Maschinelles Lernen
- Informatik -

Rapp, DHBW Lörrach

12.01.2024

Inhaltsübersicht

Themenübersicht

Vorläufiger Vorlesungsplan

Datum	12.01.2024	19.01.2024	26.01.2024	02.02.2024	09.02.2024	16.02.2024	23.02.2024	01.03.2024	08.03.2024	15.03.2024
Thema	Ranking im Vektorraum Modell	Empfehlungsdienst	Künstliche Neuronale Netze	Transformer	Computer Vision	Behavior Cloning	Zeitreihen-Vorhersage	Aufgaben-bearbeitung	Abschlusspräsentation	Ersatztermin
Beispiele	Term Frequency - Inverse Document Frequency	Mean Average Precision@K-Metrik	Long-Short-Term Memory (LSTM)	BERT, GPT	Convolutional Neural Network (CNN)	Autonomes Fahren	Hidden Markov Modell			

Gruppeneinteilung Prüfungsleistung und Projektbeschreib.

Gruppe	Name	Gruppensprecher
A	Nick Riesterer	
	Luis Hilbert	
	Marc Scheller	
	Uwe Meyer	
	Colin Vavra	X
B	Nico Römer	
	Jessica Prall	
	Khemjira Sakulkla	
	Méline Broutin	
	Robin Liebschwager	X
C	Mikail Şen	
	Patrick Furtwängler	
	Niclas Gugel	
	Marvin Obert	
	Ugurtan Can Cetin	X
D	Antonio Bellizzi	
	Linus Fischer	X
	Florian Grundmann	
	Samuel Lobenstein	
	Christopher Ströbele	

Die perfekte Kunstmesse!

Entwickeln Sie ein KPI-To-Action Framework für ein hybrides (d.h. physisches und virtuelles) Kunstevent mit dem Ziel, das Verkaufsvolumen und die Profitabilität zu maximieren.

Prüfungsleistung

Kombinierte Prüfung mit insgesamt 1 Note

1. 5. Semester: schriftliche Klausur
2. 6. Semester: Programmentwurf

Liefergegenstände für Programmentwurf je Gruppe auf Moodle

- **Programmcodes** zu den einzelnen Aufgaben einschl. **Inline-Dokumentation**

Liefertermine auf Moodle

- **Do 08.02.2024:** Aufgaben 1-4
- **Fr 01.03.2024:** 2 Wahlaufgaben aus den Aufgaben 5, 6 und 7
- **Do 07.03.2024:** Präsentationsfolien (z.B. PDF)

Abgabetermin auf Moodle	Aufgaben-Nr.	Aufgabentitel
Do 08.02.2024	1	Ranking von Art Stories
	2	Empfehlung von Kunstwerken
	3	Art Ontology for ad-hoc SPARQL-reporting
	4	Art Chatbot
Fr 01.03.2024	5	Instanzensegmentierung von Kunstwerken
	6	Show Robot
	7	Dynamic Pricing Strategie für Ticketverkäufe

Die Abschlusspräsentationen finden planm. am 08.03.2024 während der regulären Vorlesungszeit statt.

Prüfungsaufgabe 1

a) Programmieren Sie eine Python-Funktion für das Ranking von Art Stories im Word2vec Embedding, die einer Anfrage durch einen Besucher Ihrer Kunst-Website am ähnlichsten sind.
b) Beurteilen Sie den Einfluss der Erweiterung auf das Doc2Vec Embedding auf die Ergebnisqualität.
c) Programmieren Sie einen Empfehlungsdienst für Art Stories durch Clustering mit Top2Vec.

Rahmen

- *zu a)*
  - **Funktionsname:** rank_art_stories_python_function(user_query_string)
  - **return:** Gerankte Art Stories mit jeweiligem Ähnlichkeitsmaß als Pandas DataFrame
- *zu c)*
  - **Funktionsname:** recommend_art_stories_python_function(identifier_-of_visited_art_stories_list)
    z.B. hat der Kunst-Website Besucher zuvor die Art Stories mit den eindeutigen Identifiern 1,4 und 7 gelesen → [1,4,7].
  - **return:** Empfehlung der Top-3 Art Stories in gerankter Liste mit jeweiligem Ähnlichkeitsmaß als Pandas Dataframe.
    z.B. lautet die Empfehlung die Stories [3,5,6] zu lesen mit den zugehörigen Ähnlichkeitsmassen [0.8,0.6,0.3]
- **Datenquelle:** *https://www.artbasel.com/stories-features*

Beispiele für Bewertungskriterien

- Die Modellkomplexität deckt die Anforderungen potentieller Besucher der Website für die Anzeige ähnlicher Art Stories ab.
- Die Ergebnisqualität des Rankings kann aus dem Programmcode nachvollzogen werden.
- Aus dem Programmcode ist ersichtlich, dass die wesentlichen Konzepte aus der Vorlesung verstanden und auf die konkrete Problemstellung angewendet wurden.
- Der Fremdanteil am Programmcode ist mit entsprechenden Quellenangaben gekennzeichnet.
- ...

Tipp: Betrachten Sie die Art Stories als Documents.

Überblick

Suchmaschine vs. Empfehlungsdienst

Fragestellung

Wie können wir den Nutzer dabei unterstützen, relevante Informationen zu erhalten?

Suchmaschine	**Empfehlungsdienst**
→ **Pull-Modus**, d.h. **User ergreift Initiative** → Ad hoc Informationsbedürfnis	→ **Push-Modus**, d.h. **System ergreift Initiative** → System zieht Rückschlüsse auf User-Bedürfnisse

Datentypen

Unstrukturierte Daten

- Können nicht in Zeilen, Spalten und relationalen Datenbanken dargestellt werden
- z.B. Texte, (Bild-) Dateien, Video, Emails, PowerPoint
- entspricht ca. 80% der Unternehmensdaten *(lt. Gartner)*

Semistrukturierte Daten

- unterliegen keiner allgemeinen Struktur, enthalten jedoch Metadaten (z.B. Tags) zur besseren Verarbeitung
- z.B. HTML, XML

Strukturierte Daten

- wurden in vorgegebenem Format strukturiert, bevor sie im Datenspeicher abgelegt wurden *(auch: Schema-on-Write)*
- relationale Datenbanken mit präzisen Feldern wie z.B. Kreditkartennummer oder Adresse
- für Algorithmen des maschinellen Lernens einfach nutzbar (z.B. Bearbeitung, Abfrage)

The university has 5600 students.
John's ID is number 1, he is 18 years old and already holds a B.Sc. degree.
David's ID is number 2, he is 31 years old and holds a Ph.D. degree. Robert's ID is number 3, he is 51 years old and also holds the same degree as David, a Ph.D. degree.

