



# Einführung in die Computerlinguistik Information Retrieval: tf.idf

---

Dr. Benjamin Roth & Annemarie Friedrich  
Centrum für Informations- und Sprachverarbeitung  
LMU München  
WS 2016/2017

# Referenzen

---

Dan Jurafsky & Christopher Manning: Natural Language Processing)

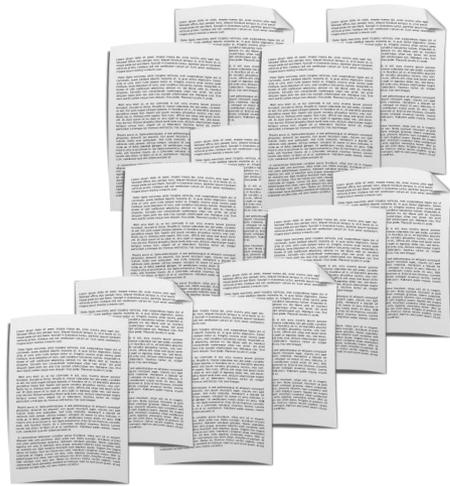
<https://web.stanford.edu/~jurafsky/NLPCourseraSlides.html>

→ Ranked Information Retrieval

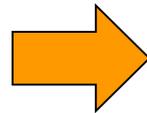
→ *Beispiele teilweise aus diesen Slides!*

# Information Retrieval

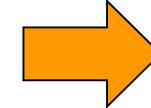
---



Document  
collection



Query/Suchanfrage:  
*ides of march*



Ergebnis

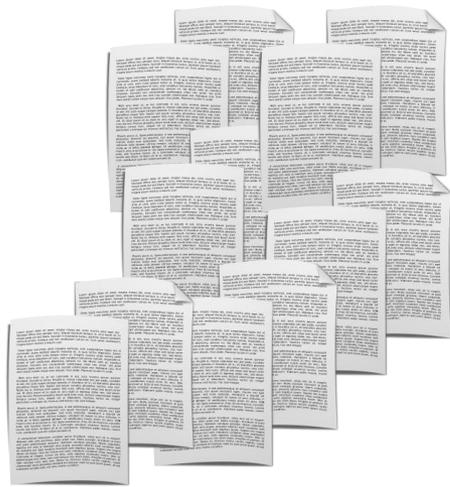
Beispiel:  
alle Dokumente, die die  
Suchbegriffe enthalten  
(Boolean retrieval)

## Problem:

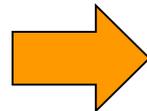
- Ergebnis enthält entweder keine oder zu viele Dokumente
- Dem Nutzer sollen nur ca. 10 Dokumente angezeigt werden

# Ranked Information Retrieval

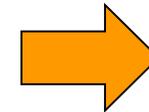
---



Document  
collection



Query/Suchanfrage:  
*ides of march*



1		0.97
2		0.92
3		0.83

...

Ergebnis

- Jedes Dokument erhält einen „Score“, der anzeigt, wie gut das Dokument und die Suchanfrage „zusammenpassen“.

# Beispiel: Query-Document Overlap

---



Dokument 1:  
*caesar died in march*

Suchanfrage:  
*ides of march*



Dokument 2:  
*the long march*

**Bag of words** = Dokumente werden als Menge der in ihnen vorkommenden Wörter (Terme) betrachtet

- Grammatik, Reihenfolge der Wörter wird ignoriert
- wie häufig ein Term vorkommt, kann berücksichtigt werden

# Beispiel: Query-Document Overlap

---



Dokument 1:  
*caesar died in march*

Suchanfrage:  
*ides of march*



Dokument 2:  
*the long march*

**Jaccard-Koeffizient** = misst Überschneidung zweier Mengen, Score zwischen 0 und 1

- $\text{jaccard}(A,B) = |A \cap B| / |A \cup B|$
- $\text{jaccard}(A,A) = 1$
- $\text{jaccard}(A,B) = 0$  if  $A \cap B = 0$
- $A$  und  $B$  müssen nicht gleich groß sein.

