Universität Potsdam
Institut für Informatik
Lehrstuhl Maschinelles Lernen

Zusammenfassung

Tobias Scheffer
Paul Prasse
Michael Großhans
Uwe Dick

Statistische Sprachmodelle

- Wie wahrscheinlich ist ein bestimmter Satz in einem Kontext:
  - P(„ich pflücke Beeren“) ist vielleicht 0.0001,
  - P(„ich pflücke Bären“) ist vielleicht 10^{-25}.

$$P(w_1, \ldots, w_T)$$

- Statistische Sprachmodelle dieser Vorlesung:
  - N-Gramme,
  - Probabilistische kontextfreie Grammatiken.

N-Gramm-Modelle

- Markov-Annahme (N-1). Ordnung (bsp: 2. Ordnung):
  - P(„Wald“ | „ich pflücke Beeren im“) = P(„Wald“ | „Beeren im“).
- Modell speichert Wahrscheinlichkeiten für Wortfolgen der Länge maximal N.
- N=1: „Unigramm“,
- N=2: „Bigramm“,
- N=3: „Trigramm“.

N-Gramm-Modelle
Erweiterung: Interpolation

- Dem N-Gramm-Sprachmodell liegt ein generatives Prozessmodell zugrunde.
  - Wörter werden iterativ „ausgewürfelt“.
  - Jedes Wort w_i wird nach der Wahrscheinlichkeit $P(w_i \mid w_{i-1},\ldots, w_{i-n+1})$ ausgewürfelt.

- Generatives Modell für interpoliertes N-Gramm:
  - Für jedes Wort w_i wird zunächst Markov-Grad n zwischen 1 und N nach $p(n)$ gezogen.
  - Wort w_i wird dann nach $P(w_i \mid w_{i-1},\ldots, w_{i-n+1})$ gezogen.

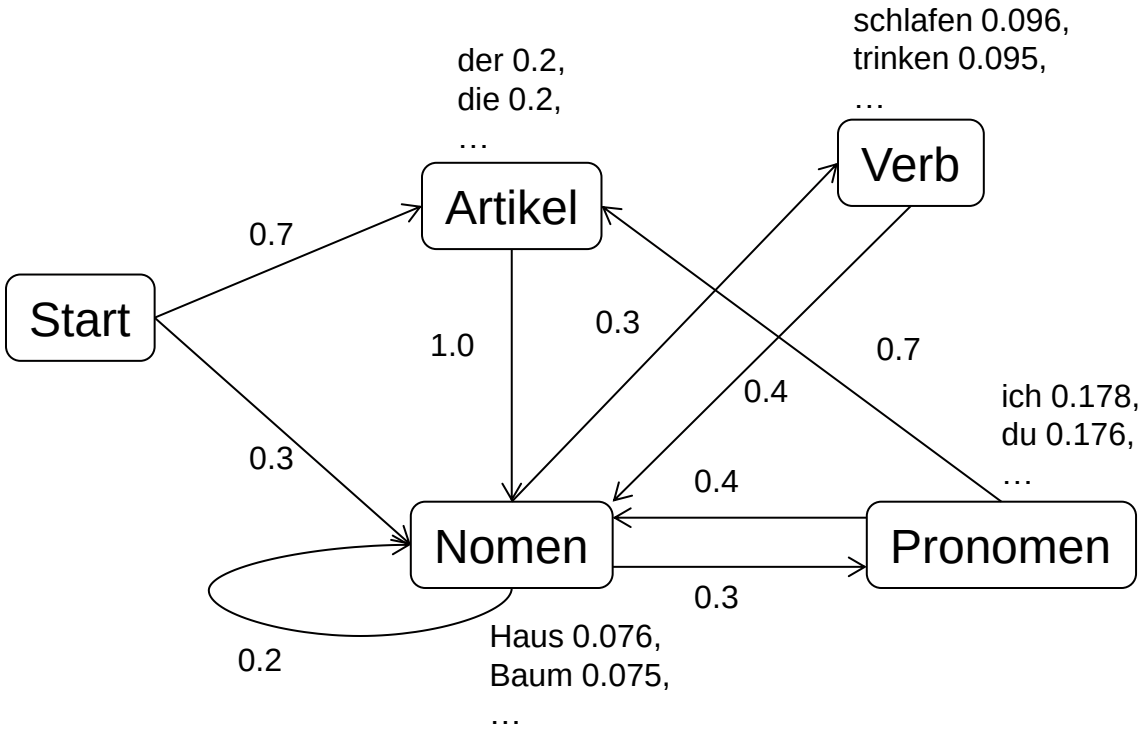
Lernen mit EM-Algorithmus

NLP- (Natural Language Processing-) Pipeline

1. Tokenisierung,
2. Wortarterkennung,
3. Eigennamenerkennung,
4. Parsing,
5. Informationsextraktion.

Markov-Modelle
Wortarterkennung

- Modellierung: Ein Zustand pro POS-Tag.
- Trainingsdaten: „Getaggtes“ Korpus. Zustände sind sichtbar.
- Anwendung: Zustände sind noch nicht sichtbar.



Probabilistische kontextfreie Grammatiken
Parsing

- Probabilistic context-free grammar (PCFG): Statistisches Sprachmodell.

S → NP VP	1.0	NP → NP PP	0.4
PP → P NP	1.0	NP → *astronomers*	0.1
VP → V NP	0.7	NP → *ears*	0.18
VP → VP PP	0.3	NP → *saw*	0.04
P → *with*	1.0	NP → *stars*	0.18
V → *saw*	1.0	NP → *telescopes*	0.1

Probabilistische kontextfreie Grammatiken
Parsing

- Drei Fragen:
  - Was ist die Wahrscheinlichkeit eines Satzes w_{1T} gegeben eine Grammatik?

