



Session 2: **Co-Occurrence Analysen**



„Magie“ verstehen: Klassische Schritte

1. Preprocessing

2. Analyse

3. Test auf
Qualitätskriterien



Session 1
(Einführung &
Preprocessing)



Session 2-4
(Co-Occurrence,
Diktionäre,
Topic Modeling)



Session 5
(Qualitätskriterien)

Bag-of-words Voraussetzung



Warum die bag-of-word Annahme problematisch ist

Vermutlich **✗** falsche Annahme:

- Can we "treat every word as having a distinct, unique meaning" (Grimmer et al., [2022](#), S. 79)?
- Can represent text "as if it were a bag of words, [...] with their position ignored, keeping only their frequency in the document" (Jurafsky & Martin, [2023](#), S. 60)?

Warum die bag-of-word Annahme problematisch ist

Annahme ✗ verletzt bzw. nicht hilfreich bei ...

- **Polysemie:** "Ich gehe zur Bank." vs. "Der Typ ist ne Bank!"
- **Verneinung:** "Nicht schlecht!"
- **Named Entities:** "Vereinigte Staaten", "Olaf Scholz"
- **Features mit ähnlichen Bedeutungen:** "I mag Gemüse." vs. "I mag Grünzeug."



Mehr als „bag-of-words“

Text-as-Data Repräsentationen, die diese Annahme weniger stark verletzen...

- Ngram-basierte Repräsentation (z. B. Collocations & Keywords-in-Context)
- Syntax-basierte Repräsentation (z. B. Part-of-Speech Tagging & Dependency Parsing)
- Vektor-basierte Repräsentation in semantischen, n-dimensionalen Räumen (z. B. Word Embeddings)



Mehr als „bag-of-words“

Text-as-Data Repräsentationen, die diese Annahme weniger stark verletzen...

- **Ngram-basierte Repräsentation** (z. B. Collocations & Keywords-in-Context)
- **Syntax-basierte Repräsentation (z. B. Part-of-Speech Tagging & Dependency Parsing)**
- Vektor-basierte Repräsentation in semantischen, n-dimensionalen Räumen (z. B. Word Embeddings)



Agenda

1. Ngrams und mehr
2. Part-of-Speech Tagging
3. Dependency Parsing
4. Anwendungsbeispiel in der Kowi
5. Outro

2. Ngrams und mehr

Ngrams

- Statt unigrams (d.h. einzelne Wörter) als Feature zu nutzen, können wir bigrams, trigrams, etc. nutzen, d.h. *Sequenzen von N aufeinander folgenden Features*
 - Unigram: "that"
 - Bigram: "that is"
 - Trigram: "that is great"



Kennt ihr Features, die
in Kombination eine
andere Bedeutung
haben?



Ngrams

- Statt unigrams (d.h. einzelne Wörter) als Feature zu nutzen, können wir bigrams, trigrams, etc. nutzen
 - Unigram: *"that"*
 - Bigram: *"that is"*
 - Trigram: *"that is great"*
- Im Rahmen von Preprocessing kann man einzelne Features auch zu einem Feature **zusammenfassen** (z.B. *"United"* und *"States"* zu *"United_States"*), damit diese vom Computer als einziges Feature interpretiert werden

Keywords-in-Context

- Keywords-in-Context beschreiben eher qualitativ den Kontext von Schlüsselwörtern
- Welche Wörter kommen um ein Fenster (*Window*) von n Features vor und nach einem Schlüsselwort vor?

```
Keyword-in-context with 5 matches.  
[text124, 7]          a | hero | that  
[text140, 38]        Rebel | hero | .  
[text336, 8] prestigious | hero | academy  
[text336, 19]         a | hero | ,  
[text756, 8]         a | hero | of
```

Kollokationen

- Kollokationen beschreiben eher quantitativ, welche Features häufig (und damit vermutlich nicht-zufällig) nacheinander auftreten, was z. B. auf eine gemeinsame semantische Bedeutung hindeutet (z.B. *United* und *States*).

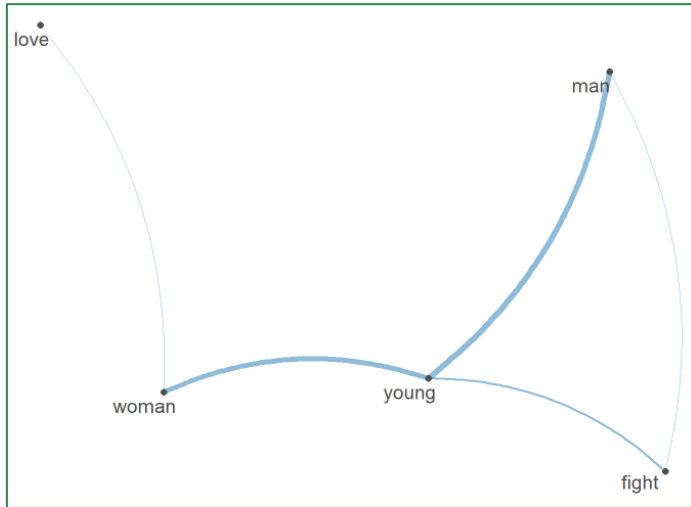
collocation	count	count_nested	length	lambda	z
los angel	22	0	2	11.992530	7.856166
new york	39	0	2	9.635186	6.744491
serial killer	10	0	2	8.665918	11.849150
person profession	13	0	2	7.817347	12.191558
antholog seri	10	0	2	7.632806	8.612064
high school	22	0	2	7.041380	16.490961
best friend	25	0	2	7.006030	15.088077
york citi	19	0	2	5.810906	16.071342
seri follow	10	0	2	4.323497	11.362725



Was sind typische
Beispiele für
Kollokationen?

