Informationssysteme

Sommersemester 2016

Prof. Dr.-Ing. Sebastian Michel
TU Kaiserslautern

smichel@cs.uni-kl.de

Retrieval Methoden

- **Das Boolesche Modell**
- **TF*IDF**
- **Vektorraummodell**
- **Berücksichtigung von Vielfalt und Neuheit**
- **Dokumentenähnlichkeit, Shingling**
- **Latent-Semantic-Indexing (LSI)**

Boolesches Modell: Term-Dokument-Matrix (Incidence-Matrix)

	Froschk.	Hänsel& G.	Rapunzel	Rotkä.	7 Zwerge
Vater	1	1	0	0	1
Mutter	0	1	1	1	0
König/in	1	0	1	0	1
Zwerge	0	0	0	0	1
Königstochter	1	0	0	0	1
Wolf	0	0	0	1	0
Gold	0	0	1	0	1

Ein einfaches Boolesches Modell zur Suche

- Variablen beschreiben ob ein Wort in einem Dokument auftritt
- **Boolesche Operatoren AND**, **OR**, und **NOT**
- Anfragen können beliebig komplizierte (verschachtelte) Konstrukte aus diesen Operatoren sein.
- Z.B.
  - Haus **AND** Hexe **AND** Wald
  - Wolf **AND** Haus **AND NOT** Schweinchen
  - Frosch **OR** Gold

Das Ergebnis ist eine nicht geordnete Menge von Dokumenten, die diese Anfrage erfüllen.

Wieso ist das nicht gut?

Im Gegensatz zum Booleschen Modell sehen wir unten die **Häufigkeit** mit der die einzelnen Terme in den Dokumenten auftreten.

	Froschk.	Hänsel& G.	Rapunzel	Rotkä.	7 Zwerge
Vater	3	7	0	0	1
Mutter	0	2	1	2	0
König/in	9	0	5	0	19
Zwerge	0	0	0	0	10
Königstochter	6	0	0	0	1
Wolf	0	0	0	6	0
Gold	0	0	1	0	2

Vektorraummodell

Beobachtung

- Boolesches Modell hat keine (oder nur sehr einfache) Möglichkeit Resultate nach Güte zu Ordnen.
- Diese Einschränkung ist insbesondere bei großen Datenmengen (Google!) ein Problem.

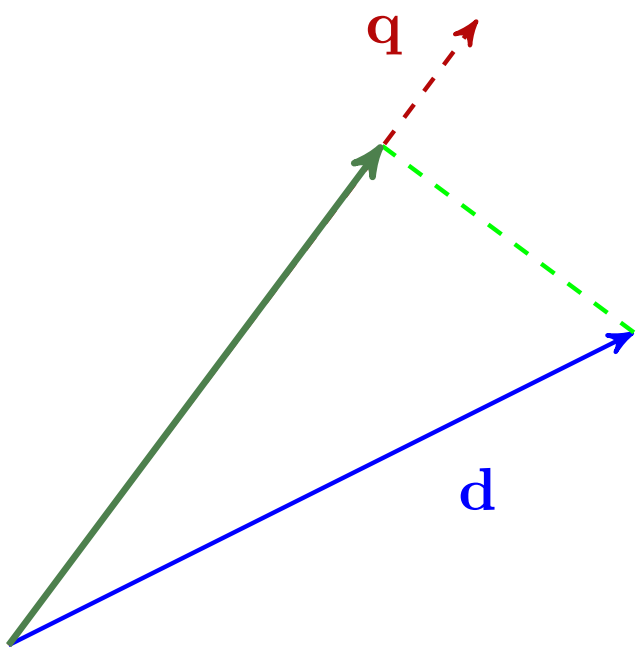
Vektorraummodell

- Im Vektorraummodell wird **jedes Dokument als v-dimensionaler Vektor** betrachtet, wobei $v = |V|$ die Anzahl an Worten ist (V=Vokabular), d.h. eine Dimension pro Wort.
- Beispiel aus den Daten von zuvor: $d_{7\ \text{Zwerge}} = \langle 1,0,19,10,1,0,2 \rangle$
- Ebenso wird die **Anfrage als Vektor** ausgedrückt, z.B. die Anfrage {Zwerge, Gold} entspricht dem Vektor $q = \langle 0,0,0,1,0,0,1 \rangle$

Vektorraummodell: Kosinus-Ähnlichkeit

Die **Kosinus-Ähnlichkeit** zwischen zwei Vektoren $\mathbf{q}$ und $\mathbf{d}$ ist der **Kosinus des Winkels zwischen den Vektoren**:

$$
\begin{aligned}
sim(\mathbf{q},\mathbf{d}) &= \frac{\mathbf{q} \cdot \mathbf{d}}{||\mathbf{q}||\,||\mathbf{d}||} \\
&= \frac{\sum_{i=1}^{|V|} \mathbf{q}_i \mathbf{d}_i}{\sqrt{\sum_{i=1}^{|V|} \mathbf{q}_i^2}\sqrt{\sum_{i=1}^{|V|} \mathbf{d}_i^2}} \\
&= \frac{\mathbf{q}}{||\mathbf{q}||}\frac{\mathbf{d}}{||\mathbf{d}||}
\end{aligned}
$$



Mit $\mathbf{d} = \langle 1,0,19,10,1,0,2 \rangle$ und $\mathbf{q} = \langle 0,0,0,1,0,0,1 \rangle$ haben wir $\mathbf{q} \cdot \mathbf{d} = 12$, $||\mathbf{d}|| = \sqrt{467}$ und $||\mathbf{q}|| = \sqrt{2}$ und erhalten $sim(\mathbf{q},\mathbf{d}) = 0.392652$.