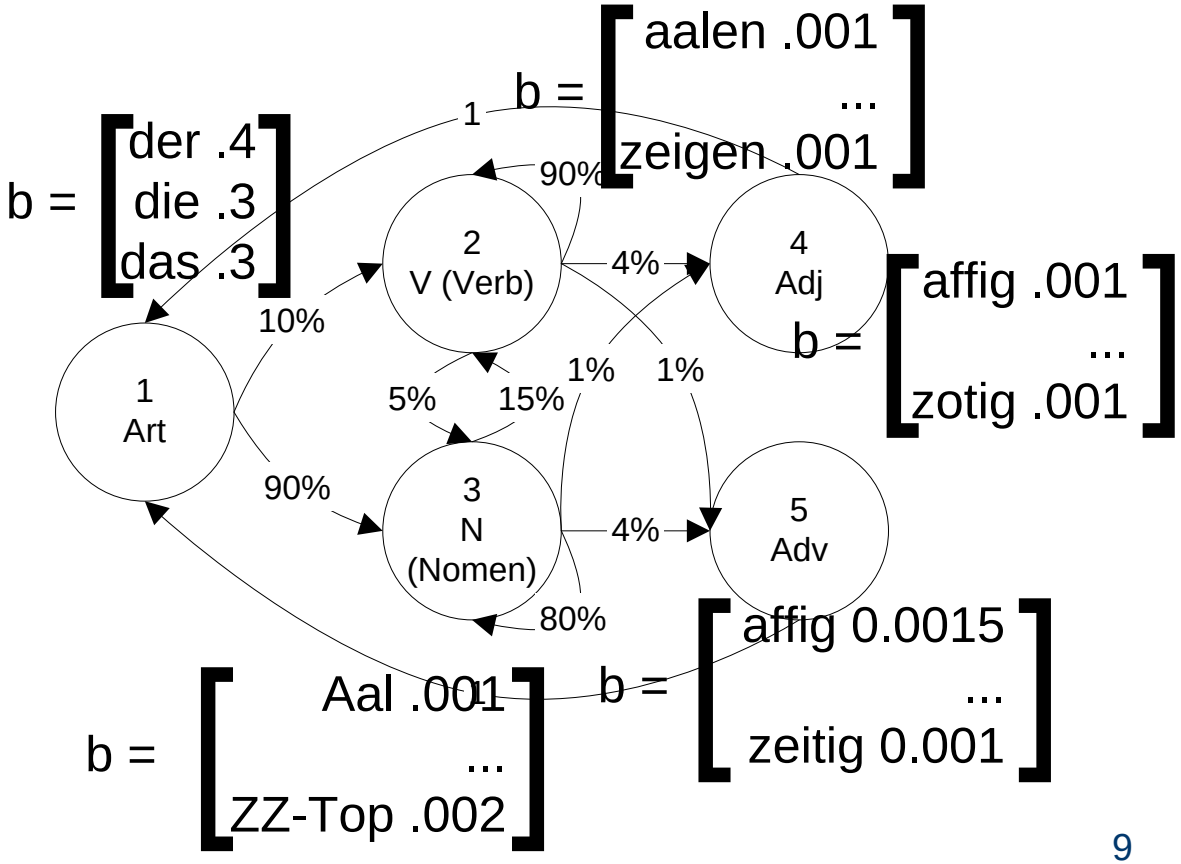
    Inside-Algorithmus
  - Was ist der wahrscheinlichste Pars-Baum gegeben Satz und Grammatik?

    „Viterbi-Algorithmus
  - Welche Wahrscheinlichkeiten für die Grammatikregeln können wir aus einem gegebenen Trainingskorpus lernen?

    EM-Algorithmus

Hidden-Markov-Modell

- Folge der Zustände ist nicht sichtbar.
- Statt dessen: Zustände emittieren Beobachtungen O_t (mit Wahrscheinlichkeit $b_i(O_t)$).



Hidden-Markov-Modell
Basisprobleme

- Problem 1: Likelihood einer Beobachtungsfolge:
  - Berechne $P(O_1,...,O_T \mid \theta)$ — Forward-Algorithmus
- Problem 2: Optimale Zustandskette finden:
  - „Welche Zustandskette hat die Beobachtung am wahrscheinlichsten erzeugt?“
  - Berechne $\arg\max_{q_1,...,q_T} P(q_1,...,q_T \mid O_1,...,O_T,\theta)$ — Viterbi-Algorithmus
- Problem 3: Lernproblem:
  - „Gegeben viele Beobachtungsfolgen, finde die Parameter des HMMs!“
  - Berechne $\arg\max_\theta P(\{(O_1,...,O_T),...\} \mid \theta)$ — Baum-Welch-Algorithmus

Maschinelle Übersetzung

Maschinelle Übersetzung
Problemstellung

- Gegeben: Text in Quellsprache,
- Gesucht: Äquivalenter Text in Zielsprache.

- Anwendungen:
  - Übersetzung von Webseiten.
  - Ermöglicht Verstehen eines fremdsprachlichen Textes, auch wenn Übersetzung nicht genau ist.

Maschinelle Übersetzung
Teilprobleme

- Modellierung:
  - z.B. Finite State Transducer.
- Trainingsdaten:
  - Alignierung auf Satzebene z.B. längenbasiert.
  - Alignierung auf Wortebene z.B. durch wahrscheinlichste Zuordnung.
- Parameterschätzung:
  - EM-Algorithmus.
- Dekodierung:
  - z.B. Phrasenbasierte oder hierarchische Modelle.

Invertierter Index
Was ist das?

1 6 9 11 17 19 24 28 33 40 46 50 55 60

This is a text. A text has many words. Words are made from letters.

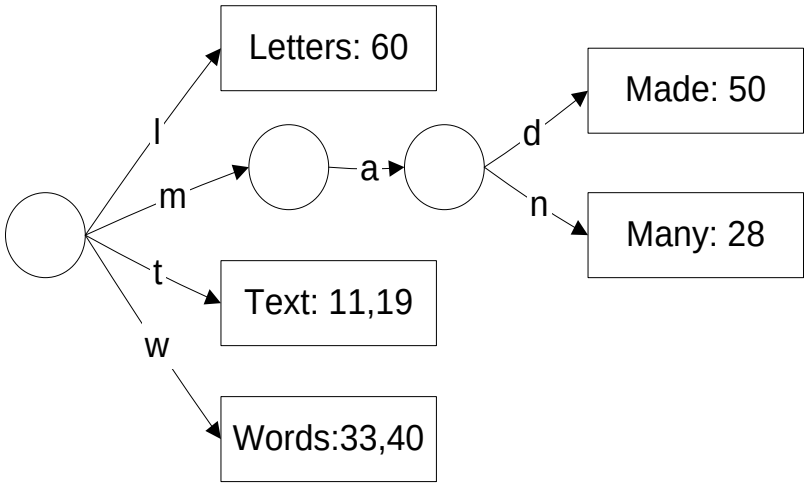
Terme	Vorkommen
Letters	60
Made	50
Many	28
Text	11, 19
words	33, 40