Semantische Netzwerke

- Semantische Netzwerke visualisieren Ko-Okkurrenzen, d.h., welche Features häufig gemeinsam vorkommen



Beispiel: Wie werden Männer bzw. Frauen in Serienbeschreibungen dargestellt?

Für welche Fragestellungen eignen sich solche Verfahren?

- Textbereinigung: Entfernung von Duplikaten via ngram-shingling (Nicholls, [2019](#))

International Journal of Communication 13(2019), 4173–4197

1932–8036/20190005

Detecting Textual Reuse in News Stories, At Scale

TOM NICHOLLS¹

University of Oxford, UK

Motivated by the debate around “churnalism” and online media, this article develops, evaluates, and validates a computational method for detecting shared text between different news articles, at scale, using n -gram shingling. It differentiates between newswire copy, public relations material, source-to-source copying, and common-source and incidental overlaps. I evaluate the method, quantitatively and qualitatively, and show that it can effectively handle newswire content, copying, and other forms of reuse. Substantively, I find lower levels of news agency and press release copy reuse than is suggested by previous studies, and conclude that the news agency finding is robust, but the lack of press release copy found might reflect limitations of the method and the changing practices of journalists.

Keywords: computational methods, news production, churnalism, news agency, automated content analysis, online news

Für welche Fragestellungen eignen sich solche Verfahren?

- Textanalyse, u. a.: Analyse von Stereotypen (Arendt & Karadas, [2017](#)), Labeling, ggf. Frames (?) ([Ruigrok & van Atteveldt, 2007](#))

COMMUNICATION METHODS AND MEASURES
2017, VOL. 11, NO. 2, 105–120
<http://dx.doi.org/10.1080/19312458.2016.1276894>

 **Routledge**
Taylor & Francis Group

Content Analysis of Mediated Associations: An Automated Text-Analytic Approach

Florian Arendt and Narin Karadas

University of Munich (LMU), Munich, Germany

ABSTRACT

Due to the fact that mediated associations are a central aspect of many mass communication theories, their measurement is of central interest for communication research. Mediated associations are defined as the repeated pairing of an object (e.g., social group, political party) with specific attributes (e.g., crime, economy). In this article, we introduce a recently developed, automated text-analytic technique. We present an application of this method in the media stereotyping domain via the content analysis of German news coverage of Islam. As predicted, the analysis revealed substantial mediated associations between Islam-related concepts and violence (e.g., "Koran + violence"), terror (e.g., "Islam + terror"), dehumanization (e.g., "Muslims + animal-related terms"), and general negativity (valence). We discuss the promises and pitfalls of this method, make software suggestions, and provide application-related information for speedy dissemination in communication research.



Wie kann ich diese Analysen in R anwenden?

- Kleine Auswahl möglicher Analysen

- Ngrams
- Keywords-in-Context
- Collocations
- Semantische Netzwerke

- Kleine Auswahl möglicher R-Pakete

- „base R“ (z.B. für Identifikation von Features via `grepl()` etc,)
- „quanteda“ bzw. „quanteda.textstats“ bzw. „quanteda.textplots“ (z.B. für Identifikation von KWIC, Collocations oder semantische Netzwerkanalysen)
- „textreuse“ (z.B. zur Identifikation von Text-Duplikaten via ngrams)

```
function(scope, element, attr, ngSwitchController) {  
  var watchExpr = attr.ngSwitch || attr.on,  
      selectedTranscludes = [],  
      selectedElements = [],  
      previousElements = [],  
      selectedScopes = [];  
  
  scope.$watch(watchExpr, function ngSwitchWatchAction(value) {  
    var i, ii;  
    for (i = 0, ii = previousElements.length; i < ii; ++i) {  
      previousElements[i].remove();  
    }  
    previousElements.length = 0;  
  
    for (i = 0, ii = selectedScopes.length; i < ii; ++i) {  
      var selected = selectedElements[i];  
      selectedScope[i].destroy();  
    }  
  });  
  
  selectedElements.length = 0;  
  selectedScopes.length = 0;  
  
  if ((selectedTransclude = ...))
```

Zeit für R!

```
selectedElements.length = 0;  
selectedScopes.length = 0;  
  
if ((selectedTransclude = ...))
```

Pakete installieren & aktivieren

```
#install.packages("tidyverse")  
#install.packages("quanteda")  
#install.packages("quanteda.textplots")  
#install.packages("RCurl")  
#install.packages("quanteda.textstats")  
#install.packages("udpipe")
```

```
library("tidyverse")  
library("quanteda")  
library("quanteda.textplots")  
library("RCurl")  
library("quanteda.textstats")  
library("udpipe")
```

```
install.packages("rsyntax")  
library("rsyntax")
```

Daten einlesen & Preprocessing

```
#Daten laden
url <- getURL("https://raw.githubusercontent.com/valeriehase/textasdata-ms/main/data")
data <- read.csv2(text = url)

#Preprocessing
tokens <- tokens(data$Description,
  what = "word", #Tokenisierung, hier zu Wörtern als Analyseeinheit
  remove_punct = TRUE, #Entfernung von Satzzeichen
  remove_numbers = TRUE) %>% #Entfernung von Zahlen

#Kleinschreibung
tokens_tolower() %>%

#Entfernung von Stoppwörtern
tokens_remove(stopwords("english")) %>%

#Stemming
tokens_wordstem()
```

Daten einlesen & Preprocessing

```
#Text-as-Data Repräsentation als Document-Feature-Matrix
dfm <- tokens %>%
  dfm() %>%

#Relative pruning
dfm_trim( min_docfreq = 0.005,
          max_docfreq = 0.99,
          docfreq_type = "prop",
          verbose = TRUE)
```