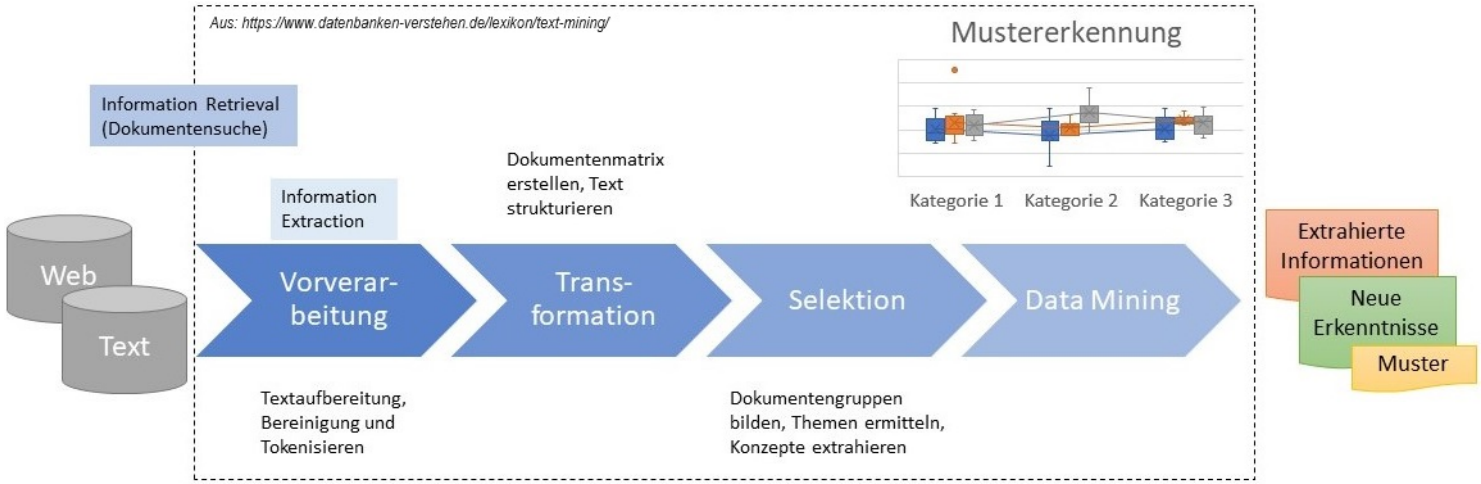
```
<University>
 <Student ID="1">
   <Name>John</Name>
   <Age>18</Age>
   <Degree>B.Sc.</Degree>
 </Student>
 <Student ID="2">
   <Name>David</Name>
   <Age>31</Age>
   <Degree>Ph.D. </Degree>
 </Student>
 ....
</University>
```

ID	Name	Age	Degree
1	John	18	B.Sc.
2	David	31	Ph.D.
3	Robert	51	Ph.D.
4	Rick	26	M.Sc.
5	Michael	19	B.Sc.

Idea Group Inc.

Beispiele für unstrukturierte *(links)*, semistrukturierte *(Mitte)* und strukturierte *(rechts)* Daten.

Text Mining Prozess



Information Retrieval *(auch: Dokumentensuche)*

- Sammeln von Daten aus verschiedenen Quellen (z.B. Web, Emails) zu einem Thema, das mithilfe einer Query abgefragt werden kann *(→ klass. Suchmaschine)*
- Spezialfall **„Text Retrieval“** bezieht sich auf Textdaten.

Information Extraction

- Suche nach Fakten in Texten, um eine zusammengefasste Darstellung des Dokuments zu erhalten.

Natural Language Processing (NLP)

- Aufbereitung, Bereinigung und Tokenisieren von unstrukturierten Texten mithilfe von linguistischen Methoden des maschinellen Lernens. Die daraus erzeugten strukturierten Daten können anschließend weiter analysiert werden.

Data Mining

- Generierung von Informationen aus Daten mithilfe von statistischen Methoden.
- Spezialfall *„Text Mining“* bezieht sich auf unstrukturierte Textdaten.

Text Retrieval

Problemformulierung Text Retrieval

- **Vokabular:** $V = \{w_1, ..., w_N\}$ der Sprache
  - z.B. Oxford English dictionary: insgesamt ca. 200.000 Wörter aktuell in Gebrauch
  - testyourvocab.com: erwachsener Muttersprachler hat ca. 20.000 englische Wörter abhängig vom Ausbildungsgrad etc.
- **Query:** $q = q_1, ..., q_m$, wobei $q_i \in V$
  - z.B. $q =$„presidential campaign news“
- **Dokument:** $d_i = d_{i1}, ..., d_{im_i}$, wobei $d_{ij} \in V$
  - z.B. PDF-Dokument mit Wörtern aus dem Vokabular V
- **Collection (Korpus):** $C = \{d_1, ..., d_M\}$ *(auch: Textkorpus)*
  - Sammlung von maschinenlesbaren, natürlich vorkommenden Texten, die oft nach bestimmten Kriterien selektiert wurden
  - Beispiele: Wikipedia, British National Corpus, etc.
- **Menge relevanter Dokumente:** $R(q) \subseteq C$
  - *Unbekannt* und *abhängig vom Nutzer!*
  - Query kann lediglich einen „*Hinweis*“ darauf liefern, welches Dokument in $R(q)$ enthalten sein könnte

Zielformulierung des Text Retrieval Tasks

Berechnung von $R'(q)$ als Approximation von $R(q)$.

Berechnungsstrategien

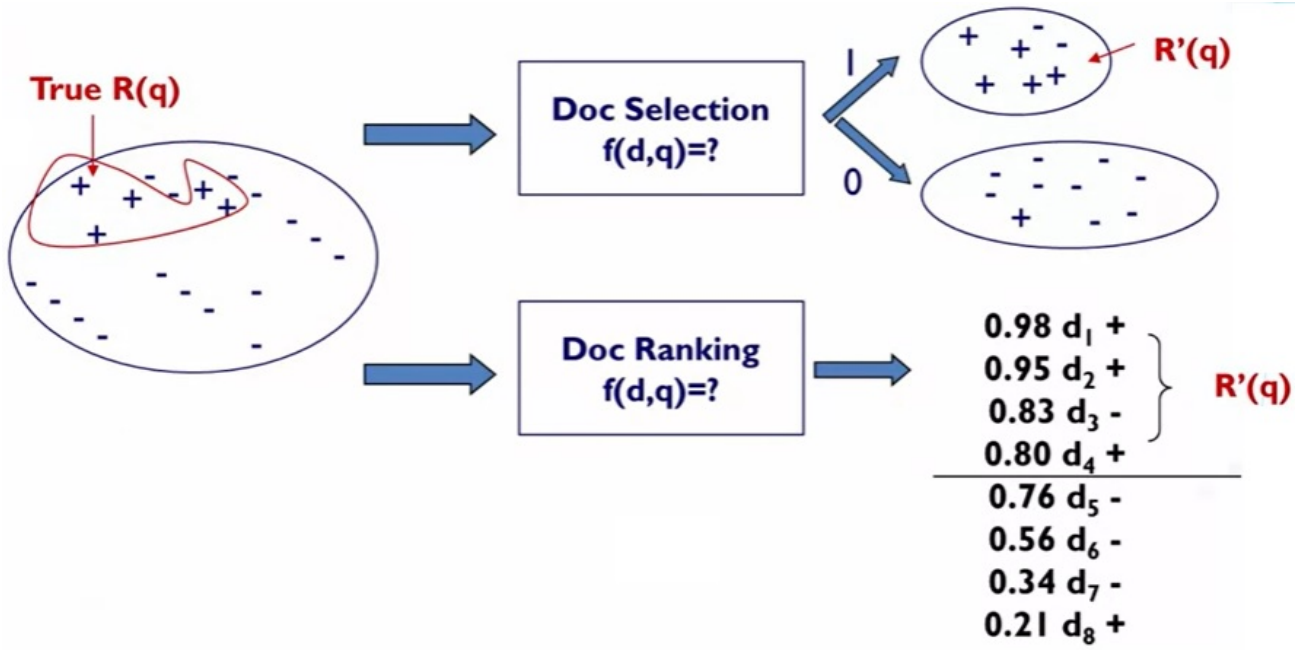
Zwei Berechnungsstrategien von R'(q) für die Approximation von R(q)

1. **Dokument-Auswahl**
   - $R'(q) = \{c \in C | f(q,d) = 1\}$, mit $f(q,d) \in \{0,1\}$ *binärer Klassifikator* (1: relevant, 0: nicht relevant)
     $\rightarrow$ **Absolute Relevanz:** System entscheidet, ob ein Dokument relevant ist oder nicht.
2. **Dokument-Ranking**
   - $R'(q) = \{d \in C | f(q,d) > \theta\}$, mit $f(q,d) \in \mathbb{R}$ *Relevanzmaßfunktion* und θ manuell festgelegter *Schwellwert*
     $\rightarrow$ **Relative Relevanz:** Das System entscheidet lediglich, ob ein Dokument relevanter ist als ein anderes.