- Abbildung von Termen auf Dokumente / Textpositionen.
- Oft werden noch Zusatzinformationen gespeichert:
  - Schriftgrad, HTML-Tag

Invertierter Index
Tries

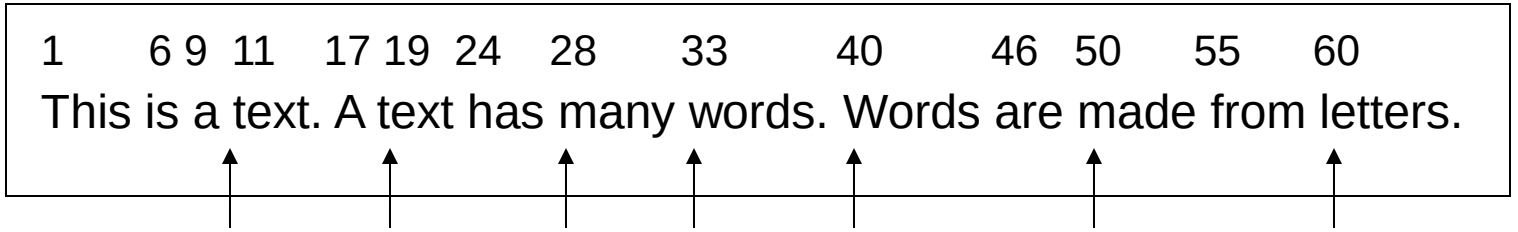
- Elementare Datenstruktur für Re*trie*val-Aufgaben.
- Kanten sind mit Buchstaben beschriftet.
- Knoten sind leer oder mit Wörtern und Fundstellen beschriftet.

1 6 9 11 17 19 24 28 33 40 46 50 55 60
This is a text. A text has many words. Words are made from letters.

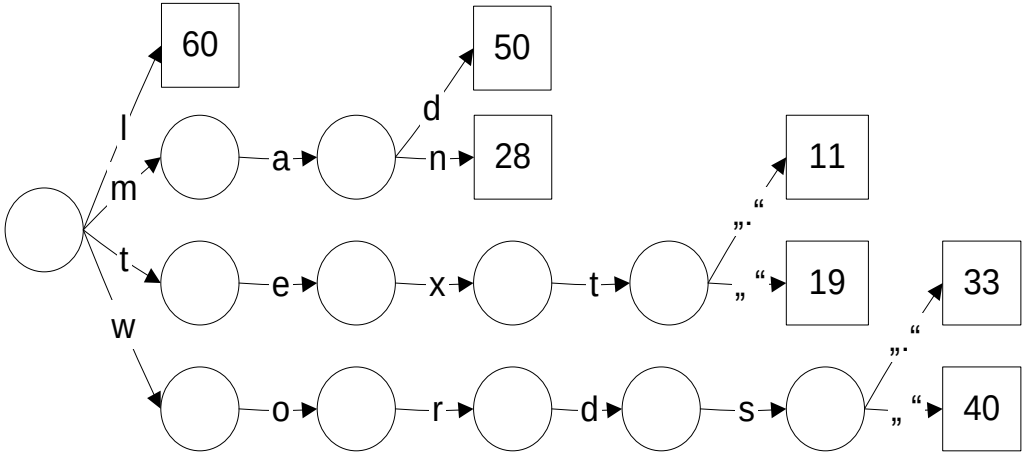


Invertierter Index
Suffix-Trie

- Aufbau eines Suffix-Tries:
- Für alle Indexpunkte:
  - Füge Suffix ab Indexpunkt in den Trie ein.



Suffix-Trie:

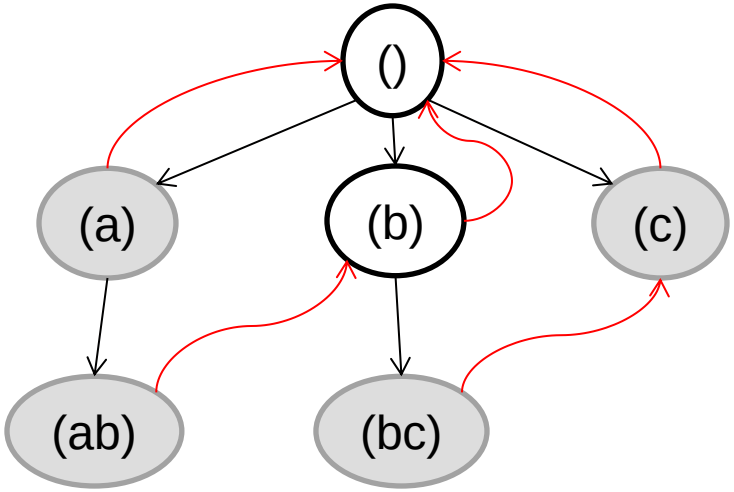


Suche
Knuth-Morris-Pratt

- Algorithmus, um ein Muster in einer Zeichenkette zu finden.
  - Nutzt Struktur des Suchmusters aus, indem teilweise Zeichen nach Missmatch übersprungen werden.
- Besteht aus zwei Teilen:
  - Vorlaufalgorithmus:
    - Bestimmt Präfixtabelle, die angibt, um wie viele Positionen das Muster bei einem Missmatch verschoben werden kann.
  - Suchalgorithmus:
    - Durchsucht Zeichenkette nach dem Muster mithilfe der Präfixtabelle.