Ngrams

```
tokens %>%
```

```
#Umwandlung in bigrams
```

```
tokens_ngrams(n = 2) %>%
```

```
#Ausgabe für erstes Dokument
```

```
head(1)
```

Tokens consisting of 1 document.

text1 :

[1]	"nine_nobl"	"nobl_famili"	"famili_fight"
[4]	"fight_control"	"control_land"	"land_westero"
[7]	"westero_ancient"	"ancient_enemi"	"enemi_return"
[10]	"return_dormant"	"dormant_millennia"	

Häufige Ngrams

```
tokens %>%
```

```
#Umwandlung in bigrams
```

```
tokens_ngrams(n = 2) %>%
```

```
#Umwandlung in dfm für topfeatures-Befehl
```

```
dfm() %>%
```

```
#Ausgabe der häufigsten Features
```

```
topfeatures(10) %>%
```

```
#Umwandlung in einen "schöneren" Dataframe mit der Spalte "Häufigkeit"
```

```
as.data.frame() %>%
```

```
rename("Häufigkeit" = '.')
```

	Häufigkeit
new_york	39
best_friend	25
high_school	22
los_angel	22
york_citi	19
person_profession	13
serial_killer	10
antholog_seri	10
seri_follow	10
young_boy	9


Ngrams zusammenfassen

```
# Definition häufiger Ngrams auf Basis der vorherigen Ausgabe
ngrams <- c("los angel", "new york citi", "serial killer", "high school", "best friend")

# Text-as-Data Repräsentation als Document-Feature-Matrix
dfm <- tokens %>%

# Zusätzlicher Schritt, um Ngrams als einzelnes Feature einzulesen
tokens_compound(pattern = phrase(ngrams)) %>%

# reguläre DFM, inkl. Relative Pruning
dfm() %>%
dfm_trim( min_docfreq = 0.005,
          max_docfreq = 0.99,
          docfreq_type = "prop",
          verbose = TRUE)
```



Ngrams zusammenfassen

```
# Beispiel: Wie wird das Feature "Los Angeles" eingelesen?
dfm %>%

# Umwandlung zu Data-Frame
convert(to = "data.frame") %>%

# Reduktion auf Doc ID und Features, die mit "los" beginnen
select(doc_id, starts_with("los")) %>%

# Ausgabe ausgewählter Serien (Zeile 125 bis 130)
slice(125:130)
```

	doc_id	lost	los_angel
1	text125	0	0
2	text126	0	1
3	text127	0	0
4	text128	0	0
5	text129	0	1
6	text130	0	0

Keywords-in-Context

```
tokens %>%
```

```
# Keywords-in-Context mit Window von 1 Wort vor und nach Schlüsselwort
```

```
kwic(pattern = "hero",  
      window = 1) %>%
```

```
# Ausgabe der ersten Zeilen
```

```
head()
```

Keyword-in-context with 6 matches.

```
[text75, 4]  child | hero | now  
[text124, 3] saitama | hero | just  
[text140, 18] rebel | hero |  
[text230, 24] team | hero |  
[text241, 9] team | hero | villain  
[text292, 15] v | hero | put
```

Collocations

```
tokens %>%
```

```
# Identifikation von Collocations, die mind. 10 Mal vorkommen
```

```
textstat_collocations(min_count = 10) %>%
```

```
# Sortierung nach lambda: Je grösser,
```

```
# desto wahrscheinlicher handelt es sich um nicht-zufällige Collocations  
arrange(-lambda) %>%
```

```
# Ausgabe der häufigsten 10 Collocation
```

```
head(10)
```

	collocation	count	count_nested	length	lambda	z
8	los angel	22	0	2	11.992530	7.856166
9	new york	39	0	2	9.635186	6.744491
5	serial killer	10	0	2	8.665918	11.849150
4	person profession	13	0	2	7.817347	12.191558

Semantische Netzwerke

```
tokens %>%
```

```
# Umwandlung in eine Feature-Co-Occurrence-Matrix
```

```
fcm(context = "document") %>%
```

```
# Ausgabe der ersten Zeilen
```

```
head()
```

Feature co-occurrence matrix of: 6 by 4,246 features.

features

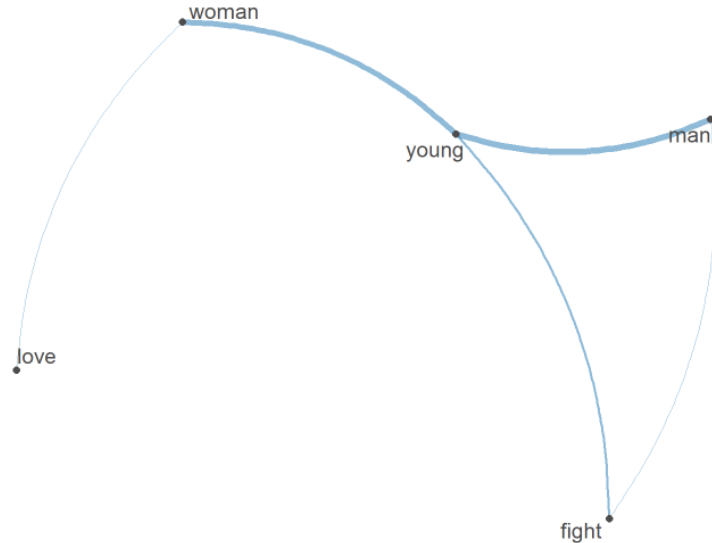
features	nine	nobl	famili	fight	control	land	westero	ancient	enemi	return
nine	0	1	1	1	1	1	1	1	1	1
nobl	0	0	1	1	1	1	1	1	1	1
famili	0	0	8	6	2	3	1	1	4	6
fight	0	0	0	1	2	2	1	2	2	2
control	0	0	0	0	0	1	1	1	1	1
land	0	0	0	0	0	0	1	1	1	1

[reached max_nfeat ... 4,236 more features]

Semantische Netzwerke

```
fcm <- tokens %>%  
  
# Erstellung einer FCM mit einem Window von 8  
fcm(window = 8) %>%  
  
# Reduktion auf ausgewählter Features  
fcm_select(pattern = c("fight", "man",  
                      "love", "young", "woman"),  
            selection = "keep")
```

```
# Plot des semantischen Netzwerks  
textplot_network(fcm)
```





Mehr als „bag-of-words“

Text-as-Data Repräsentationen, die diese Annahme weniger stark verletzen...