Dokument-Auswahl vs. -Ranking

Probability Ranking Principle

Das Probability Ranking Principle stellt sicher, dass Relevanz eines Dokuments für eine Query eine probabilistische Interpretation hat.



Dokument-Ranking stellt die optimale Strategie *(vgl. Robertson, '77)* dar, wenn

1. der Nutzen eines Dokuments für einen User **unabhängig** vom Nutzen anderer Dokumente ist und
2. sich der User **sequentiell** durch die einzelnen Dokumente klickt

Erstellung einer Ranking Funktion

Ranking-Funktion *(auch: „Relevanzmaßfunktion“)* $f(q, d) \in \mathbb{R}$

- **Query:** $q = q_1, ..., q_m$, wobei $q_i \in V$
- **Dokument:** $d = d_1, ..., d_n$, wobei $d_i \in C$

Eine **sinnvolle** Ranking-Funktion rankt für eine Query **relevante oberhalb von nicht-relevanten** Dokumenten.

Größte Herausforderung

Wie messen wir die Wahrscheinlichkeit, dass Dokument d für den User **relevant** in Bezug auf seine Query q ist?

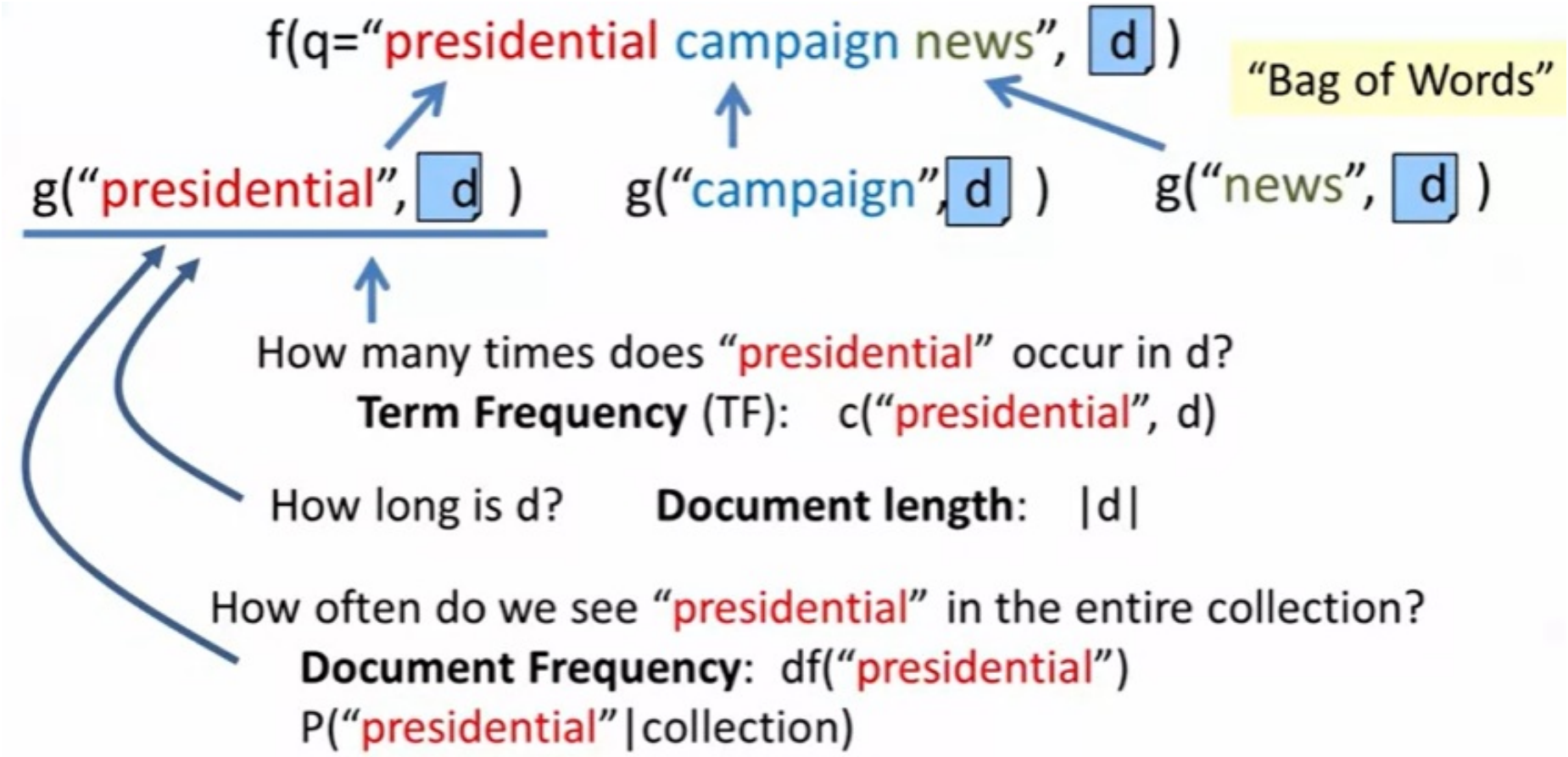
Gesucht!

Beispiele für Text Retrieval Modelle

1. **Ähnlichkeitsbasierte Modelle:** $f(q,d) = similarity(q,d)$
   - z.B. **Vector Space Model (VSM)**
2. **Probabilistische Modelle:** $f(q,d) = p(R = 1|q,d)$, wobei $R \in \{0,1\}$
   - Klassisches probabilistisches Modell
   - Sprachmodell
   - Divergence-From-Randomness Modell
3. **Probabilistisches Inferenz Modell:** $f(q,d) = p(d \rightarrow q)$
4. **Axiomatisches Modell:** $f(q,d)$ erfüllt bestimmte Bedingungen

Alle diese Modelle resultieren in **ähnlichen Ranking Funktionen** und ähnlichen Variablen.

Ideen in State-Of-The-Art Retrieval Modellen



From: Text Retrieval and Search Engines,
Zhai, University of Illinois at Urbana-Champaign

Modellvergleich von Retrieval Modellen

Die folgenden Modelle erzielen vergleichbare Ergebnisse (Fang, '11)

1. BM25 *(beliebteste Methode)*
2. Pivoted Length Normalization
3. Query Likelihood

Zusammenfassung

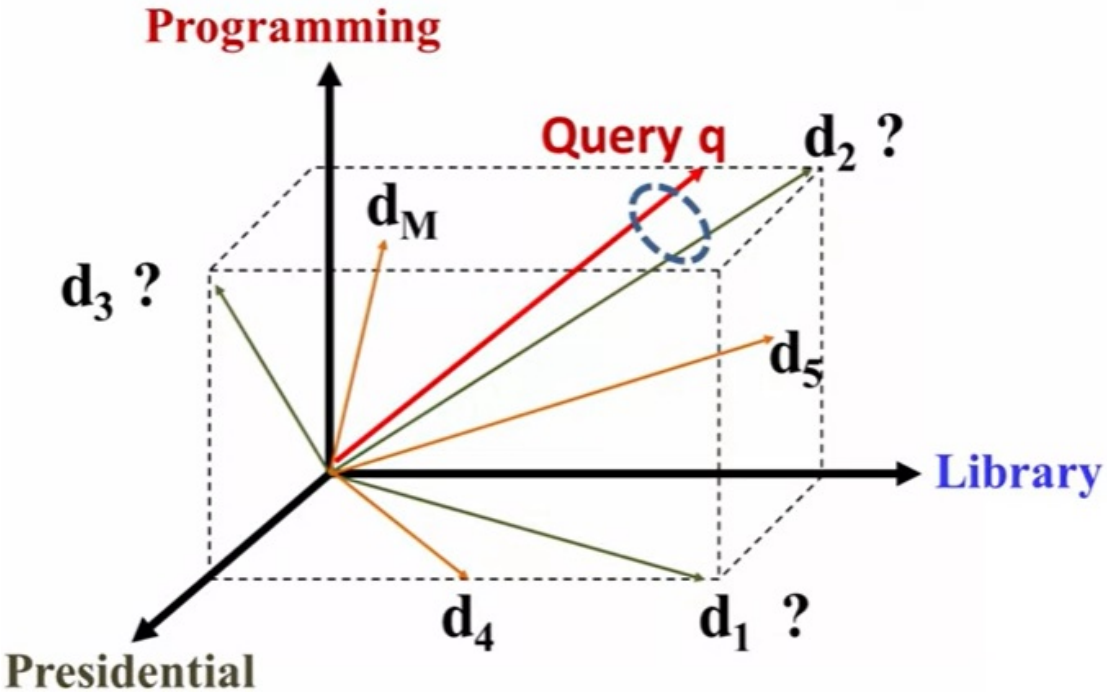
- Erstellung einer Ranking-Funktion $f(q, d)$ erfordert ein **Retrieval Modell** mit einer **formellen Definition** von **Relevanz.**
- Viele Modelle sind **ähnlich effektiv** und es gibt **keinen klaren Gewinner.**
- State-Of-The-Art Ranking Funktionen basieren auf
  - Bag Of Words Darstellung
  - Dokumentenlänge $|d|$
  - Term Frequency (TF) und Document Frequency (DF) der Wörter

