hell

- Windows benutzt standardmäßig Windows-1252 (ähnlich ISO/IEC 8859-1 / Latin-1 / ASCII)
- Unix und Linux benutzen standardmäßig UTF-8 (Unicode-basiert)
- Die meisten deutschsprachigen historischen Korpora sind (wegen der Sonderzeichen) UTF-8-kodiert.
- Windows kann UTF-8, aber manchmal nur widerwillig…
- Daher bei Arbeit mit Windows oder Microsoft-Programmen (Excel!!) immer darauf achten, dass keine Sonderzeichen verlorengehen.

Encoding Hell

- Im Blick auf Encoding hat das kostenlose Calc einige Vorteil ggü. Excel
- (dafür jedoch teils schlechtere Performance und weniger Optionen)

Hinter den Kulissen eines Korpus

- tinyurl.com/korpling-siegen1

Was bringt der Blick hinter die Kulissen?

- Korpusdateien sehen oft furchteinflößend komplex aus…
- …aber sie sind hochstrukturiert!
- Das hat viele Vorteile, wenn man Suchabfragen machen will, die die jeweilige Suchabfragesyntax nicht kann.

Was bringt der Blick hinter die Kulissen?

- Beispiel: Das Referenzkorpus Mittelhochdeutsch verfügt über eine Satzgrenzenannotation…
- … aber es ist derzeit nicht möglich, ANNIS zu sagen: "Suche nur innerhalb eines bestimmten Satzes!"
- Wenn man das Korpus hingegen mit eigenen Skripten durchsucht, ist das kein Problem.

Beispiel: DWDS/DTA

```
<TextCorpus xmlns="http://www.dspin.de/data/textcorpus" lang="de">
  <tokens>
    <token ID="w1">Herrn</token>
    <token ID="w2">Hannß</token>
    <token ID="w3">Aßmanns</token>
    <token ID="w4">Freyherrn</token>
    <token ID="w5">von</token>
    <token ID="w6">Ab&#x017F;chatz</token>
    <token ID="w7">/</token>
    <token ID="w8">Weyl</token>
    <token ID="w9">.</token>
    <token ID="wa">gewe&#x017F;enen</token>
    <token ID="wb">Landes-Be&#x017F;tellten</token>
    <token ID="wc">im</token>
    <token ID="wd">Fu&#x0364;r&#x017F;tenthum</token>
    <token ID="we">Lignitz</token>
    <token ID="wf">/</token>
    <token ID="w10">und</token>
    <token ID="w11">bey</token>
    <token ID="w12">den</token>
    <token ID="w13">Publ</token>
    <token ID="w14">.</token>
    <token ID="w15">Conventibus</token>
    <token ID="w16">in</token>
    <token ID="w17">Breßlau</token>
    <token ID="w18">Hochan&#x017F;ehnl</token>
    <token ID="w19">.</token>
```

Beispiel: REM

```
<token id="t12" trans="in||handon(.)" type="token">
  <tok_dipl id="t12_d1" trans="inhandon" utf="inhandon"/>
  <tok_anno ascii="in" id="t12_m1" trans="in|" utf="in">
    <norm tag="in"/>
    <token_type tag="MS1"/>
    <lemma tag="in"/>
    <lemma_gen tag="in"/>
    <lemma_idmwb tag="81741000"/>
    <pos tag="APPR"/>
    <pos_gen tag="AP"/>
    <infl tag="c.D"/>
    <inflClass tag="--"/>
    <inflClass_gen tag="--"/>
  </tok_anno>
  <tok_anno ascii="handon" id="t12_m2" trans="|handon" utf="handon">
    <norm tag="handen"/>
    <token_type tag="MS2"/>
    <lemma tag="hant"/>
    <lemma_gen tag="hant"/>
    <lemma_idmwb tag="68277000"/>
    <pos tag="NA"/>
    <pos_gen tag="NA"/>
    <infl tag="Dat.Pl"/>
    <inflClass tag="st(u).Fem"/>
    <inflClass_gen tag="st(u).Fem"/>
    <punc tag="DE"/>
  </tok_anno>
</token>
```