- Ngram-basierte Repräsentation (z. B. Collocations & Keywords-in-Context)
- **Syntax-basierte Repräsentation (z. B. Part-of-Speech Tagging & Dependency Parsing)**
- Vektor-basierte Repräsentation in semantischen, n-dimensionalen Räumen (z. B. Word Embeddings)



Pause

3. Part-of-Speech Tagging

Part-of-speech Tagging

- Analyse der Syntax, um Features Wortarten zuzuordnen
- Part-of-speech tagging as the "process of assigning a part-of-speech to each word in a text" (Jurafsky & Martin, [2023](#), S. 163)

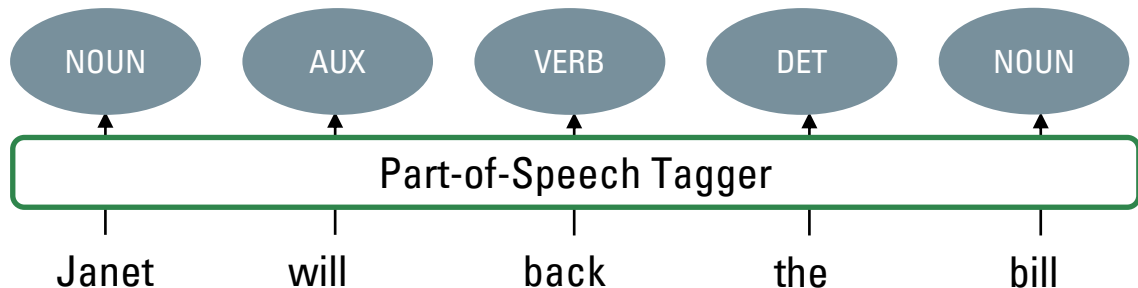


Abbildung s. Jurafsky & Martin ([2023](#), S. 164).
Für Infos zu Tags, siehe de Marneffe et al. ([2021](#)).



Part-of-speech Tagging

Damit können wir ...

- z. B. analysieren, ob es sich bei einem Feature um ein Adjektiv handelt, das sich auf ein bestimmtes Substantiv bezieht
- z. B. zwischen gleichen Features mit unterschiedlichen Bedeutungen unterscheiden („*Sound solution*“ vs. „*What is that sound*“?)

Für welche Fragestellungen eignen sich solche Verfahren?

- Textanalyse, u. a.: Identifikation von Text-Reuse: Wie sehr nutzen Journalist:innen

Material von Nachrichtenagenturen? (Welbers et al., [2018](#))

A GATEKEEPER AMONG GATEKEEPERS
News agency influence in print and online
newspapers in the Netherlands

Kasper Welbers, Wouter van Atteveldt, Jan Kleinnijhuis, and
Nel Ruigrok

This paper investigates the influence of news agency Algemeen Nederlands Persbureau (ANP) on the coverage and diversity of political news in Dutch national newspapers. Using computational text analysis, we analyzed the influence on print newspapers across three years (1996, 2008, and 2013) and compared influence on print and online newspapers in 2013. Results indicate that the influence of ANP on print newspapers only increased slightly. Online newspapers, however, depend heavily on ANP and are highly similar as a result of such dependence. We draw conclusions relating to the gatekeeping role of news agencies in the digital age in general, and in the context of the Netherlands in particular. Additionally, we demonstrate that techniques from the field of information retrieval can be used to perform these analyses on a large scale. Our scripts and instructions are published open-source to stimulate the use of these techniques in communication studies.



Wie kann ich diese Analysen in R anwenden?

- Kleine Auswahl möglicher R-Pakete
 - „udpipe“
 - „spacyr“
 - „openNLP“

```
function(scope, element, attr, ngSwitchController) {  
  var watchExpr = attr.ngSwitch || attr.on,  
      selectedTranscludes = [],  
      selectedElements = [],  
      previousElements = [],  
      selectedScopes = [];  
  
  scope.$watch(watchExpr, function ngSwitchWatchAction(value) {  
    var i, ii;  
    for (i = 0, ii = previousElements.length; i < ii; ++i) {  
      previousElements[i].remove();  
    }  
    previousElements.length = 0;  
  
    for (i = 0, ii = selectedScopes.length; i < ii; ++i) {  
      var selected = selectedElements[i];  
      selectedScope[i].destroy();  
    }  
  });  
  
  selectedElements.length = 0;  
  selectedScopes.length = 0;  
  
  if ((selectedTransclude =
```

Zeit für R!

```
selectedElements.length = 0;  
selectedScopes.length = 0;  
  
if ((selectedTransclude =
```

Part-of-Speech Tagging

```
data_pos_tagged <- data$Description %>%  
  
  # Format für das udpipe Paket anpassen  
  as_tibble() %>%  
  mutate(doc_id = paste0("text", 1:n())) %>%  
  rename(text = value) %>%  
  
  # Part-of-speech tagging  
  udpipe("english") %>%  
  
  # Wir reduzieren die Ausgabe auf relevante Variablen (z.B. Text-ID, Tag)  
  select(doc_id, sentence_id, token_id, token, lemma, upos, head_token_id)  
  
  # Wir schauen uns die Ausgabe an  
  head(data_pos_tagged)
```

Part-of-Speech Tagging

	doc_id	sentence_id	token_id	token	lemma	upos	head_token_id
1	text1	1	1	Nine	nine	PROPN	3
2	text1	1	2	noble	noble	ADJ	3
3	text1	1	3	families	family	NOUN	4
4	text1	1	4	fight	fight	VERB	0
5	text1	1	5	for	for	ADP	6
6	text1	1	6	control	control	NOUN	4

Beispiel: Mit welchen Adjektiven werden Familien beschrieben?