Vektorraum-Modell

Übersicht Vector Space Modell

Vektorraum-Modell *(engl. Vector Space Modell, VSM)*

- ähnlichkeitsbasiertes Modell: $f(q, d) = similarity(q, d)$
- auch: *„Vektorraum-Retrieval“*



Informationen werden als Punkte in einem hochdimensionalen Vektorraum dargestellt.

Vector Space Modell als Framework

Das **Vector Space Modell ist ein Framework** *('Ordnungsrahmen')*
Darstellung von **Dokumenten** und **Queries** durch einen **Term** Vektor.

Term

- Grundlegendes Konzept (z.B. Wort)
- Jeder Term (z.B. aus Vokabular) definiert eine Dimension
- N Terme definieren einen **N-dimensionalen Vektorraum**

Query Vektor
$q = (x_1, ..., x_N)$, wobei die $x_i \in \mathbb{R}$ **Query Term Gewichte** sind
Dokument Vektor
$d = (y_1, ..., y_N)$, wobei die $y_i \in \mathbb{R}$ **Dokument Term Gewichte** sind

Grundlegende Annahme

Die Relevanz eines Dokuments d in Bezug auf eine User-Query q ist proportional zur Ähnlichkeit zwischen Query und Dokument:

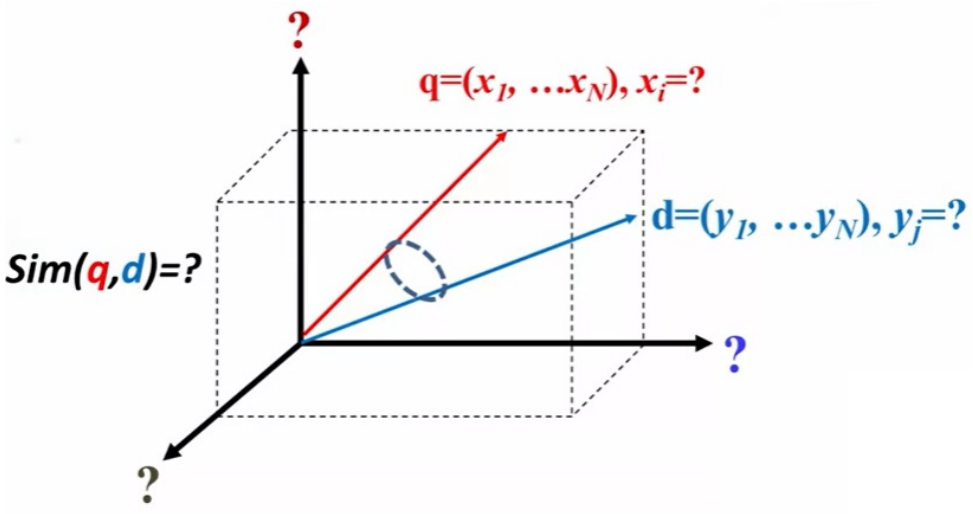
$$relevance(q, d) \propto similarity(q, d)$$

Ähnlichkeitsmaß

Das Ähnlichkeitsmaß ($similarity = sim$) wird durch die Ranking-Funktion f ausgedrückt: $similarity(q, d) = f(q, d)$

Vector Space Modell - Fragen

Was sagt das Vector Space Modell nicht aus?



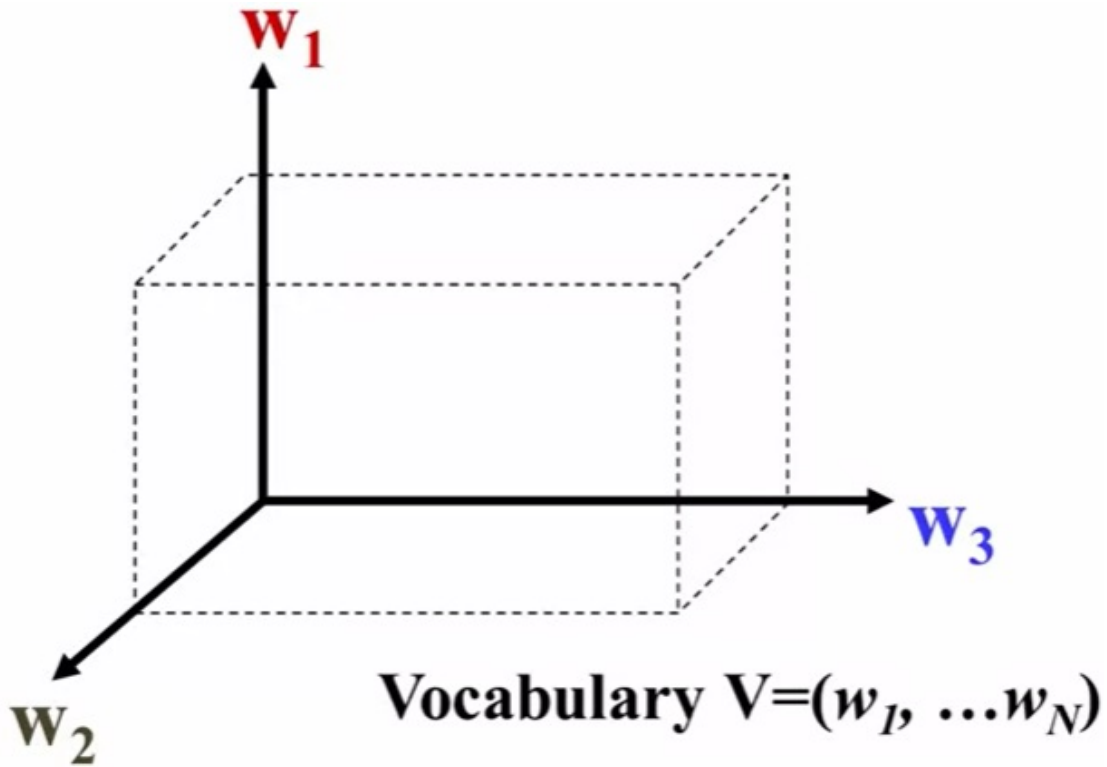
- Wie soll das grundlegende Konzept definiert werden?
  - Konzepte werden als **orthogonal** angenommen

  Wie werden die Term Gewichte von Dokumenten und Queries im Raum festgelegt?
  - Term Gewicht einer Query gibt die **Wichtigkeit eines Terms** an
  - Term Gewicht eines Dokuments gibt an, **wie stark der Term das Dokument charakterisiert**
- Wie wird das Ähnlichkeitsmaß $Sim(q, d)$ definiert?