**Scores hier?**

# Ranking

---

- Jaccard-Koeffizient berücksichtigt nicht, wie häufig ein Wort (Term) in der Query oder in einem Dokument vorkommt
- **Wunschliste für Scoring:**
  - wenn ein Dokument einen Suchbegriff oft enthält, ist es vermutlich relevanter
  - generell selten vorkommende Begriffe sind vermutlich informativer

# Term-Document-Matrix

---

- Einträge: wie häufig kommt das Wort im Dokument vor?
- Jedes Dokument ist ein Vektor in  $\mathbb{N}^{|V|}$  (V=Vokabular)
  - Spalten in der Matrix!

## DOKUMENTE

<b>TERME</b>	<b>Antony and Cleopatra</b>	<b>Julius Caesar</b>	<b>The Tempest</b>	<b>Hamlet</b>	<b>Othello</b>	<b>Macbeth</b>
<b>Antony</b>	157	73	0	0	0	0
<b>Brutus</b>	4	157	0	1	0	0
<b>Caesar</b>	232	227	0	2	1	1
<b>Calpurnia</b>	0	10	0	0	0	0
<b>Cleopatra</b>	57	0	0	0	0	0
<b>mercy</b>	2	0	3	5	5	1
<b>worser</b>	2	0	1	1	1	0

# Term frequency

---

*term frequency*  $tf_{t,d}$  von Term  $t$  im Dokument  $d$  ist definiert als die Anzahl der Vorkommen von  $t$  in  $d$ .

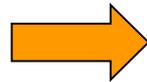
- frequency → hier: Anzahl („count“)
- term frequencies direkt verwenden funktioniert nicht gut
  - ein Dokument, das einen Suchbegriff 10x enthält ist vermutlich relevanter als eines, das den Begriff nur 1x enthält, jedoch nicht 10x so relevant
  - Relevanz wächst nicht proportional mit der term frequency

# Term frequency: Normalisierung

- einfache Variante

$$tf_{t,d} = \frac{\text{Anzahl Vorkommen von } t \text{ in } d}{\text{maxOccurrences}}$$

- *maxOccurrences* = maximale Anzahl von Vorkommen (irgendeines) Terms in *d* (*d.h. den häufigsten Term suchen und dessen Anzahl verwenden!*)



caesar	1
die	1
in	1
march	2
the	1
long	1



$$tf_{long,d1} = ?$$
$$tf_{march,d1} = ?$$

**Dokument d1:**  
*caesar died*  
*in march*  
*the long march*

# Document frequency

---

- seltene Begriffe sind informativer als häufige Begriffe
  - Stopwords
- Domänenabhängigkeit: nicht alle Begriffe sind in allen document collections gleich wichtig
- Begriffe, die in der document collection selten sind, sind informativer (falls sie in der Query vorkommen, wollen wir den Dokumenten, in denen sie vorkommen, einen hohen Score zuweisen)

# Document frequency

---

*document frequency*  $df_t$  von Term  $t$  ist definiert als die Anzahl der Dokumente, in denen  $t$  vorkommt.

- $df_t$  ist invers zur “Informativität” des Terms  $t$ 
  - in je mehr Dokumenten ein Term vorkommt, desto unwichtiger ist er für das Ranking

*inverse document frequency*  $idf_t$

$$idf_t = \log_{10} (N/df_t)$$

Warum log?

# Beispiel: inverse document frequency

---

- Es gibt genau einen Wert  $df_t$  für jeden Term  $t$  in einer document collection.

term	$df_t$	$idf_t$
calpurnia	1	
animal	100	
sunday	1,000	
fly	10,000	
under	100,000	
the	1,000,000	

$$idf_t = \log_{10} (N/df_t)$$

$$N = 1,000,000$$

# tf.idf Weighting

---

- Bekanntestes Weighting-Schema für Information Retrieval

tf.idf weight  $tf.idf_{t,d}$  eines Terms  $t$  bezüglich Dokument  $d$ :

$$\begin{aligned}tf.idf_{t,d} &= tf_{t,d} * idf_t \\ &= tf_{t,d} * \log(N/df_t)\end{aligned}$$

Was passiert, ...