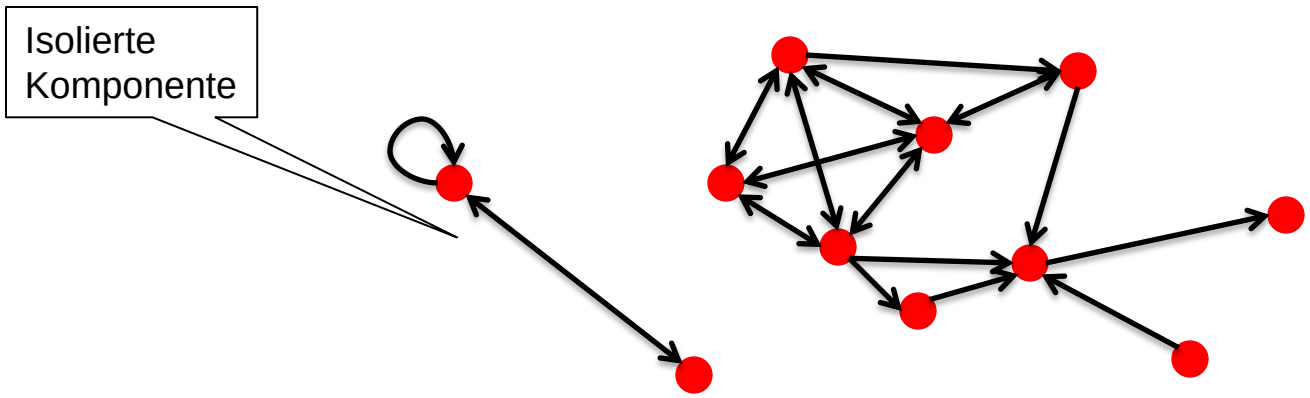
Suche
Aho-Corasick-Trie

- **Idee**: mehrere Suchstrings
  - Suchstrings werden in Automaten gespeichert,
  - Kanten akzeptieren Buchstaben,
  - Wenn keine passende Kante mehr existiert, Sprung über „failure transition“ in Zustand, der der größten Übereinstimmung zwischen Text und einem Präfix eines Suchstrings entspricht.



Websuche
Graph-Struktur

- Große zentrale, stark verknüpfte Komponente.
  - Jede Seite von jeder Seite erreichbar.
- Randgebiete die nur
  - Auf die zentrale Komponente verweisen;
  - Von zentraler Komponente aus verwiesen werden.
- Isolierte Komponenten.



Websuche
Indexierung: Crawler

- **Crawling**: Sammeln von Internetseiten, um sie indexieren zu können.
- **Aufgaben**:
  - Schnell neue Internetseiten besuchen.
  - Effizient arbeiten.
    - So viele brauchbare Internetseiten wie möglich in kurzer Zeit bearbeiten;
    - Relevante Informationen extrahieren und speichern.
- Implementierung von lauffähigen Crawlern sind im Netz frei verfügbar.

Ranking von Webseiten
Hubs & Authorities (HITS)

- Erst wird Menge der relevanten Dokumente bestimmt.
- Diese werden mittels des Hub & Authority Score sortiert.
- Hubs & Authorities (HITS):
  - Jeder Knoten besitzt…
    - *Hub Score:* Wie gut sind seine Verweise auf „kompetente“ Knoten
      = ausgehende Kanten.
    - *Authority Score:* Wie kompetent ist der Knoten, d.h. wie viele kompetente Knoten verweisen auf ihn
      = eingehende Kanten.

Ranking von Webseiten
Authority-Ranking: PageRank

- **Gegeben**: Trainingsdaten $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ mit gegebenen lokalen *Authority Scores* (Kompetenz-Bewertungen):
  - Authority Score $A_{ij} = a(\mathbf{x}_i, \mathbf{x}_j)$ gibt an wie „kompetent“ $\mathbf{x}_j$ aus Sicht von $\mathbf{x}_i$ ist.
  - Link oder kein Link, Link-Position.

Authority Matrix

- **Gesucht**: Modell $f : A \in R^{n \times n} \mapsto s \in R^n$ welches für Instanzen $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ globale Authority Scores (Ranking) s_i liefert.
  - Annahme: Je kompetenter $\mathbf{x}_i$ und je höher der Authority Score A_{ij}, desto kompetenter $\mathbf{x}_j$.

Random Surfer

22

Websuche
RankSVM

- Suchterme $\rightarrow$ Index $\rightarrow$ Liste von URLs.
- Texte mit mehreren Suchtermen durch Mengenoperationen.
- Aufbau eines Ähnlichkeitsvektors mit verschiedenen Merkmalen

$$\Phi(x,q) = \begin{pmatrix} sim(x,y) \text{ im Vektorraummodell} \\ \#\text{ gleicher Terme in } \langle \text{h1} \rangle\text{-Tags} \\ \ldots \\ \text{PageRank}(x) \end{pmatrix}$$

- Rückgabe der Seiten $\arg\max_x w\Phi(x,q)$
- Manuelle Konstruktion von w.
- Lernen von w aus Klickdaten.

RankSVM

- **Idee**: Lernen des Gewichtsvektors durch Klickdaten.

1. SVM-Support Vector Machines
www.support-vector-machines.org/
2. SVM-Light Support Vector Machine
svmlight.joachims.org/
3. Kernel-*Machines*.Org — Kernel *Machines*
www.kernel-machines.org/
***4. Support Vector Machines* - The Book**
www.support-vector.net/
5. *Support Vector Machines*
www.svms.org/
6. SVM - Support Vector Machines
www.dtreg.com/svm.htm

$$l_1 > l_2$$
$$l_1 > l_3$$
$$l_1 > l_5$$
$$l_1 > l_6$$
$$l_4 > l_2$$
$$l_4 > l_3$$
$$l_4 > l_5$$
$$l_4 > l_6$$

Themenmodellierung (LDA)

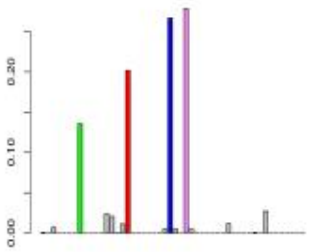
Chance and Statistical Significance in Protein and DNA Sequence Analysis

Samuel Karlin and Volker Brendel

Top words from the top topics (by term score)

sequence	measured	residues	computer
region	average	binding	methods
pcr	range	domains	number
identified	values	helix	two
fragments	different	cys	principle
two	size	regions	design
genes	three	structure	access
three	calculated	terminus	processing
cdna	two	terminal	advantage
analysis	low	site	important

Expected topic proportions



Abstract with the most likely topic assignments

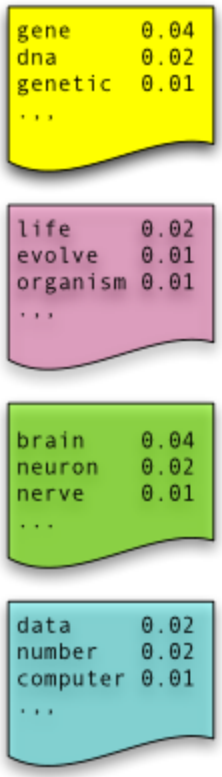
Statistical approaches help in the determination of significant configurations in protein and nucleic acid sequence data. Three recent statistical methods are discussed: (i) score-based sequence analysis that provides a means for characterizing anomalies in local sequence text and for evaluating sequence comparisons; (ii) quantile distributions of amino acid usage that reveal genera compositional biases in proteins and evolutionary relations; and (iii) *r*-scan statistics that can be applied to the analysis of spacings of sequence markers.