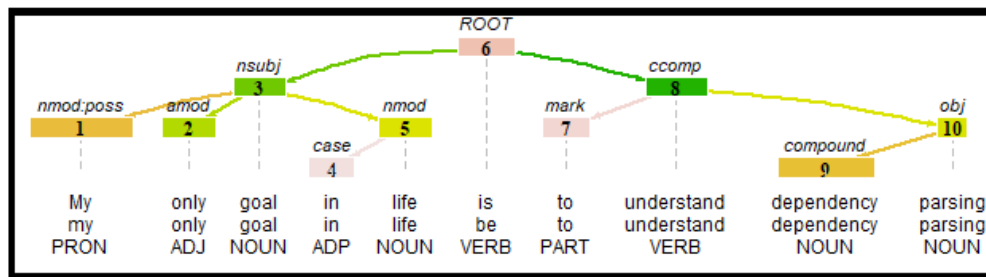
```
data_pos_tagged %>%  
  
# Wir filtern den Datensatz nach dem Substantiv "Family"  
filter(upos == "NOUN" & lemma == "family") %>%  
  
# Für alle gefundenen Fälle suchen wir die zugehörigen Sätze im "vollen" Datensatz  
# Das Matching geschieht via doc_id (ID des Dokuments) und sentence_id (ID des Satzes)  
inner_join(data_pos_tagged, by = c("doc_id", "sentence_id"), relationship = "many-to-many") %>%  
  
# Wir behalten mit filter nur Adjektive, die sich auf Familie beziehen  
# Nämlich solche, die bei "head_token" die "token_id" des Features "Family" haben  
filter(upos.y == "ADJ" & head_token_id.y == token_id.x) %>%  
  
# Wir benennen manche Variablen um, damit das Ganze besser verständlich ist  
rename(token_id = token_id.y,  
       token = token.y) %>%  
  
# Wir wählen nur relevante Variablen aus  
select(doc_id, sentence_id, token_id, token) %>%  
  
# erste Zeilen ausgeben  
head()
```

	doc_id	sentence_id	token_id	token
1	text1	1	2	noble
2	text48	1	9	dysfunctional
3	text61	2	6	spoiled
4	text61	2	8	dysfunctional
5	text69	1	7	loyal
6	text98	1	8	British

4. Dependency Parsing

Dependency Parsing

- Describing “the syntactic structure of a sentence [...] in terms of directed binary grammatical relations between the words” (Jurafsky & Martin, [2023](#), S. 381)
- Wir analysieren die semantische Bedeutung von Features über ihre syntaktische Abhängigkeitsbeziehung zu einem “root” (Dependenzstruktur)



Beispiel für Dependency Parsing



Dependency Parsing

Damit können wir ...

- z. B. analysieren, ob es sich bei einem Feature um ein Adjektiv handelt, das sich auf ein bestimmtes Substantiv bezieht
- z. B. zwischen gleichen Features mit unterschiedlichen Bedeutungen unterscheiden („*Sound solution*“ vs. „*What is that sound*“?)

Für welche Fragestellungen eignen sich solche Verfahren?

- Textanalyse, u. a.: Wer sagt was über wen? (Fogel-Dror et al., [2018](#))





Wie kann ich diese Analysen in R anwenden?

- Kleine Auswahl möglicher R-Pakete
 - „udpipe“
 - „spacyr“
 - „openNLP“
 - „rsyntax“ (z.B. für Visualisierung)

```
function(scope, element, attr, ngSwitchController) {  
  var watchExpr = attr.ngSwitch || attr.on,  
      selectedTranscludes = [],  
      selectedElements = [],  
      previousElements = [],  
      selectedScopes = [];  
  
  scope.$watch(watchExpr, function ngSwitchWatchAction(value) {  
    var i, ii;  
    for (i = 0, ii = previousElements.length; i < ii; ++i) {  
      previousElements[i].remove();  
    }  
    previousElements.length = 0;  
  
    for (i = 0, ii = selectedScopes.length; i < ii; ++i) {  
      var selected = selectedElements[i];  
      selectedScope[i].destroy();  
    }  
  });  
  
  selectedElements.length = 0;  
  selectedScopes.length = 0;  
  
  if ((selectedTransclude = ...))
```

Zeit für R!

```
selectedElements.length = 0;  
selectedScopes.length = 0;  
  
if ((selectedTransclude = ...))
```

Dependency Parsing

```
data$Description %>%
```

```
# Format für das udpipe Paket anpassen
```

```
as_tibble() %>%
```

```
mutate(doc_id = paste0("text", 1:n())) %>%
```

```
rename(text = value) %>%
```

```
# Der Einfachheit halber machen wir diese Analyse nur für einen Text
```

```
slice(1) %>%
```

```
# dependency parsing
```

```
udpipe("english") %>%
```

```
# relevanten Variablen auswählen
```

```
select(doc_id, sentence_id, token_id, token, head_token_id, dep_rel) %>%
```

```
# erste Zeilen ausgeben
```

```
head(5)
```