- wenn  $tf_{t,d}$  steigt?
- wenn  $df_t$  steigt?

# Ranking von Dokumenten

---

- Ranking von Dokumenten für Query (Suchanfrage)  $q$ :

$$score(q, d) = \sum_{t \in q \cap d} tf \cdot idf_{t,d}$$

- Suchanfragen:
  - Wörter mit höherer *idf* haben größeren Einfluss auf das Ranking
- Hat die *idf* einen Effekt auf das Ranking von Dokumenten für Ein-Wort-Suchanfragen?

# Vector Space Model

---

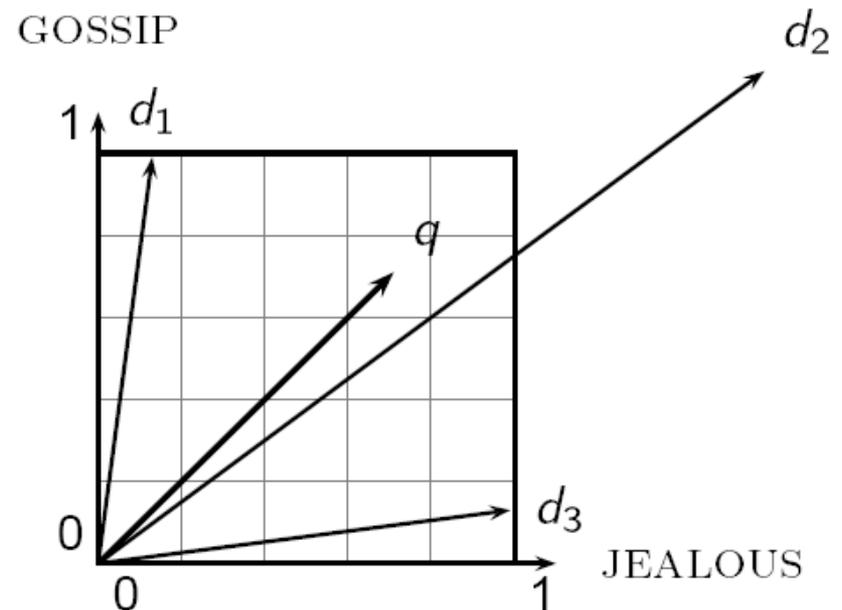
- Dokumente = Spaltenvektoren  $\in \mathbb{R}^{|V|}$
- Einträge:  $\text{tf.idf}_{t,d}$  für jeden Term  $t$  (Dimensionen)
- Suchanfrage wird genauso dargestellt („kurzes Dokument“)