TF*IDF

- Wie können die einzelnen Einträge $\mathbf{q}_t$ und $\mathbf{d}_t$ für einen Term t berechnet werden?
- Im Booleschen-Modell hätten wir Einträge aus $\{0,1\}$
- Im Beispiel zuvor haben wir die Anzahl des Auftretens eines Terms benutzt, die sogenannte Termfrequenz.
- **Termfrequenz** $tf_{t,d}$ als die Anzahl des Auftretens von Term t in Dokument d
- **Termfrequenz sollte gewichtet werden und nicht direkt benutzt werden. Wieso?**

TF*IDF

- Wie können die einzelnen Einträge $\mathbf{q}_t$ und $\mathbf{d}_t$ für einen Term t berechnet werden?
- Im Booleschen-Modell hätten wir Einträge aus $\{0,1\}$
- Im Beispiel zuvor haben wir die Anzahl des Auftretens eines Terms benutzt, die sogenannte Termfrequenz.
- **Termfrequenz** $tf_{t,d}$ als die Anzahl des Auftretens von Term t in Dokument d
- **Termfrequenz sollte gewichtet werden und nicht direkt benutzt werden. Wieso?**
- **Dokumentfrequenz** df_t als die Anzahl der Dokumente, in denen Term t auftritt
- **Inverse Dokumentfrequenz** idf_t als

$$idf_t = \frac{|D|}{df_t}$$

wobei $|D|$ die Anzahl der Dokumente in der Kollektion sind.

TF*IDF

- Das $tf.idf$ **Gewicht** von Term t in Dokument d ist dann definiert als

$$tf.idf_{t,d} = tf_{t,d} \times idf_t$$

- Beobachtungen: $tf.idf_{t,d}$ ist ....
  - **größer falls t oft in d auftritt**
  - **kleiner falls t nicht oft in d auftritt und/oder t in vielen Dokumenten auftritt**
- Nun können wir im **Vektorraummodell** die einzelnen Komponenten der Vektoren bestimmen:

$$\mathbf{d}_t = tf.idf_{t,d} \qquad \mathbf{q}_t = tf.idf_{t,q}$$

- Eine einfachere Möglichkeit die Güte von Dokument d zu berechnen ist durch eine einfache **Summe der einzelnen $tf.idf$ Werte**:

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

Dämpfung, Längennormalisierung etc.

Es gibt verschiedene Variationen des klassischen tf.idf Maßes.

Logarithmische Dämpfung der inversen Dokumentfrequenz

$$idf_t = log \frac{|D|}{df_t}$$

verhindert, dass zu sehr exotische Terme zu viel Gewicht erhalten.

Sublineare Skalierung der Termfrequenz

$$wf_{t,d} = \begin{cases} 1 + log\ tf_{t,d} & \text{falls } tf_{t,d} > 0 \\ 0 & \text{sonst} \end{cases}$$