Einfachste Instanziierung

Vector Space Modell - Instanziierung

Instanziierung der N Vektorraumdimensionen durch Bag Of Words



Bag-Of-Words

Bag-Of-Words (BOW)

- Jedes Dokument des Korpus wird in der BOW-Darstellung als Vektor dargestellt
- Sortierung z.B. alphabetisch oder nach Häufigkeit des Auftretens
- Je häufiger das Wort, desto wichtiger ist es innerhalb des Satzes der BoW-Darstellung. Hinweis: Problem bei häufig vorkommenden (z.B. Stopp-) Wörtern.

Beispiel-Darstellung als JSON

- Key: Wort. Ausgenommen Stoppwörter *(z.B. for)*
- Value: Worthäufigkeit. Im Ggs. zu binärem *„One-Hot-Encoding“*
- Wortreihenfolge spielt keine Rolle

Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.

Term	Document 1	Document 2
aid	0	1
all	0	1
back	1	0
brown	1	0
come	0	1
dog	1	0
fox	1	0
good	0	1
jump	1	0
lazy	1	0
men	0	1
now	0	1
over	1	0
party	0	1
quick	1	0
their	0	1
time	0	1

Beispiele

- $BoW1 = \{'the' : 2, 'quick' : 1, 'brown' : 1, 'fox' : 1, 'jumped' : 1, 'over' : 1, 'lazy' : 1, 'dog' : 1, 'back' : 1\}$
- $BoW2 = \{'now' : 1, 'the' : 2, 'time' : 1, 'all' : 1, 'good' : 1, 'men' : 1, 'come' : 1, 'aid' : 1, 'their' : 1, 'party' : 1\}$
  Union:
- $BoW1 + BoW2 = \{'now' : 1, 'the' : 4, 'time' : 1, 'all' : 1, 'good' : 1, 'men' : 1, 'come' : 1, 'aid' : 1, 'their' : 1, 'party' : 1, 'quick' : 1, 'brown' : 1, 'fox' : 1, 'jumped' : 1, 'over' : 1, 'lazy' : 1, 'dog' : 1, 'back' : 1\}$

Tokenization und Stemming

Tokenization

Zerlegung von Text in logisch zusammengehörige Einheiten *('Tokens')*.

- Wörter mit ähnlicher Bedeutung sollen auf denselben Index-Term gemappt werden.

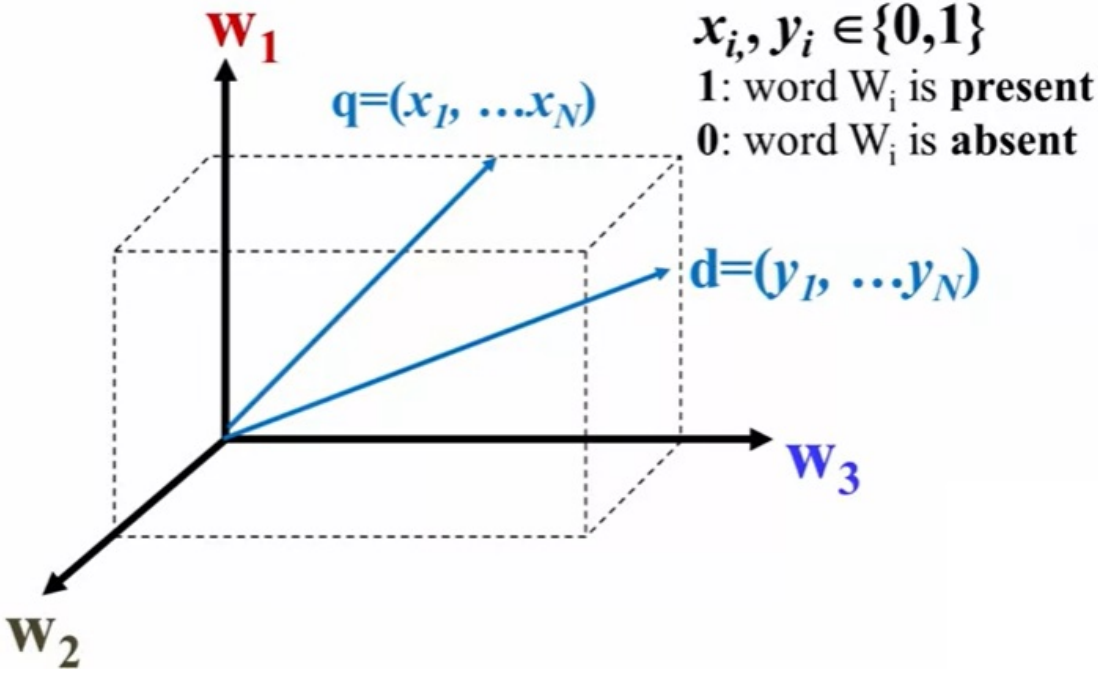
Stemming *(='Stammformreduktion')*

Zurückführen sämtlicher Flexionen auf einen gemeinsamen Wortstamm, z.B.

- computer $\rightarrow$ compute
- computation $\rightarrow$ compute
- computing $\rightarrow$ compute

Bit Vektor

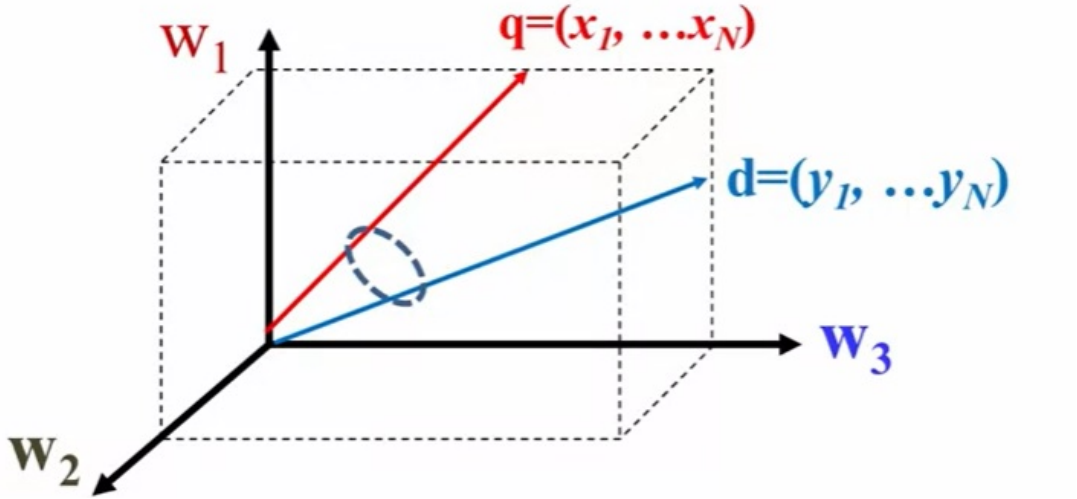
Vektordarstellung: Bit Vektor



Skalarprodukt als Ähnlichkeitsmaß

Ähnlichkeitsinstanziierung durch Skalarprodukt

$$Sim(q,d) = q.d = x_1 y_1 + \ldots + x_N y_N = \sum_{i=1}^{N} x_i y_i$$



Ranking im einfachsten Vector Space Modell

Einfachstes Vector Space Modell

Einfachstes VSM = Bit-Vektor + Skalarprodukt + BOW

$q=(x_1, \ldots x_N)$ $\quad x_i, y_i \in \{0,1\}$

$d=(y_1, \ldots y_N)$ $\quad$ **1**: word W_i is **present**

0: word W_i is **absent**

$$Sim(q,d)=q.d= x_1 y_1 + \ldots + x_N y_N = \sum_{i=1}^{N} x_i y_i$$

Zentrale Fragestellungen

- Welche Informationen bildet diese Rankingfunktion ab bzw. nicht ab?
- Stellt dies eine gute Rankingfunktion dar?