Top Ten Similar Documents

Exhaustive Matching of the Entire Protein Sequence Database
How Big Is the Universe of Exons?
Counting and Discounting the Universe of Exons
Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy for Multiple Alignment
Ancient Conserved Regions in New Gene Sequences and the Protein Databases
A Method to Identify Protein Sequences that Fold into a Known Three- Dimensional Structure
Testing the Exon Theory of Genes: The Evidence from Protein Structure
Predicting Coiled Coils from Protein Sequences
Genome Sequence of the Nematode C. elegans: A Platform for Investigating Biology

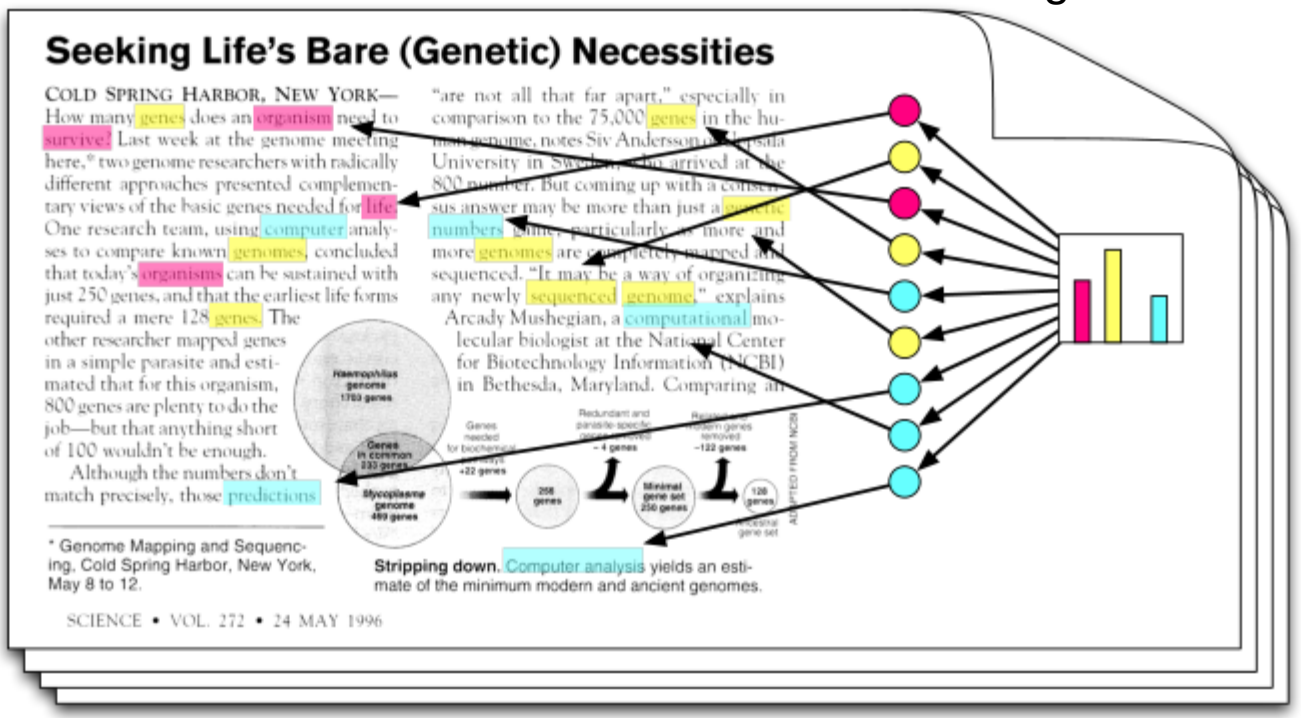
Themenmodellierung (LDA)
Generatives Modell

Themen



Dokumente

Themenanteile und Zuweisungen

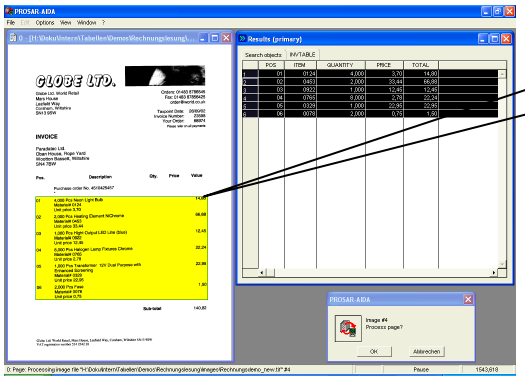


- Themen sind Verteilungen über die Wörter des Vokabulars.
- Jedes Dokument ist eine Mischung aus Themen.
- Jedes Wort ist aus einem der Themen gezogen.