Längennormalisierung und max-tf Normalisierung

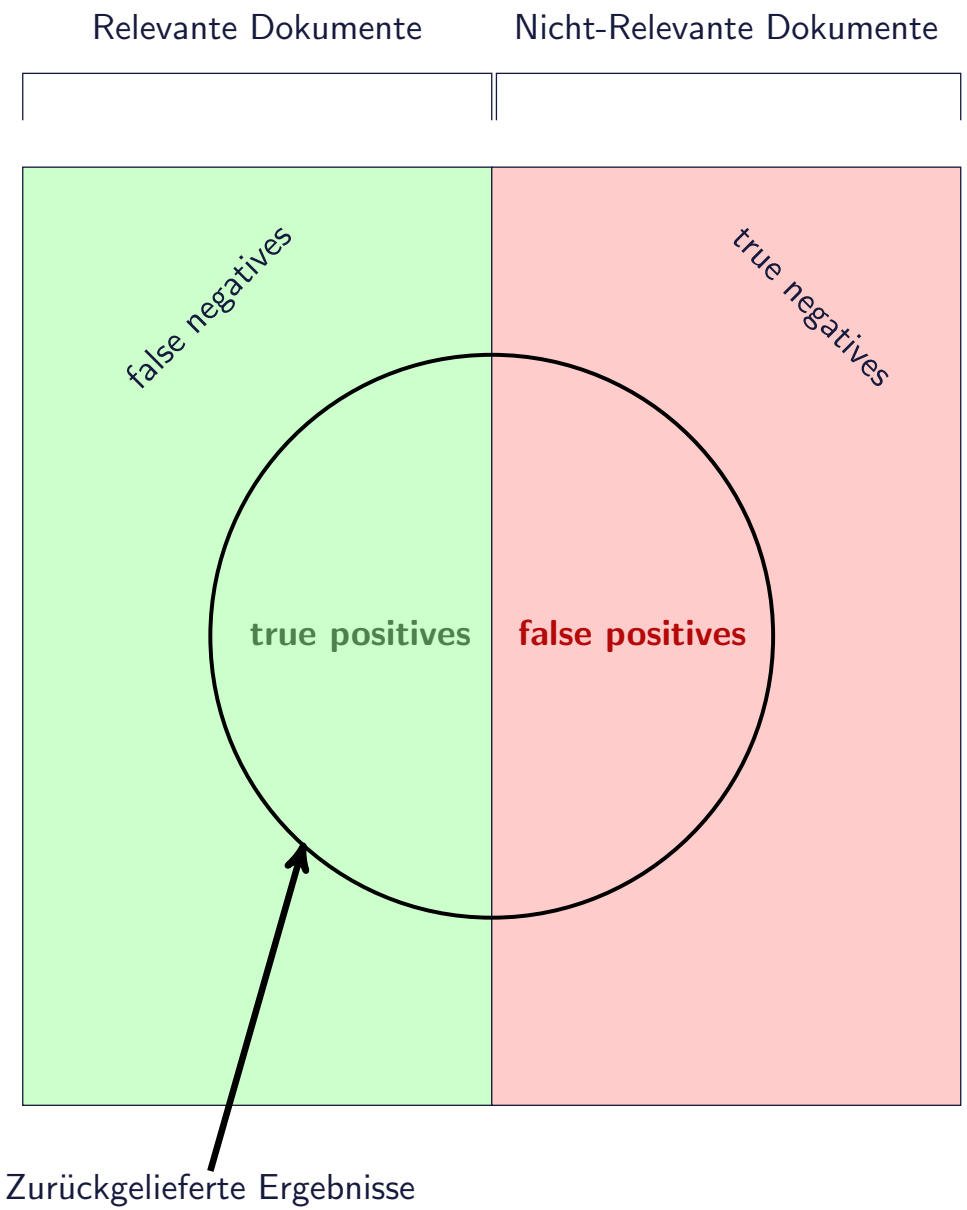
$$rtf_{t,d} = \frac{tf_{t,d}}{\sum_{v \in \mathbf{d}} tf_{v,d}} \qquad ntf_{t,d} = \frac{tf_{t,d}}{max_{v \in \mathbf{d}} tf_{v,d}}$$

verhindern, dass sehr lange Dokumente zu sehr favorisiert werden.

Bewertung der Resultatgüte

- Es gibt also verschiedene Möglichkeiten zu berechnen wie gut bzw. wie schlecht ein Dokument als Ergebnis einer Anfrage geeignet ist (und es gibt noch zahlreiche andere, die hier nicht betrachtet wurden).

- Wie kann nun gesagt werden, dass eine Möglichkeit besser als eine andere ist?


Relevante Dokumente
Nicht-Relevante Dokumente
false negatives
true negatives
true positives
false positives
Zurückgelieferte Ergebnisse

Klassifizierung der Ergebnisse

Ein System gibt eine Menge von Dokumenten als Antwort zu einer Anfrage zurück. Wir unterscheiden zwischen den folgenden vier Arten von Dokumenten:

	relevant	**nicht relevant**
zurückgegeben	true positives (tp)	false positives (fp)
nicht zurückgegeben	false negatives (fn)	true negatives (tn)

Was sollte ein gutes Informationssystem erfüllen?

- Viele (alle) der zurückgelieferten Dokumente sollten relevant sein, d.h. viele true positives und wenig (keine) false positives.
- Viele (alle) der relevanten Dokumente sollten zurückgeliefert werden, d.h. wenig (keine) false negatives.

Präzision (Precision) und Ausbeute (Recall)

Precision P ist der Anteil der relevanten Dokumente in den zurückgegebenen Dokumenten

$$P = \frac{tp}{tp + fp}$$

Recall R ist der Anteil der relevanten zurückgegebenen Dokumente an allen relevanten Dokumenten

$$R = \frac{tp}{tp + fn}$$

Was ist wichtiger, Precision oder Recall?

F-Measure

F-measure kombiniert Precision und Recall

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

wobei mit dem Parameter β zwischen Precision und Recall gewichtet werden kann.

- $\beta = 1$ ist balanciert
- $\beta < 1$ höherer Einfluss von Precision
- $\beta > 1$ höherer Einfluss von Recall

Average-Precision und Mean-Average-Precision

- Precision, Recall und auch das F-Maß ignorieren die Ordnung der Resultate.
- **Average Precision (AP)**
  - Sei $\{d_1, \ldots, d_m\}$ die Menge der relevanten Dokumente der Anfrage q
  - Sei R_k die Menge der geordneten Ergebnisse für Anfrage von oben bis zum relevanten Ergebnis d_k.

$$AP(q) = \frac{1}{m} \sum_{d_k \in \{d_1, \ldots, d_m\}} \text{Precision}(R_k)$$

- Für eine Menge Q von Anfrage kann **Mean-Average-Precision (MAP)** berechnet werden durch

$$MAP(Q) = \frac{1}{|Q|} \sum_{q \in Q} AP(q)$$

Average Precision: Beispiel

Rang	Relevant?
1	ja
2	ja
3	nein
4	nein
5	ja
6	nein
7	nein
8	nein
9	nein
10	ja
11	nein
12	nein
13	ja
14	nein
15	nein

d_i sei das Dokument auf Rang i.