Beispiel

Frage: Wie würden Sie diese Dokumente ranken?

Query = "news about presidential campaign"

Ideal Ranking?

d1 … news about …

d2 … news about organic food campaign…

d3 … news of presidential campaign …

d4 … news of presidential campaign …
… presidential candidate …

d5 … news of organic food campaign…
campaign…campaign…campaign…

Beispiel

Frage: Wie würden Sie diese Dokumente ranken?

Query = "news about presidential campaign"

		Ideal Ranking?
d1	… news about …	d4 +
d2	… news about organic food campaign…	d3 +
d3	… news of presidential campaign …	
d4	… news of presidential campaign … … presidential candidate …	d1 -
d5	… news of organic food campaign… campaign…campaign…campaign…	d2 - d5 -

Beispiel

Ranking mit dem einfachsten VSM

Query = 'news about presidential campaign'

d1 … news about …

d3 … news of presidential campaign …

V= {news, about, presidential, campaign, food …. }

q= (1, 1, 1, 1, 0, …)

d1= (1, 1, 0, 0, 0, …)

f(q,d1)= 1*1+1*1 +1* 0+1*0+0*0 + …= 2

d3= (1, 0, 1, 1, 0, …)

f(q,d3)= 1*1+1*0 +1*1 +1*1+0*0+ … = 3

Beispiel

Ist das einfachste VSM effektiv?

Query = "news about presidential campaign"

d1	… **news about** …	f(q,d1)=2
d2	… **news about** organic food **campaign**…	f(q,d2)=3
d3	… **news** of **presidential campaign** …	f(q,d3)=3
d4	… **news** of **presidential campaign** … … **presidential** candidate …	f(q,d4)=3
d5	… **news** of organic food **campaign**… **campaign**…**campaign**…**campaign**…	f(q,d5)=2

Zusammenfassung

VSM Instanziierungen

1. Dimension
2. Vektor Darstellung
3. Ähnlichkeit

Einfachstes VSM

- Dimension = Wort
- Vektor = 0-1 Bit Vektor (Wort vorhanden/nicht vorhanden)
- Ähnlichkeitmaß = Skalarprodukt

f(q,d) stellt im einfachsten VSM die Anzahl eindeutiger Query Wörter dar, die in *d* gematcht wurden.

Verbesserte Instanziierung der Vektor Darstellung

Probleme

Zwei Probleme mit dem einfachsten VSM

Query = "news about presidential campaign"

d2	… **news about** organic food **campaign**…	**f(q,d2)=3**
d3	… **news** of **presidential campaign** …	**f(q,d3)=3**
d4	… **news** of **presidential campaign** … … **presidential** candidate …	**f(q,d4)=3**

1. Mehrfaches Matching von **presidential** sollte die Relevanz steigern.
2. Matching von **presidential** wäre wichtiger als **about.**

Term-Frequency

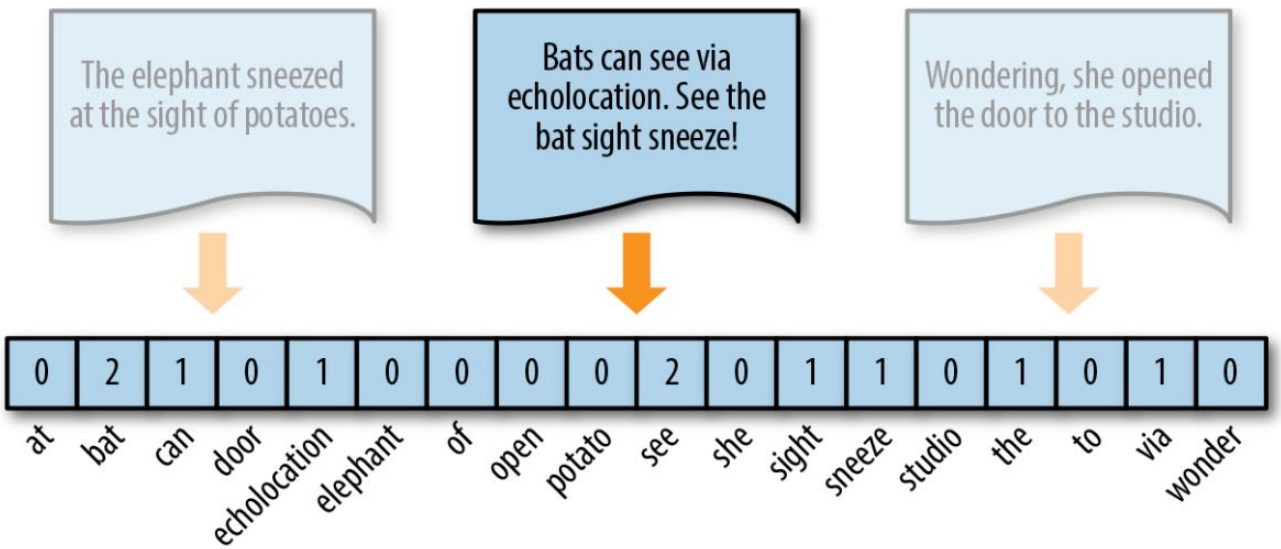
Term-Frequency (normalisiert)