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

# Vector Space Model

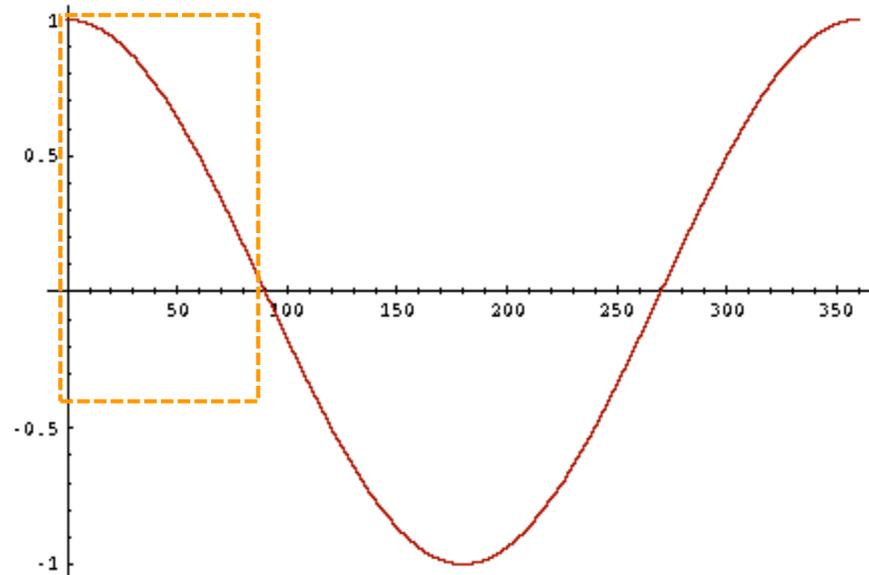
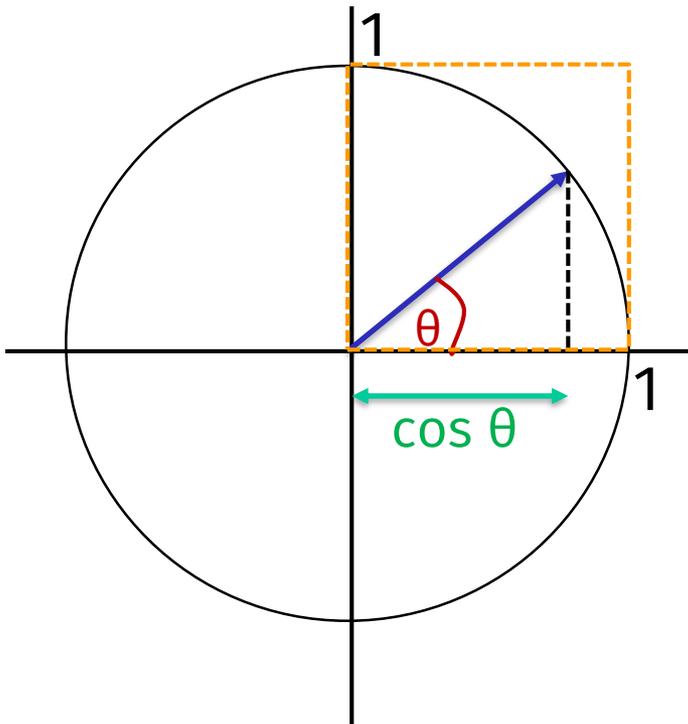
---

- Suchanfrage wird auch als Vektor dargestellt ( $q$ )
- Ranking von Dokumenten nach der **Ähnlichkeit** der Dokument-Vektoren mit dem Query-Vektor
  - hier: je kleiner der Winkel zwischen zwei Vektoren, desto ähnlicher die Vektoren



# Ranking mit Kosinus-Ähnlichkeit

- Ranking: Dokumente mit kleinerem Winkel zur Suchanfrage zuerst
- Kosinus: im Intervall  $[0^\circ, 180^\circ]$  monoton fallende Funktion



# Ranking mit Kosinus-Ähnlichkeit

---

- Dokumente und Query werden jeweils als Vektoren dargestellt
- Ähnlichkeit zwischen Query-Vektor und Vektor von Dokument d:

$$\cos(\vec{q}, \vec{d}) = \frac{\vec{q} * \vec{d}}{|\vec{q}| * |\vec{d}|} = \frac{\sum_{i=1}^{|\mathcal{V}|} q_i d_i}{|\vec{q}| * |\vec{d}|}$$

- Vereinfachung für das Projekt:
  - $|\vec{q}|$  kann weggelassen werden
  - Nenner: es reicht aus, über die Dimensionen (Terme) zu iterieren, die in der Query vorkommen! (warum?)

# Betrag eines Vektors

---

- Länge eines Vektors

$$\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$$

$$\vec{x} = (2, 3, 4)$$

$$|\vec{x}| = ?$$

# Beispiel

---

- Berechnen Sie die Kosinus-Ähnlichkeit zwischen den drei Novellen (ohne tf.idf-Gewichtung)
- **SaS**: *Sense and Sensibility*, **PaP**: *Pride and Prejudice*, and **WH**: *Wuthering Heights*

term	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6
wuthering	0	0	38