Wir haben also die Menge der relevanten Dokumente $\{d_1, d_2, d_5, d_{10}\}$

- $R_1 = \{d_1\}$ und Precision$(R_1) = 1/1 = 1$
- $R_2 = \{d_1, d_2\}$ und Precision$(R_2) = 2/2 = 1$
- $R_5 = \{d_1, d_2, d_3, d_4, d_5\}$ und Precision$(R_5) = 3/5 = 0.6$
- $R_{10} = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}\}$ und Precision$(R_{10}) = 4/10 = 0.4$
- $R_{13} = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}, d_{11}, d_{12}, d_{13}\}$ und Precision$(R_{13}) = 5/13 \approx 0.384$

Wir haben also eine Average-Precision von $AP = (1+1+3/5+4/10+5/13)/5 = 44/65 \approx 0.677$

Precision@k

- Es ist unrealistisch anzunehmen, dass Anwender tatsächlich die gesamte Ergebnisliste anschauen.
- Vielmehr werden Anwender vielleicht nur die top-10 oder top-20 Ergebnisse betrachten, wie z.B. in der Websuche üblich.
- **Precision@k (=P@k) ist die Precision, die in den ersten k (also top-k) Ergebnissen erreicht wird.**

Precision@k: Beispiel

Rang	Relevant?
1	ja
2	ja
3	nein
4	nein
5	ja
6	nein
7	nein
8	nein
9	nein
10	ja
11	nein
12	nein
13	ja
14	nein
15	nein

- Precision@1 $= 1/1 = 1$
- Precision@5 $= 3/5 = 0.6$
- Precision@10 $= 4/10 = 0.4$
- Precision@15$= 5/15 = 1/3 = 0.\bar{3}$

Vielfalt und Neuheit

- Die **betrachteten Methoden** (Boolesches-Modell und tf.idf) gehen davon aus, dass die **Relevanz eines Dokuments unabhängig von der Relevanz anderer Dokumente** ist.
- **In der Realität ist diese Annahme allerdings problematisch**, da zurückgegebene Ergebnisse gleiche oder sehr ähnliche Informationen enthalten können, sogar Duplikate sein können.
- **Idee:** Wir versuchen, dass jedes weitere Ergebnis möglichst verschieden ist zu den zuvor gelesenen Dokumenten (angenommen Dokumente sind sortiert nach Güte und der Anwender liest von oben nach unten).

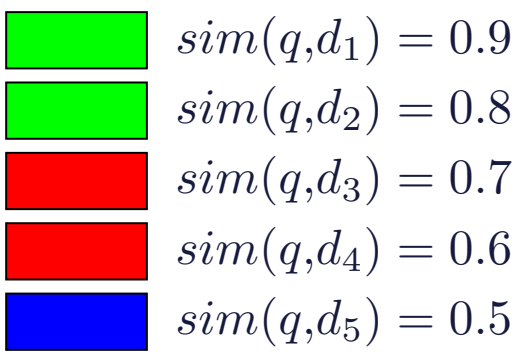
Maximale Marginale Relevanz (MMR)

Das nächste zurückzugebende Dokument d_i soll relevant zur Anfrage q sein, aber auch verschieden zu den bislang zurückgegebenen Dokumenten $d_1, \ldots d_{i-1}$

$$argmax_{d_i \in D} \left(\lambda \; sim(q,d_i) - (1-\lambda) \; max_{d_j : 1 \leq j < i} \; sim(d_i,d_j) \right)$$

mit Tuning-Parameter λ und Ähnlichkeitsmaß sim.

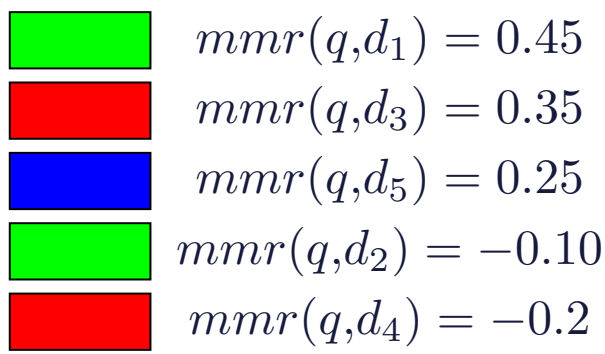
Ergebnisliste nach Güte



$$sim(d,d') = \begin{cases} 1.0 & \text{gleiche Farbe} \\ 0.0 & \text{sonst} \end{cases}$$

$\lambda = 0.5$

Finale Ergebnisse



Ähnlichkeitsmaß Jaccard-Koeffizient

Bekanntes Maß zur **Berechnung von Ähnlichkeiten zweier Mengen**.

Gegeben zwei Mengen A und B von Elementen. Der Jaccard-Koeffizient $J(A,B)$ dieser Mengen ist

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Zur Berechnung der **Ähnlichkeit von Text-Dokumenten** zwecks Duplikaterkennung werden in der Regel nicht die einzelnen Worte, sondern Wortsequenzen, sogenannte **k-grams** oder **k-shingles**, betrachtet.

Wieso ist dies i.d.R. aussagekräftiger?