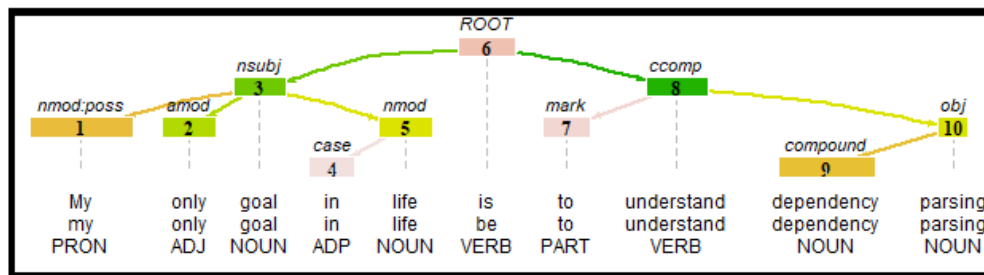
	doc_id	sentence_id	token_id	token	head_token_id	dep_rel
1	text1	1	1	Nine	3	compound
2	text1	1	2	noble	3	amod
3	text1	1	3	families	4	nsubj
4	text1	1	4	fight	0	root
5	text1	1	5	for	6	case

Dependency Parsing

```
# Beispielsatz in udpipe
udpipe("My only goal in life is to understand dependency parsing", "english") %>%

# Umwandlung in Format für rsyntax-Paket
as_tokenindex() %>%

# Visualisierung
plot_tree(., token, lemma, upos)
```



Beispiel für Dependency Parsing

5. Anwendungsaufgabe



Jetzt seid ihr dran!



AUFGABE 1

Die folgende Übung fasst alles zusammen, was wir bisher gelernt haben: Preprocessing und Co-Occurrence Analysen.

Bitte arbeitet für die Übung mit dem Horoskop-Datensatz (Download der CSV-Datei entweder via der Webseite oder Einlesen via [dieser](#) URL)

Jetzt seid ihr dran!



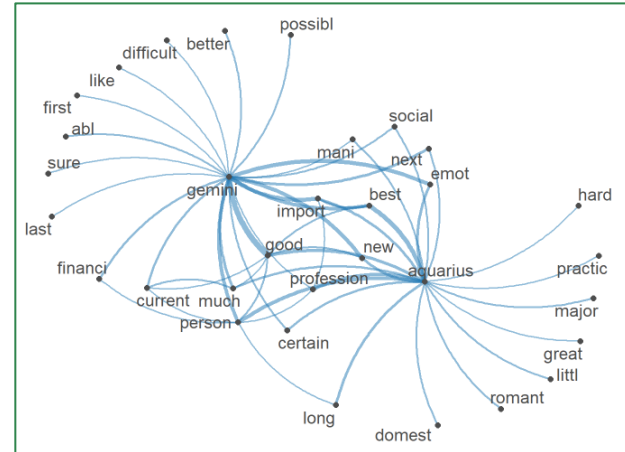
- Aufgabe 1.1 (Basis): Lest den Datensatz ein und verschafft euch einen Überblick über die Daten. Welche Variablen sind dort vorhanden?
- Aufgabe 1.2 (Basis): Bereitet den Datensatz durch Preprocessing und das Umwandeln in eine DFM für die Analyse vor. Hinterfragt kritisch, welche Bereinigung- und Normalisierungsschritte ihr tatsächlich braucht.
- Aufgabe 1.3 (Basis): Schaut euch als erste Analyse an, welcher Ausdruck häufiger vorkommt: “secret fear” oder “in love”?

Jetzt seid ihr dran!



- Aufgabe 1.4 (Fortgeschritten): Jetzt wollen wir wissen, bei welchem Sternzeichen es am mysteriösesten wird: Bei welchem Sternzeichen fällt am häufigsten das Stichwort “secret”?
- Aufgabe 1.5 (Expert:in): Visualisiert auf Basis eines semantischen Netzwerk, mit welchen Adjektiven die Sternzeichen “Aquarius” (Wassermann) vs. “Gemini” (Zwilling) häufig in den Horoskopen assoziiert werden.

Das Netzwerk könnte etwa so aussehen:



Jetzt seid ihr dran!



AUFGABE 1.1 (BASIS)

Lest den Datensatz ein und verschafft euch einen Überblick über die Daten. Welche Variablen sind dort vorhanden?

```
#Daten einlesen
url <- getURL("https://raw.githubusercontent.com/valeriehase/textasdata-ms/main/data/d
horoscope_df <- read.csv2(text = url)

#Überblick über Datensatz verschaffen
str(horoscope_df)
```

```
'data.frame':  1000 obs. of  3 variables:
 $ Date      : chr  "06-06-2021" "06-06-2021" "06-06-2021" "06-06-2021" ...
 $ Signs     : chr  "ARIES" "TAURUS" "GEMINI" "CANCER" ...
 $ Horoscope : chr  " Your week falls neatly into distinct phases. The completion of rou
```

Jetzt seid ihr dran!