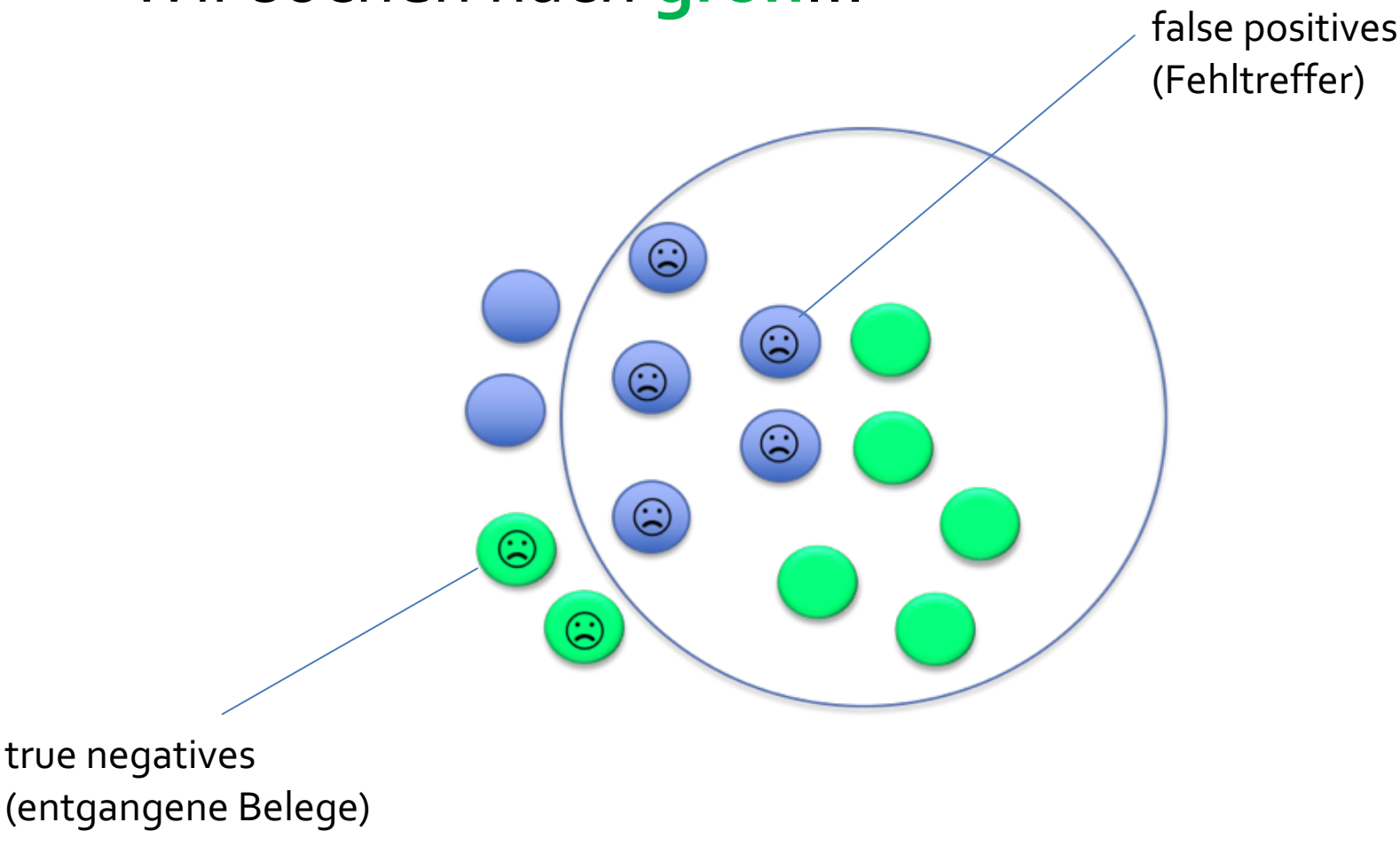
Precision und Recall

Finde ich, was ich suche?

- Bei einer Korpusrecherche wollen wir möglichst **alle** für uns relevanten Belege finden.
- Gleichzeitig möchten wir die Zahl der Fehltreffer möglichst **gering** halten.
- Man spricht hier auch von *Precision* und *Recall*

Precision und Recall

- Wir suchen nach **grün**…



Precision und Recall

- $Precision = \frac{Richtige\ Treffer}{Richtige\ Treffer + Fehltreffer}$

- $Recall = \frac{Richtige\ Treffer}{Richtige\ Treffer + entgangene\ Belege}$

- Ideal: 100% Precision und 100 % Recall

- Was ist wichtiger: Precision oder Recall?

Precision und Recall

Bitte überlegen Sie:

- Welche Faktoren können dazu führen, dass uns Treffer **entgehen**?

- Wie können wir die Zahl der Fehltreffer und der entgangenen Belege gering halten?