Shingling

- Shingling berechnet für ein Dokument d eine **Menge von Sequenzen jeweils bestehend k Worten**, die in dem Dokument **nebeneinander** auftreten.
- Solch eine Sequenz wird **k-shingle** oder auch **k-gram** genannt.
- **Beispiel:** Für ein Dokument $[a\ b\ b\ c]$, wobei a, b und c Worte sind, haben wir die folgende Menge an 2-shingles, die dieses Dokument beschreiben: $\{ab, bb, bc\}$.
- Die k-shingles eines Dokuments d_i werden mit $S(d_i)$ bezeichnet.

Shingling

- Shingling berechnet für ein Dokument d eine **Menge von Sequenzen jeweils bestehend k Worten**, die in dem Dokument **nebeneinander** auftreten.
- Solch eine Sequenz wird **k-shingle** oder auch **k-gram** genannt.
- **Beispiel:** Für ein Dokument $[a\ b\ b\ c]$, wobei a, b und c Worte sind, haben wir die folgende Menge an 2-shingles, die dieses Dokument beschreiben: $\{ab, bb, bc\}$.
- Die k-shingles eines Dokuments d_i werden mit $S(d_i)$ bezeichnet.

Um die **Ähnlichkeit zweier Dokumente** d_1 und d_2 zu berechnen kann nun der Jaccard-Koeffizient zwischen den Mengen der Shingles von d_1 und d_2 berechnet werden:

$$sim(d_1, d_2) = \frac{|S(d_1) \cap S(d_2)|}{|S(d_1) \cup S(d_2)|}$$

Latent-Semantic-Indexing (LSI)

Beobachtung:

Bislang betrachtete Ansätze (z.B. TF*IDF) können nicht mit **Synonymie** (z.B. Auto und PKW) und **Polysemie** (z.B. Java) umgehen.

- Das gemeinsame Auftreten von Worten kann hier helfen, z.B.
  - Auto und PKW treten in Dokumenten über Abgase oder Parkplatz, etc. auf
  - Java tritt gemeinsam mit Worten wie *Klasse* oder *Methode* auf, aber auch mit Worten wie *Kaffee* und *Bohnen*.

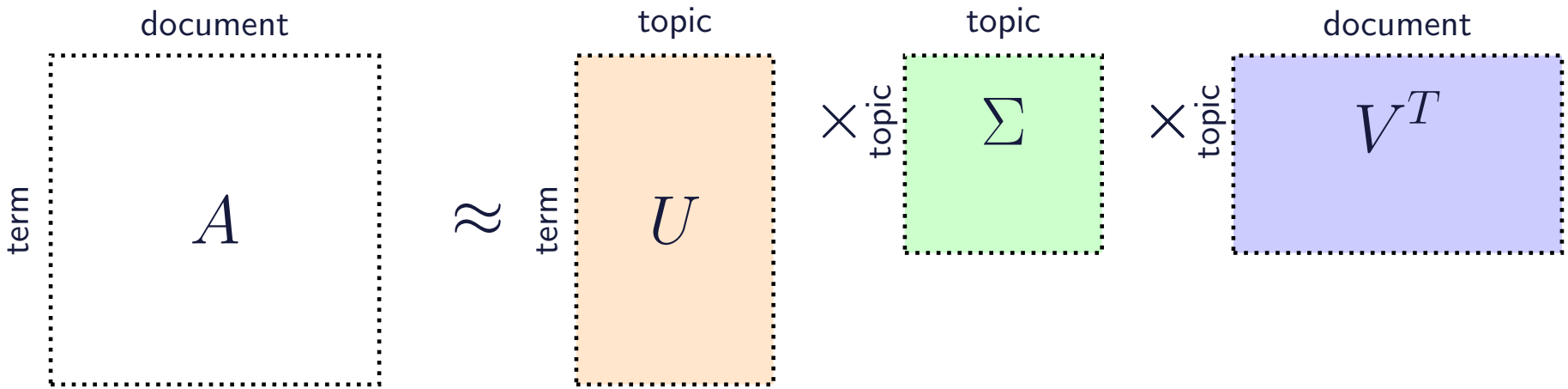
Latent Topic Models

machen die Annahme, dass Dokumente aus eine kleinen Anzahl von versteckten (latent) Konzepten (topics) bestehen.
Wir betrachten hier **Latent Semantic Indexing (LSI)** [Deerwester et al. '90], aber es gibt weitere Ansätze. `http://lsa.colorado.edu/papers/JASIS.lsi.90.pdf`

Latent-Semantic-Indexing (LSI)

Idee

Betrachte **Singulärwertzerlegung** (SVD, Englisch: Singular Value Decomposition) der $m \times n$ **Term-Dokument-Matrix** A.