Textklassifikation, Informationsextraktion

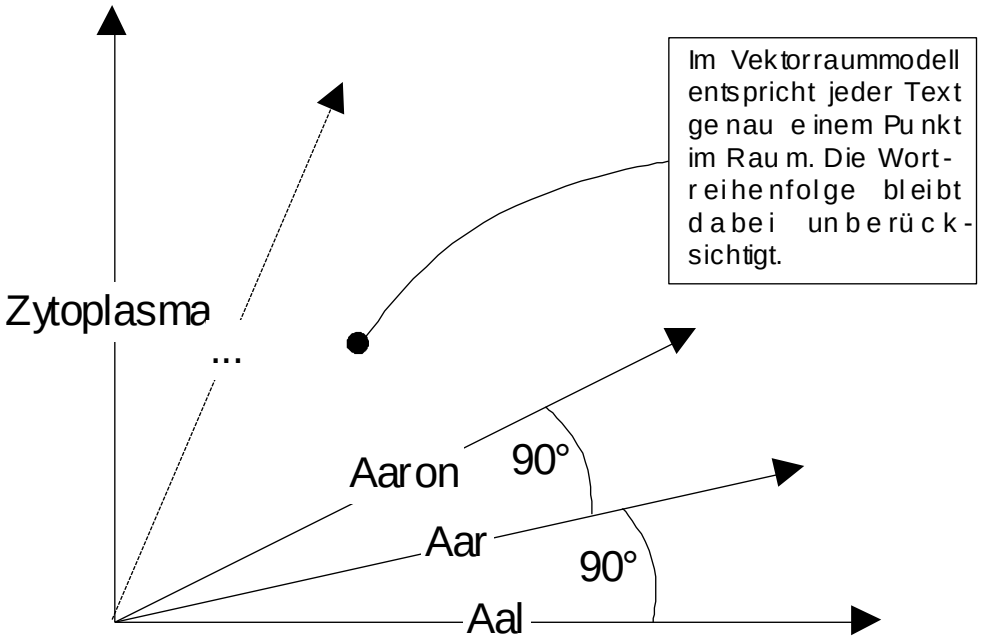
- Textklassifikation: Text → Kategorie
  - Wird i.d.R. aus annotierten Daten gelernt.
  - Anwendungsbeispiel: Posteingangsverarbeitung.
- Informationsextraktion: Identifikation definierter Felder in Dokument.
  - Wird i.d.R. auch aus Daten gelernt.
  - Anwendungsbeispiel: Automatisierung von Dokumentenverarbeitungsprozessen.

Textklassifikation: Dokument ist eine Rechnung



Informationsextraktion: Feld enthält den Preis

Textklassifikation
Vektorraum-Modell



- Text wird repräsentiert durch Punkt im hochdimensionalen Raum,
- Wortreihenfolge bleibt unberücksichtigt,
  - „Fußball ist toll“ und „Toll ist Fußball“ gleich.
- Variante: Wortstammbildung, „inverse document frequency“.

Textklassifikation
TFIDF-Repräsentation

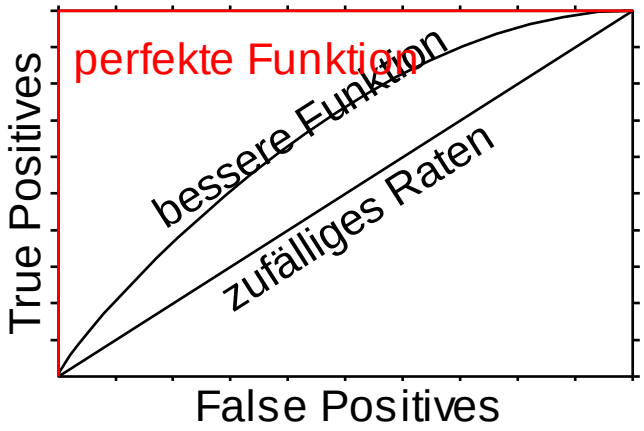
- Termfrequenz eines Wortes in einem Text = # Vorkommen des Wortes im Text.
- Problem 1: Einige Wörter sind weniger relevant
  - Z.B. und, oder, nicht, wenn, …
- Lösung: Inverse Dokumentenfrequenz

$$IDF(wort_i) = \log \frac{\#\text{Dokumente}}{\#\text{Dokumente , in denen Wort}_i \text{ vorkommt}}$$