$tf_{i,j}$ = Anzahl des Auftretens des Terms i in Dokument j

/

Anzahl Terme im Dokument

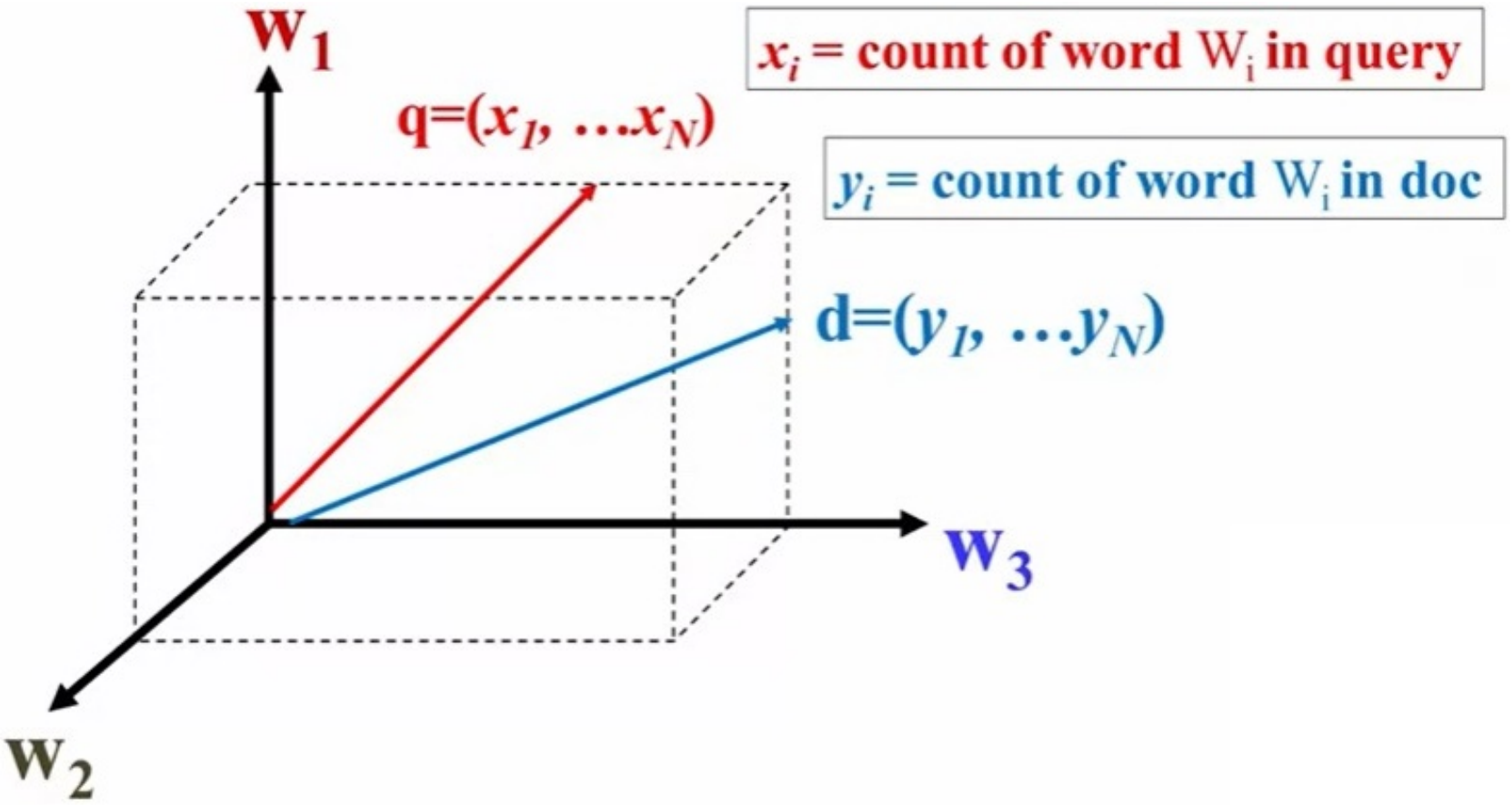
Beispiel



$tf_{bat,2} = 0.2$

Verbesserte Instanziierung der Vektor Darstellung

Term Frequency Vector: Verbesserte Vektorplatzierung im Vektorraum



Verbesserte Instanziierung der Vektor Darstellung

Verbessertes VSM mit Term Frequency Gewichtung

$q=(x_1, \ldots x_N)$ $\quad x_i$ = count of word W_i in query

$d=(y_1, \ldots y_N)$ $\quad y_i$ = count of word W_i in doc

$$Sim(q,d)=q.d= x_1 y_1 + \ldots + x_N y_N = \sum_{i=1}^{N} x_i y_i$$

Zentrale Fragestellungen

- Welche Informationen bildet diese Rankingfunktion ab bzw. nicht ab?
- Werden die Probleme des einfachsten VSM behoben?

Ranking

Ranking mit Term Frequency Gewichtung

Query: 'news about presidential campaign'

d2 … news about organic food campaign… f(q,d2)=3

q= (1, 1, 1, 1, 0, …)
d2= (1, 1, 0, 1, 1, …)

d3 … news of presidential campaign … f(q,d3)=3

q= (1, 1, 1, 1, 0, …)
d3= (1, 0, 1, 1, 0, …)

d4 … news of presidential campaign …
… presidential candidate … f(q,d4)=4!

q= (1, 1, 1, 1, 0, …)
d4= (1, 0, 2, 1, 0, …)

Zweites Problem

Wie lösen wir das zweite Problem **presidential** vs. **about**?

Query: 'news about presidential campaign'

d2 … news about organic food campaign…

d3 … news of presidential campaign …

V= {news, about, presidential, campaign, food …. }

q= (1, 1, 1, 1, 0, …)

d2= (1, 1, 0, 1, 1, …)

q= (1, 1, 1, 1, 0, …)

d3= (1, 0, 1, 1, 0, …)

Ziel:

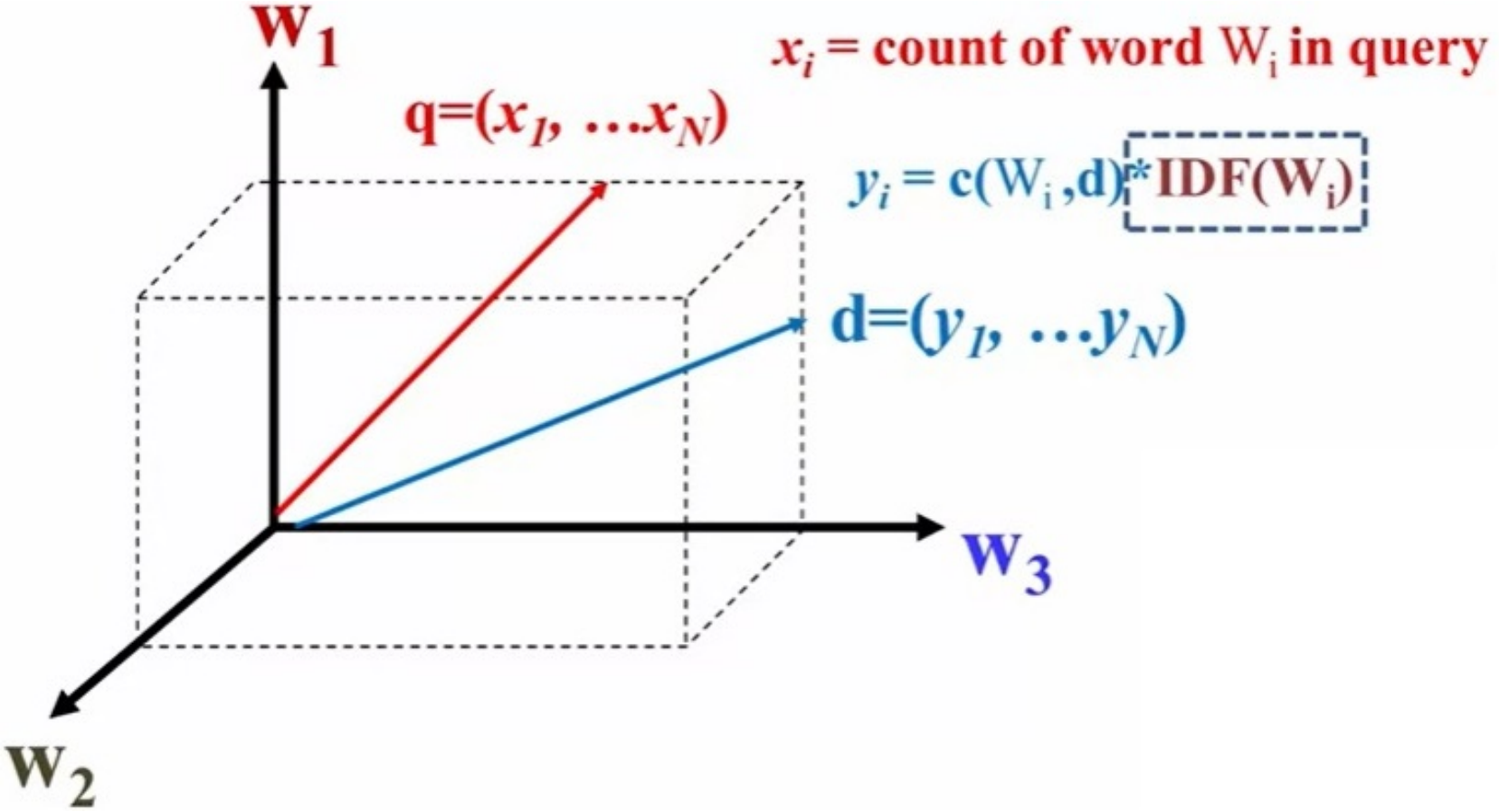
f(q,d2)<3

f(q,d3)>3

Inverse Document Frequency

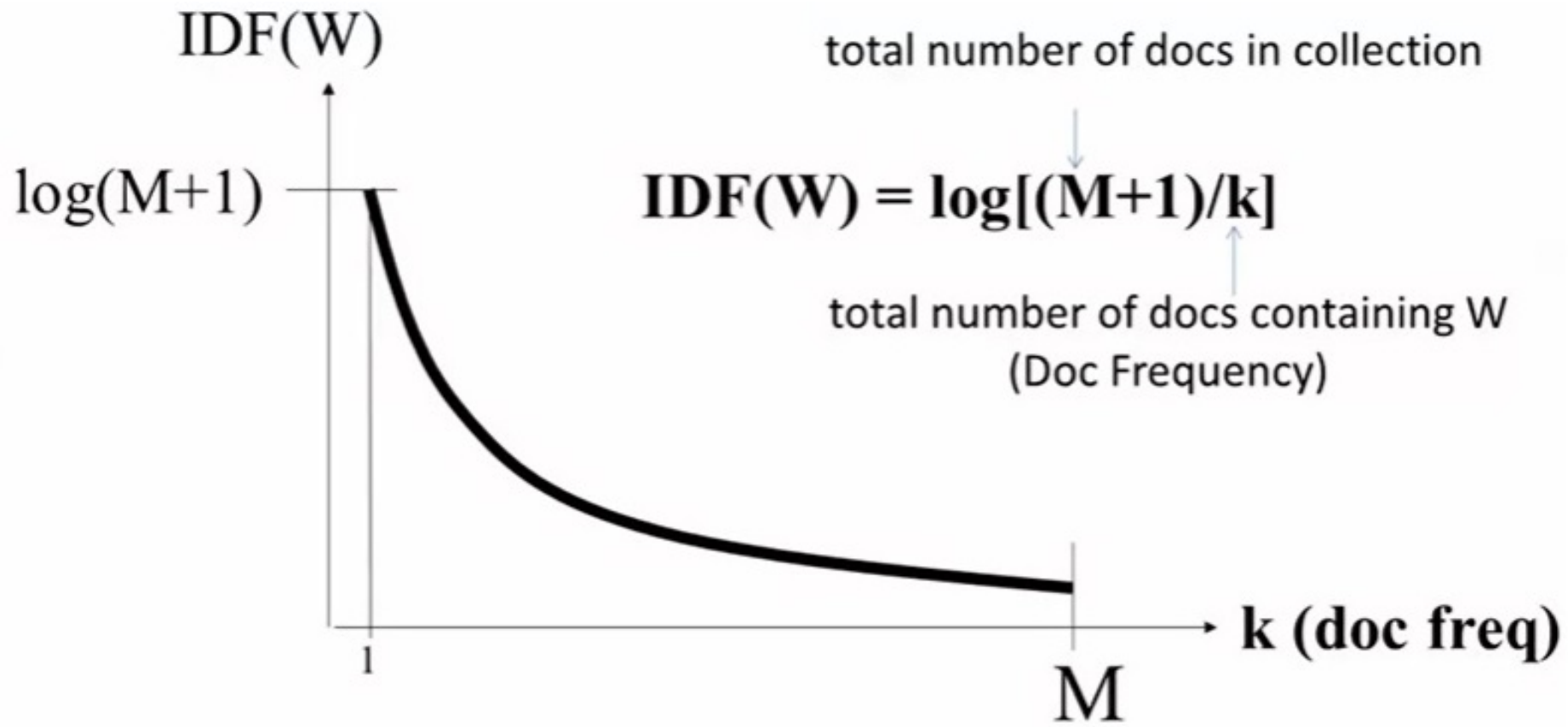
Weitere Verbesserung: IDF

Inverse Document Frequency: Weitere Verbesserung der Vektorplatzierung im Vektorraum



IDF Gewichtung

IDF Gewichtung: Bestrafung allgemeiner Begriffe



Lösung des zweiten Problems

presidential vs. **about**

d2 … news about organic food campaign…

d3 … news of presidential campaign …

Vocabulary	news	about	presidential	campaign	food
IDF	log 6/5	log 6/2	log 6/2	log 6/4	log 6/2
TF (d2)	1	1	0	1	1
q	1	1	1	1	0
Sim(q,d2)	log 6/5	log 6/2	0	log 6/4	0
TF (d3)	1	0	1	1	0
q	1	1	1	1	0
Sim(q,d3)	log 6/5	0	log 6/2	log 6/4	0
TF (d5)	1	0	0	4	1
q	1	1	1	1	0
Sim(q,d5)	log 6/5	0	0	4*log 6/4	0

Lösung des zweiten Problems:

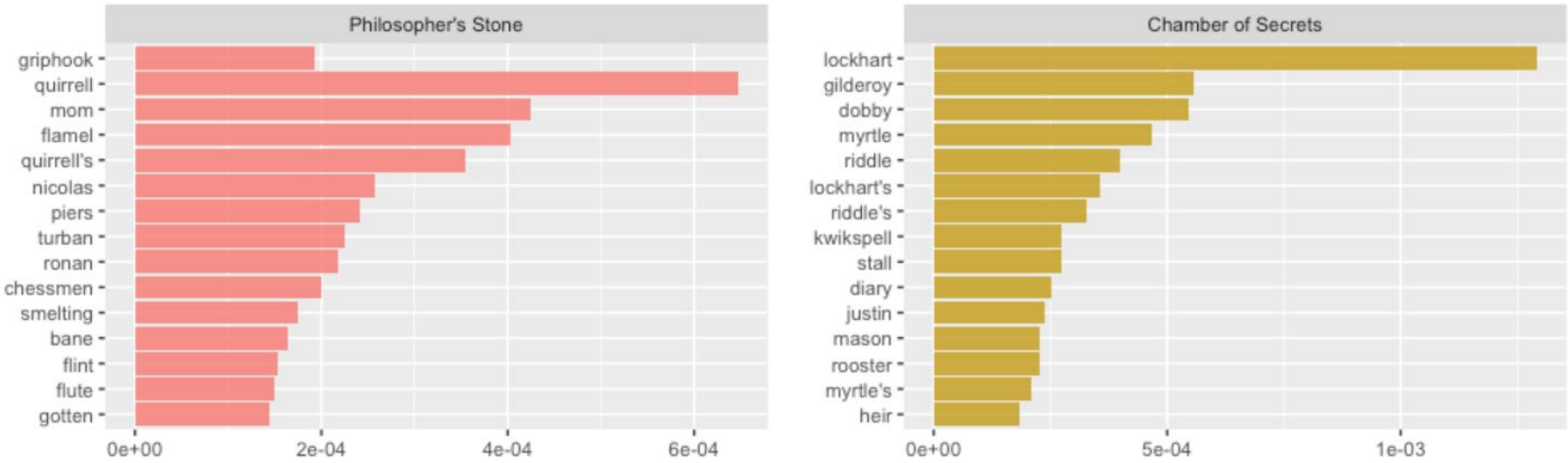
$$f(q,d2) \overset{\text{IDF(about)<IDF(presidential) for M>> 1}}{<<} f(q,d3)$$

Aber: $f(q,d3) << f(q,d5)$

Beispiel

Wörter mit dem höchsten TF-IDF

Beispiel Harry Potter: *Stein der Weisen* und *Kammer des Schreckens*



Wörter mit hohem TF-IDF bieten in diesem Beispiel spezifischen Kontext für die am häufigsten auftretenden Charaktere.
Gewöhnliche kontextfreie Wörter wie z.B. 'the', 'to' und 'and' haben hohe TF-Werte, aber ihre IDF und TF-IDF Werte sind 0.

Zusammenfassung

Verbessertes Vector Space Modell

- Vektor = TF-IDF Gewichtsvektor

Bessere Ergebnisse als einfachstes Vector Space Modell
Aber: Immer noch Probleme...

...Ausblick:

- TF Transformation
- Pivoted Length Normalization

Zusammenfassung: Vector Space Modell

- $Relevanz(q, d) = Similarity(q, d)$
- Query und Dokumente als Vektoren dargestellt
- Ranking Funktion als Heuristik
- Heuristik der Term-Gewichtungen
  - TF Gewichtung und Transformation
  - IDF Gewichtung
  - Normalisierung der Dokumentenlänge
- BM25 und Pivoted Normalisierung am effektivsten