- U bildet Terme auf Topics ab.
- V bildet Dokumente auf Topics ab

Wiederholung: Vektoroperationen

- $\mathbf{v}^T$ transponiert einen Zeilenvektor in einen Spaltenvektor und umgekehrt.
- Für Vektoren $\mathbf{v}$,$\mathbf{w} \in R^n$, $\mathbf{v}+\mathbf{w}$ ist ein Vektor mit $(\mathbf{v}+\mathbf{w})_i = \mathbf{v}_i + \mathbf{w}_i$
- Für Vektor $\mathbf{v}$ und Skalar α, $(\alpha\mathbf{v})_i = \alpha\mathbf{v}_i$
- Das Skalarprodukt zweier Vektoren $\mathbf{v}, \mathbf{w} \in R^n$ ist

$$\mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^{n} v_i w_i$$

Wiederholung: Matrixoperationen

- Die transponierte Matrix $\mathbf{A}^T$ hat die Zeilen von $\mathbf{A}$ als Spalten
- Sind $\mathbf{A}$ und $\mathbf{B}$ zwei $n \times m$ Matrizen, dann ist $\mathbf{A} + \mathbf{B}$ eine $n \times m$ Matrix mit $(\mathbf{A} + \mathbf{B})_{ij} = a_{ij} + b_{ij}$
- Ist $\mathbf{A}$ eine $n \times k$ und $\mathbf{B}$ eine $k \times m$ Matrix, dann ist $\mathbf{AB}$ eine $n \times m$ Matrix mit

$$(\mathbf{AB})_{ij} = \sum_{l=1}^{k} a_{il} b_{lj}$$

Die "innere" Dimension (k) muss übereinstimmen

SVD Beispiel

$A =$

	d_1	d_2	d_3	d_4	d_5	d_6
Schiff	1	0	1	0	0	0
Boot	0	1	0	0	0	0
Ozean	1	1	0	0	0	0
Reise	1	0	0	1	1	0
Ausflug	0	0	0	1	0	1

$U =$

	1	2	3	4	5
Schiff	0.44	-0.30	-0.57	0.58	-0.25
Boot	0.13	-0.33	0.59	0.00	-0.73
Ozean	0.48	-0.51	0.37	0.00	0.61
Reise	0.70	0.35	-0.15	-0.58	-0.16
Ausflug	0.26	0.65	0.41	0.58	0.09

$V^T =$

	1	2	3	4	5	6
1	0.75	0.28	0.20	0.45	0.33	0.12
2	-0.29	-0.53	-0.19	0.63	0.22	0.41
3	-0.28	0.75	-0.45	0.20	-0.12	0.33
4	0.00	0.00	0.58	0.00	-0.58	0.58
5	0.53	-0.29	-0.63	-0.19	-0.41	0.22

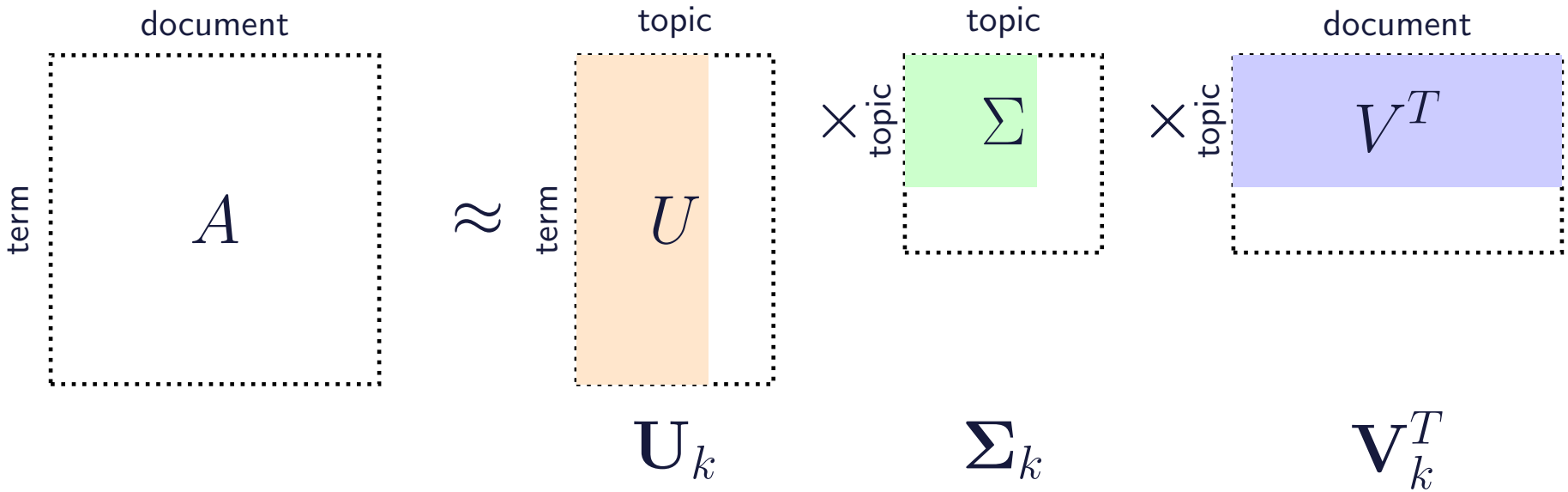
Die Singulärwerte von Σ auf der Diagonalen sind 2.16, 1.59, 1.28, 1.00, 0.39

Rang-k Approximation

- Man möchte $\mathbf{A}$ gar nicht korrekt wiederherstellen
- Sondern approximieren
- Durch Auswahl der wichtigsten (latenten) Topics
- **In Spalten von U stehen Topics beschrieben durch Terme.**
  Interpretation: Alle gleiche Vorzeichen $\rightarrow$ treten gemeinsam auf.
  Mischung aus +/- $\rightarrow$ Terme mit positivem Gewicht zusammen, aber ohne Terme mit neg. Gewicht.