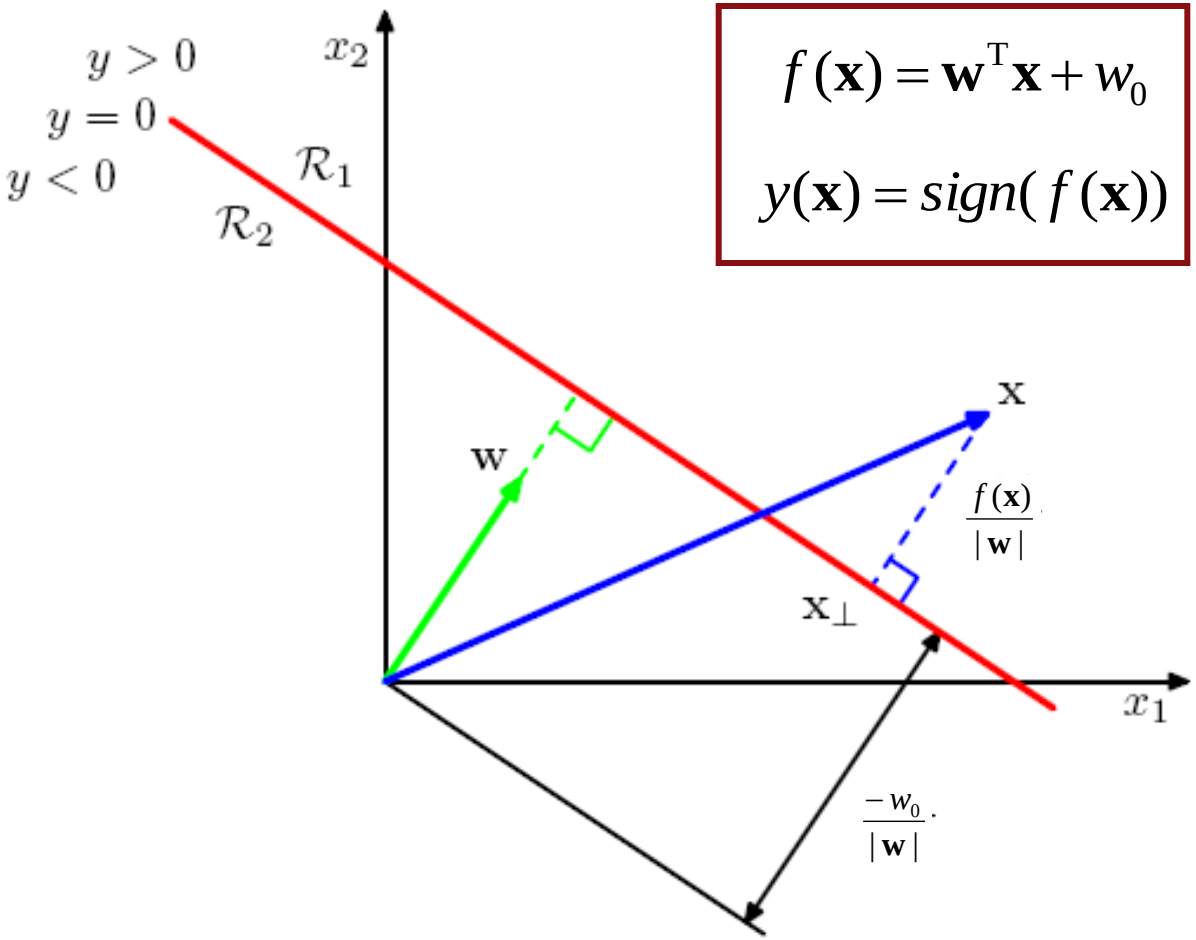
Textklassifikation
Evaluation

- Flächeninhalt AUC gibt an, wie gut gelerntes Modell ist.
- p = zufällig gezogenes Positivbeispiel
- n = zufällig gezogenes Negativbeispiel

- Theorem: AUC = P(f(p) > f(n)).



Textklassifikation
Lineare Klassifikatoren

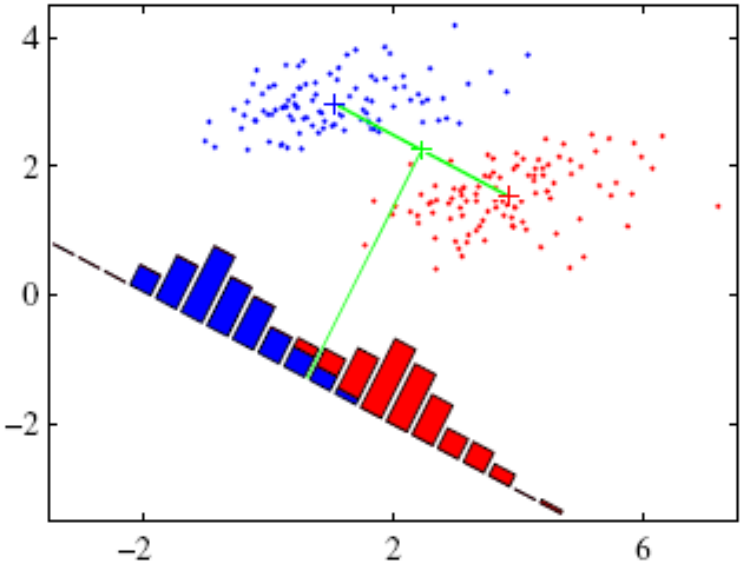


$$f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$$

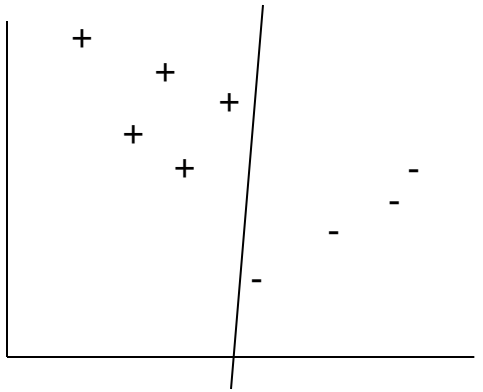
$$y(\mathbf{x}) = sign(f(\mathbf{x}))$$

Lineare Klassifikatoren
Rocchio

- Trennebene hat maximalen Abstand von den Mittelpunkten der Klassen.
- Trainingsbeispiele können falsch klassifiziert werden.
- Nachteil: Differenz der Mittelwerte kann schlechter Normalenvektor für Diskrimination sein.



Lineare Klassifikatoren
Perzeptron-Algorithmus



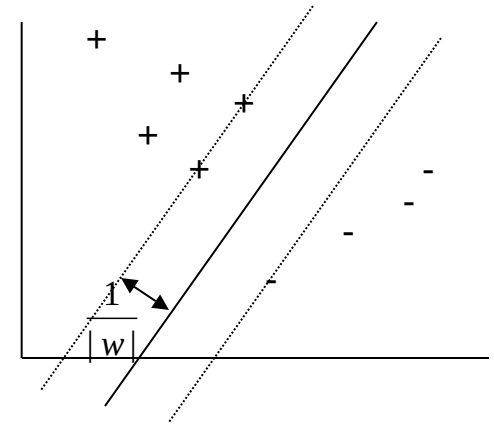
- Lineares Modell:
  - $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$
- Perzeptron-Trainingsalgorithmus:
- Solange noch Beispiele $(\mathbf{x}_i, y_i)$ mit der Hypothese inkonsistent sind ($\exists \mathbf{x}_i \in L . y_i \mathbf{w}^T \mathbf{x}_i \leq 0$), iteriere über alle Beispiele:
  - Wenn $y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i \leq 0$ dann $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$.

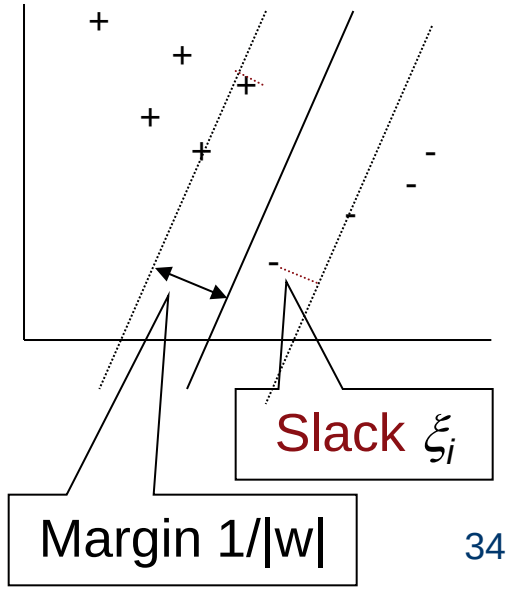
Lineare Klassifikatoren
Margin-Maximierung