AUFGABE 1.2 (BASIS)

Bereitet den Datensatz durch Preprocessing und das Umwandeln in eine DFM für die Analyse vor. Hinterfragt kritisch, welche Bereinigungs- und Normalisierungsschritte ihr tatsächlich braucht.

```
# Normalisierung
horoscope_tokens <- horoscope_df$Horoscope %>%
  tokens(what = "word",
    remove_punct = TRUE,
    remove_numbers = TRUE,
    remove_symbols = TRUE) %>%
  tokens_tolower() %>%
  tokens_remove(stopwords("english")) %>%
  tokens_wordstem()
```

Jetzt seid ihr dran!



```
# Umwandlung in eine DFM
horoscope_dfm <- horoscope_tokens %>%
  dfm() %>%
  dfm_trim( min_docfreq = 0.005,
            max_docfreq = 0.99,
            docfreq_type = "prop",
            verbose = TRUE)

head(horoscope_dfm)
```

Document-feature matrix of: 6 documents, 917 features (97.49% sparse) and 0 docvars.

	features									
docs	week	fall	phase	complet	routin	task	number	one	prioriti	deal
text1	1	1	1	1	1	1	1	1	1	1
text2	0	0	0	0	0	0	0	0	0	0
text3	0	0	0	0	0	0	0	0	0	1
text4	0	0	0	0	0	0	0	1	0	1
text5	0	0	0	0	0	0	0	0	0	0
text6	1	0	0	0	0	0	0	1	0	1

[reached max_nfeat ... 907 more features]

Jetzt seid ihr dran!



AUFGABE 1.3 (FORTGESCHRITTEN)

Schaut euch als erste Analyse an, welcher Ausdruck häufiger vorkommt: "secret fear" oder "in love"?

```
# Lösung mit regulären Ausdrücken
horoscope_df %>%

# Auszählen
summarize("Häufigkeit: in love" = sum(grepl("in love", tolower(Horoscope))),
          "Häufigkeit: secret fear" = sum(grepl("secret fear", tolower(Horoscope)))
```

```
Häufigkeit: in love Häufigkeit: secret fear
1                12                3
```


Jetzt seid ihr dran!



AUFGABE 1.4 (FORTGESCHRITTEN) [🔗](#)

Jetzt wollen wir wissen, bei welchem Sternzeichen es am mysteriösesten wird: Bei welchem Sternzeichen fällt am häufigsten das Stichwort "secret"?

Lösung 1:

```
#Version 1: Mit regulären Ausdrücken
horoscope_df %>%

#Gruppierung nach Sternzeichen
group_by(Signs) %>%

#Auszählen
summarize(Geheimnis = sum(grepl("secret", Horoscope))) %>%

#Absteigend sortieren
arrange(desc(Geheimnis))
```

```
# A tibble: 11 × 2
  Signs      Geheimnis
  <chr>      <int>
1 CANCER      4
2 VIRGO       4
3 LIBRA       3
4 SCORPIO     3
5 AQUARIUS    2
6 GEMINI      2
7 LEO         2
8 SAGITTARIUS 2
9 TAURUS      2
10 ARIES       1
11 CAPRICORN   1
```

Jetzt seid ihr dran!



Lösung 2:

```
#Version 2: Mit Quanteda
```

```
#Wir analysieren die Häufigkeit des Features gruppiert nach Sternzeichen  
textstat_frequency(horoscope_dfm, groups = horoscope_df$Signs) %>%
```

```
#Wir filtern nur für das Feature "secret"  
filter(feature == "secret") %>%
```

```
#Absteigend sortieren  
arrange(desc(docfreq))
```

	feature	frequency	rank	docfreq	group
1302	secret	4	85	4	CANCER
5135	secret	4	117	4	SCORPIO
6456	secret	4	129	4	VIRGO
227	secret	3	162	3	AQUARIUS
3923	secret	3	165	3	LIBRA
5871	secret	3	176	3	TAURUS
913	secret	2	144	2	ARIES
2794	secret	2	304	2	GEMINI
3425	secret	2	254	2	LEO
4732	secret	2	245	2	SAGITTARIUS
2333	secret	1	387	1	CAPRICORN

Jetzt seid ihr dran!



AUFGABE 1.5 (EXPERT:IN) [🔗](#)

Visualisiert auf Basis eines semantischen Netzwerk, mit welchen Adjektiven die Sternzeichen "Aquarius" (Wassermann) vs. "Gemini" (Zwilling) häufig in den Horoskopen assoziiert werden.

```
horoscope_df_filtered <- horoscope_df %>%  
  
#Reduktion des Datensatzes auf ausgewählte Sternzeichen  
filter(Signs %in% c("AQUARIUS", "GEMINI"))
```

Jetzt seid ihr dran!



```
horoscope_df_adj <- horoscope_df_filtered %>%

#Reduktion auf Textvariable "Horoscope"
select(Horoscope) %>%

#Format für das udpipe Paket anpassen
as_tibble() %>%
mutate(doc_id = paste0("text", 1:n())) %>%
rename(text = Horoscope) %>%

#Part-of-speech tagging, um Adjektive zu identifizieren
udpipe("english") %>%

#Wir behalten mit filter nur Adjektive
as_tibble() %>%
filter(upos == "ADJ") %>%

#Wir reduzieren die Ausgabe auf relevante Variablen (Text-ID, Feature
select(doc_id, token) %>%

#Wir erstellen einen auf Adjektive reduzierten Text je Horoskop
group_by(doc_id) %>%
summarize(Horoscope_adj = paste0(token, collapse = " ")) %>%
distinct(doc_id, .keep_all = T)
```

Jetzt seid ihr dran!



```
#Wir bringen den auf Adjektive reduzierten Datensatz &
#die Informationen zu den einzelnen Horoskopen zusammen
horoscope_df_filtered <- horoscope_df_filtered %>%

#Wir erstellen einen ID, um die Informationen zu matchen
mutate(doc_id = paste0("text", 1:n())) %>%

#Wir fügen die beiden Datensätze zusammen
left_join(horoscope_df_adj) %>%

#Wir fügen das Sternzeichen zum jeweiligen Text hinzu, quasi als "Titel"
mutate(Horoscope = paste0(Horoscope_adj, " ", Signs))
```