- Wir wählen die k **größten Singulärwerte aus Σ aus**.
- D.h. wir setzen die übrigen Singulärwerte auf 0
- Und entsprechend die dazugehörigen Einträge in U und V^T.

Latent-Semantic-Indexing (LSI)



- $\mathbf{U}_k, \mathbf{V}_k^T, \mathbf{\Sigma}_k$ enthalten die ersten k Singularwerte und Werte
- **Wie viele Singulärwerte sollten erhalten bleiben?** Faustregel*: so viele, dass mindestens 90% der Summe der Quadrate der Singulärwerte erhalten bleibt.

*) `http://infolab.stanford.edu/~ullman/mmds/ch11.pdf`

Beispiel: Approximation für $k = 2$

für Matrix A =

	d_1	d_2	d_3	d_4	d_5	d_6
Schiff	1	0	1	0	0	0
Boot	0	1	0	0	0	0
Ozean	1	1	0	0	0	0
Reise	1	0	0	1	1	0
Ausflug	0	0	0	1	0	1

Zur Erinnerung:

- U_k **Term-Topic-Matrix**
- V_k^T **ist Topic-Dokument-Matrix**

$$\mathbf{U}_2 = \begin{pmatrix} 0.44 & -0.30 \\ 0.13 & -0.33 \\ 0.48 & -0.51 \\ 0.70 & 0.35 \\ 0.26 & 0.65 \end{pmatrix}$$

$$\mathbf{\Sigma}_2 = \begin{pmatrix} 2.16 & 0.00 \\ 0.00 & 1.59 \end{pmatrix}$$

$$\mathbf{V}_2^T = \begin{pmatrix} 0.75 & 0.28 & 0.20 & 0.45 & 0.33 & 0.12 \\ -0.29 & -0.53 & -0.19 & 0.63 & 0.22 & 0.41 \end{pmatrix}$$

Weiteres Beispiel

$n = 5$ **Dokumente**

d_1: how to bake bread without recipes

d_2: the classic art of viennese pastry

d_3: numerical recipes: the art of scientific computing

d_4: breads, pastries, pies and cakes: quantity baking recipes

d_5: pastry: a book of the best french recipes

m=6 Terme

t_1: bak(e,ing)

t_2: recipe(s)

t_3: bread

t_4: cake

t_5: pastr(y,ies)

t_6: pie

$$A_{raw} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Weiteres Beispiel

$n = 5$ **Dokumente**

d_1: how to bake bread without recipes

d_2: the classic art of viennese pastry

d_3: numerical recipes: the art of scientific computing

d_4: breads, pastries, pies and cakes: quantity baking recipes

d_5: pastry: a book of the best french recipes

m=6 Terme

t_1: bak(e,ing)

t_2: recipe(s)

t_3: bread

t_4: cake

t_5: pastr(y,ies)

t_6: pie

$$A_{\text{normalisiert}} = \begin{pmatrix} 0.5774 & 0 & 0 & 0.4082 & 0 \\ 0.5774 & 0 & 1 & 0.4082 & 0.7071 \\ 0.5774 & 0 & 0 & 0.4082 & 0 \\ 0 & 0 & 0 & 0.4082 & 0 \\ 0 & 1 & 0 & 0.4082 & 0.7071 \\ 0 & 0 & 0 & 0.4082 & 0 \end{pmatrix}$$

http://www.cs.colorado.edu/~jessup/SUBPAGES/PS/matrices.pdf

Rang 3 Approximation

$$\mathbf{A}_3 = \mathbf{U}_3 \times \mathbf{\Sigma}_3 \times \mathbf{V}_3^T = \begin{pmatrix} -0.2670 & 0.2567 & -0.5308 \\ -0.7479 & 0.3980 & 0.5249 \\ -0.2670 & 0.2567 & -0.5308 \\ -0.1182 & 0.0127 & -0.2774 \\ -0.5197 & -0.8423 & -0.0839 \\ -0.1182 & 0.0127 & -0.2774 \end{pmatrix}$$

$$\times \begin{pmatrix} 1.6950 & 0.00000 & 0.0000 \\ 0.0000 & 1.1158 & 0.0000 \\ 0.0000 & 0.0000 & 0.8403 \end{pmatrix}$$

$$\times \begin{pmatrix} -0.4367 & -0.3066 & -0.4413 & -0.4908 & -0.5288 \\ 0.4717 & -0.7549 & 0.3567 & 0.0346 & -0.2815 \\ -0.3688 & -0.0998 & 0.6247 & -0.5710 & 0.3711 \end{pmatrix}$$

Rang 3 Approximation

Die approximierte Matrix A_3 brauchen wir gar nicht zu betrachten. Sie sieht aber wie folgt aus:

$$\mathbf{A}_3 = \begin{pmatrix} 0.4972 & -0.0330 & 0.0232 & 0.4867 & -0.0069 \\ 0.6004 & 0.0094 & 0.9933 & 0.3857 & 0.7091 \\ 0.4972 & -0.0330 & 0.0232 & 0.4867 & -0.0069 \\ 0.1801 & 0.0740 & -0.0522 & 0.2319 & 0.0155 \\ -0.0326 & 0.9866 & 0.0094 & 0.4401 & 0.7043 \\ 0.1801 & 0.0740 & -0.0522 & 0.2319 & 0.0155 \end{pmatrix}$$

Operationen im Topic-Raum

- Wir können eine **Anfrage aus dem m-dimensionalen Term-Raum in den k-dimensionalen Topic-Raum abbilden** durch

$$q \rightarrow \mathbf{U}_k^T q = q'$$

Operationen im Topic-Raum (2)

- Die Eignung (Score) der Dokumente wird dann im Topic-Raum berechnet, in dem q' **mit den Spalten der Matrix $\mathbf{V}_k^T$ verglichen** wird, anhand der Cosinus-Ähnlichkeit oder des Skalarprodukts.