- Hard-Margin-Maximierung:
  - Minimiere $|\mathbf{w}|$ unter der Nebenbedingungen:
  - für alle Beispiele $(\mathbf{x}_i, y_i)$: $y_i \mathbf{w}^T \mathbf{x}_i > 1$
- Soft-Margin-Maximierung:
  - Minimiere $|\mathbf{w}| + C \sum_i \xi_i$ unter den Nebenbedingungen:
  - für alle Beispiele $(\mathbf{x}_i, y_i)$:

    $$y_i \mathbf{w}^T \mathbf{x}_i > 1 - \xi_i$$

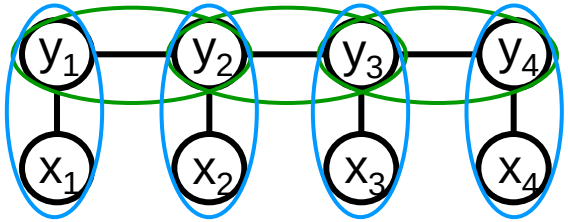
  - Alle $\xi_i \geq 0$.
- Soft-Margin-Ebene existiert immer, Hard-Margin-Ebene nicht!





Lernen mit strukturierten Ausgaberäumen

- Multiklassen-SVM.
- Klassifikation mit Taxonomien.
- Sequentielle Ein- und Ausgabe.

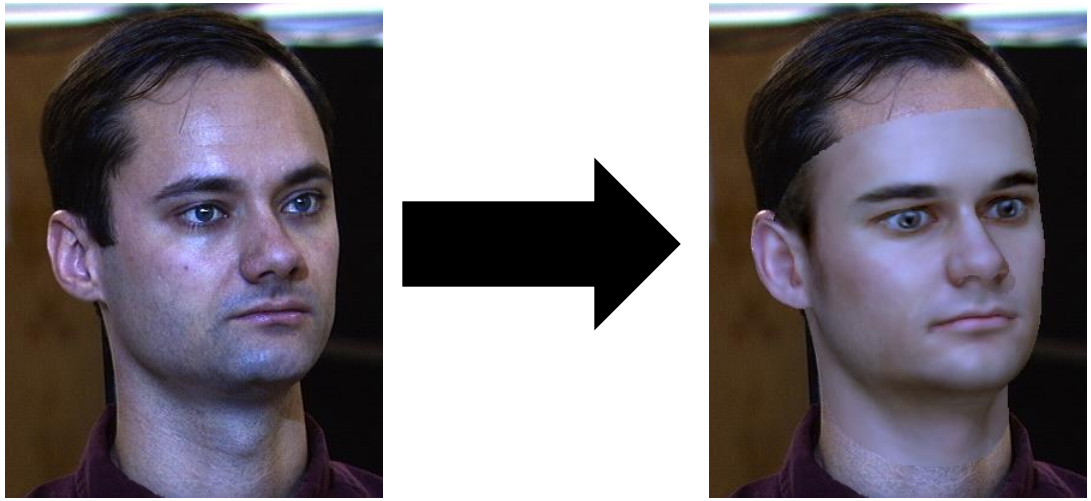


$\varphi_{123}(y_t, y_{t+1}) = [[y_t = \text{“Noun”} \wedge y_{t+1} = \text{“Verb”}]]$

$\bar{\varphi}_{234}(x_t, y_t) = [[y_t = \text{“Noun”} \wedge x_t = \text{“John”}]]$

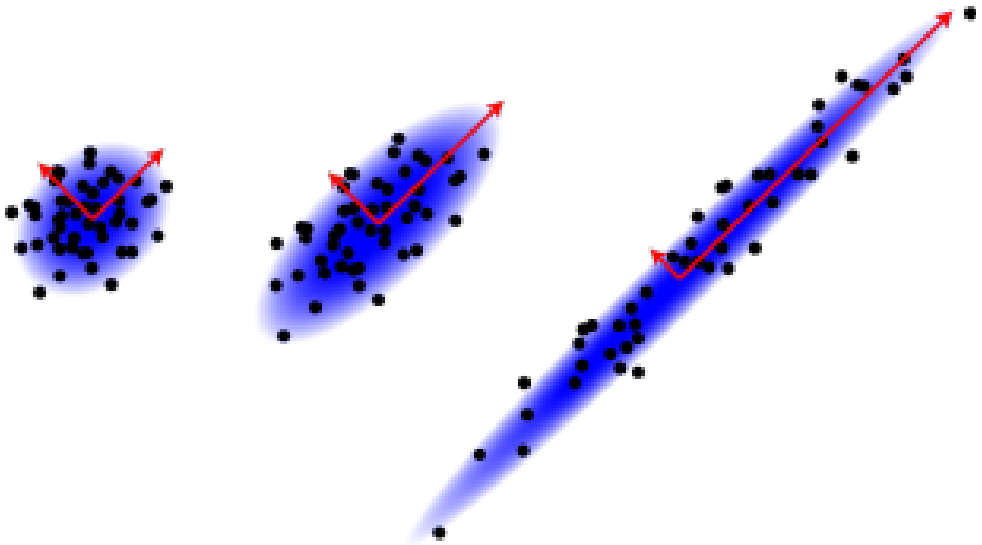
PCA
Anwendung & Interpretation

- Anwendungsbeispiel:
  - Morphace (Universität Basel)
    - 3D-Modelle von 200 verschiedenen Personen (jeweils über 150000 Feature)
    - PCA mit 199 Hauptkomponenten, jedes (3D) Gesicht wird durch 199 Parameter charakterisiert.



PCA

- PCA (Hauptkomponentenanalyse) projeziert Daten in neuen Raum:
  - Alle Komponenten sind unkorreliert.
  - Die Gesamtvarianz bleibt erhalten.



Sparse PCA

- Sparse PCA erzeugt sparse Hauptkomponenten.
  - Hauptkomponenten sind besser interpretierbar.
  - Sparse Daten sind auch im transformierten Raum sparse.

	σ(X)	1st PC	2nd PC
X_1	75.75	0.956	-0.288
X_2	13.13	0.294	0.945
X_3	0.61	0.015	-0.154
X_4	0.02	0.001	-0.002
λ		82.308	6.739

1st SPC	2nd SPC
1	0
0	1
0	0
0	0

Fragen?