Jetzt seid ihr dran!



```
#Semantisches Netzwerk

#Preprocessing des Textes
tokens(horoscope_df_filtered$Horoscope,
      what = "word", #Tokenisierung, hier zu Wörtern als Analyseeinheit
      remove_punct = TRUE, #Entfernung von Satzzeichen
      remove_numbers = TRUE) %>% #Entfernung von Zahlen

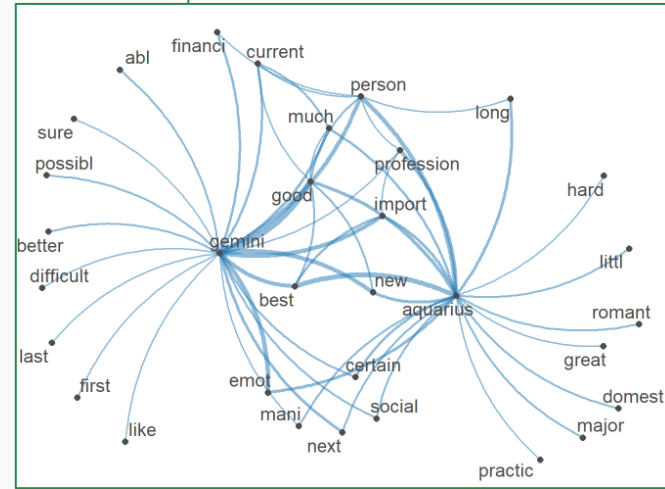
#Kleinschreibung
tokens_tolower() %>%

#Entfernung von Stoppwörtern
tokens_remove(stopwords("english")) %>%

#Stemming
tokens_wordstem() %>%

#Wir erstellen die Feature Co-Occurrence-Matrix
fcm(context = "document") %>%

#Wir erstellen das semantische Netzwerk
textplot_network(min_freq = 5)
```



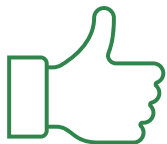
6. Outro

Vor- und Nachteile von Co-Occurrence-Analysen

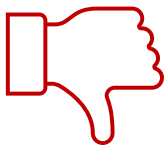


- Schnellere Analyse durch Reduktion auf relevante Features (z.B. nur Verben, Adjektive als Input für maschinelles Lernen)
- Inhaltliche Analysen für theoretische Fragen, z. B.
 - Wer spricht wie über wen? (z. B. Identifikation von Quellen und Inhalten)
 - Wie häufig findet sich Personalisierung? (z. B. Identifizierung von Named Entities)

Vor- und Nachteile von Co-Occurrence-Analysen



- Schnellere Analyse durch Reduktion auf relevante Features (z.B. nur Verben, Adjektive als Input für maschinelles Lernen)
- Inhaltliche Analysen für theoretische Fragen, z. B.
 - Wer spricht wie über wen? (z. B. Identifikation von Quellen und Inhalten)
 - Wie häufig findet sich Personalisierung? (z. B. Identifizierung von Named Entities)



- Unterschiedliche Pakete/Module = unterschiedliche Ergebnisse
- Analysen nicht immer fehlerfrei

Wie berichte ich Co-Occurrence Analysen in Papern?

3.3.1. Preprocessing

First, we identified collocations related to noun phrases (“climate change”) or named entities (“United States”). We then reduced our corpus to nouns, proper nouns, verbs, and adjectives to eliminate features with little discriminative value. Subsequently, we applied lower-case conversion, tokenization to unigrams, removed punctuation, and eliminated stop words unique to our corpus. We then applied relative pruning to remove extremely rare or frequent words.

← Kennt ihr jetzt auch!

Beispiel aus Hase et al. (2021). Climate change in news media across the globe: An automated analysis of issue attention and themes in climate change coverage in 10 countries (2006–2018). *Global Environmental Change*.



Wie berichte ich Co-Occurrence Analysen in Papern?

- **Immer:** Relevante Schritte kurz nennen & im Appendix ausführen
- **Noch besser:** Code (und ggf. Daten) teilen
- **Am besten:** Mit Multiverse-Analysen testen, wie robust Ergebnisse bei verschiedenen Paketen bzw. Modulen bleiben bzw. diese selbst validieren (s. **Sitzung 5!**)



Take-Aways



- **Ngrams:** Sequenz von N aufeinander folgenden Features
- **Keywords-in-Context:** Kontext von Schlüsselwörtern (d.h. Features um diese herum)
- **Collocations:** Aufeinander folgende Features, die überzufällig häufig gemeinsam auftreten, was z.B. auf eine gemeinsame semantische Bedeutung hindeutet
- **Semantische Netzwerke:** Netzwerk zur Visualisierung des gemeinsamen Auftretens von Features




Take-Aways







- **Part-of-Speech Tagging:** Analyse der Syntax, um Features Wortarten zuzuordnen
- **Dependency Parsing:** Analyse der semantischen Bedeutung von Features über ihre syntaktische Abhängigkeitsbeziehung zu einer “Root” bzw. “Wurzel”
(Dependenzstruktur)

Wie geht es weiter?

ZEITPLAN

 Mi, 24. Juli

- 09:00 - 12:00:  *Einführung & Preprocessing*
- 12:00 - 13:00:  *Mittagspause*
- 13:00 - 15:00:  *Co-Occurrence-Analysen*
- 15:00 - 17:00:  *Diktionäre*

 Do, 25. Juli

- 09:00 - 12:00:  *Topic Modeling*
- 12:00 - 13:00:  *Mittagspause*
- 13:00 - 15:00:  *Qualitätskriterien*
- 15:00 - 16:00:  *Ausblick*

Danke! Fragen?



Dr. Valerie Hase
IfKW, LMU Munich



valeriehase



valerie-hase.com



Luisa Kutlar
IfKW, LMU Munich



luisakutlar