Anfragen in LSI

Anfrage: baking bread

- $q = (1, 0, 1, 0, 0, 0)^T$

  Nun Abbildung von q in den Topic-Raum:

- $q' = \mathbf{U}_3^T q = (0.5340, -0.5134, 1.0616)^T$

Skalarprodukt als Ähnlichkeitsmaß der Vektoren im Topic-Raum:

- $sim(q', d_1') \approx 0.86$
- $sim(q', d_2') \approx -0.12$
- $sim(q', d_3') \approx -0.24$

Zur Erinnerung:

t_1: bak(e,ing)
t_2: recipe(s)
t_3: bread
t_4: cake
t_5: pastr(y,ies)
t_6: pie

Kurzeinführung in R (https://www.r-project.org/)

```
> x <- 10
```
Eine einfache Zuweisung (Enter drücken)

```
> x
 [1]  10
```
Enter drücken und ....
... Wert von x wird ausgegeben

```
>v <- c(1,2,3,4,5)
>v
 [1] 1 2 3 4 5
```
Definition eines Vektors

```
>vv <- 2*v
>vv
 [1] 1 4 6 8 10
```
Vektor multipliziert mit Skalar

Kurzeinführung in R

Matrizen....

```
>x <- c(1,2,3,4)
>y <- c(5,6,7,8)
>mat1 <- cbind(x,y)
>mat1
      x y
 [1,] 1 5
 [2,] 2 6
 [3,] 3 7
 [4,] 4 8
 mat2 <- rbind(x,y)
mat2
  [,1] [,2] [,3] [,4]
x    1    2    3    4
y    5    6    7    8
```

Kurzeinführung in R

```
>mat2[1,]
```
Wählt erste Zeile aus `[1] 1 2 3 4`

```
>mat2[1,1]
```
Wählt Eintrag aus

```
>t(mat2)
```
Berechnet Transponierte

```
     x y
[1,] 1 5
[2,] 2 6
[3,] 3 7
[4,] 4 8
```

```
>t(mat1)
  [,1] [,2] [,3] [,4]
x    1    2    3    4
y    5    6    7    8
```

Kurzeinführung in R

```
>2*mat1
     x  y
[1,] 2 10
[2,] 4 12
[3,] 6 14
[4,] 8 16
```

```
>mat1 %*% mat2
     [,1] [,2] [,3] [,4]
[1,]   26   32   38   44
[2,]   32   40   48   56
[3,]   38   48   58   68
[4,]   44   56   68   80
```

Multiplikation zweier Matrizen

LSI in R

```
A <- matrix(c(0.5774,0,0,0.4082,0,0.5774,0,1,0.4082,
   0.7071,0.5774,0,0,0.4082,0, 0,0,0,0.4082,
   0,0,1,0,0.4082,0.7071,0,0,0,0.4082,0),
   nrow=6, ncol=5, byrow=TRUE)
A_svd = svd(A)
```

```
> A_svd
$d
[1] 1.694953e+00 1.115800e+00 8.402774e-01 4.194943e-01 4.371327e-17

$u
           [,1]        [,2]       [,3]        [,4]          [,5]
[1,] -0.2669599  0.25673561 -0.5308244  0.28468355 -7.071068e-01
[2,] -0.7479234  0.39803133  0.5249084 -0.08156498  1.110223e-16
[3,] -0.2669599  0.25673561 -0.5308244  0.28468355  7.071068e-01
[4,] -0.1182043  0.01265192 -0.2773859 -0.63947222 -1.526557e-16
[5,] -0.5197412 -0.84227346 -0.0838825  0.11579410  1.942890e-16
[6,] -0.1182043  0.01265192 -0.2773859 -0.63947222 -3.191891e-16
$v
           [,1]        [,2]        [,3]        [,4]       [,5]
[1,] -0.4366707  0.47168093 -0.36882335  0.67142025  0.0000000
[2,] -0.3066405 -0.75486063 -0.09982715  0.27603260  0.4999976
[3,] -0.4412650  0.35672284  0.62468462 -0.19443645  0.4999976
[4,] -0.4908152  0.03458356 -0.57099734 -0.65716544  0.0000000
[5,] -0.5288440 -0.28152323  0.37112672  0.05769664 -0.7071102
```

LSI in R: Rang-k Approximation

```
u3 <- A_svd$u[,1:3]
s3 <- diag(A_svd$d)[1:3,1:3]
vt3 <- t(A_svd$v)[1:3,]
A3 <- u3 %*% s3 %*% vt3
```