

SPRACHTECHNOLOGIE IN SUCHMASCHINEN

IR-GRUNDLAGEN – TERMBASIERTE SUCHE

MASTERSEMINAR SUCHMASCHINEN UND RAG
COMPUTERLINGUISTIK
SOMMERSEMESTER 2024

STEFAN LANGER

STEFAN.LANGER@CIS.UNI-MUENCHEN.DE

IR – Grundlagen - Index

Texte durchsuchen:

- Sequenziell mit Zeichenkettenabgleich (string matching) oder mit regulären Ausdrücken (z.B. Unix grep) – dies stößt bei größeren Textmengen schnell an die Grenzen des Machbaren
- Mithilfe eines Indizes: Invertierte Dateien (inverted index)
 - Grundidee der Implementierung: Hash oder Trie (aber Indizes von großen Suchmaschinen sind noch zusätzlich optimiert)

Invertierter Index

Zerlegung des Dokuments in Terme

Im einfachsten Fall: Zuordnung von Termen zu Dokument-Ids

Dokument 1

Schöne Männer gehören nach Cannes wie
die Aschewolke an den isländischen
Himmel

Dokument 2

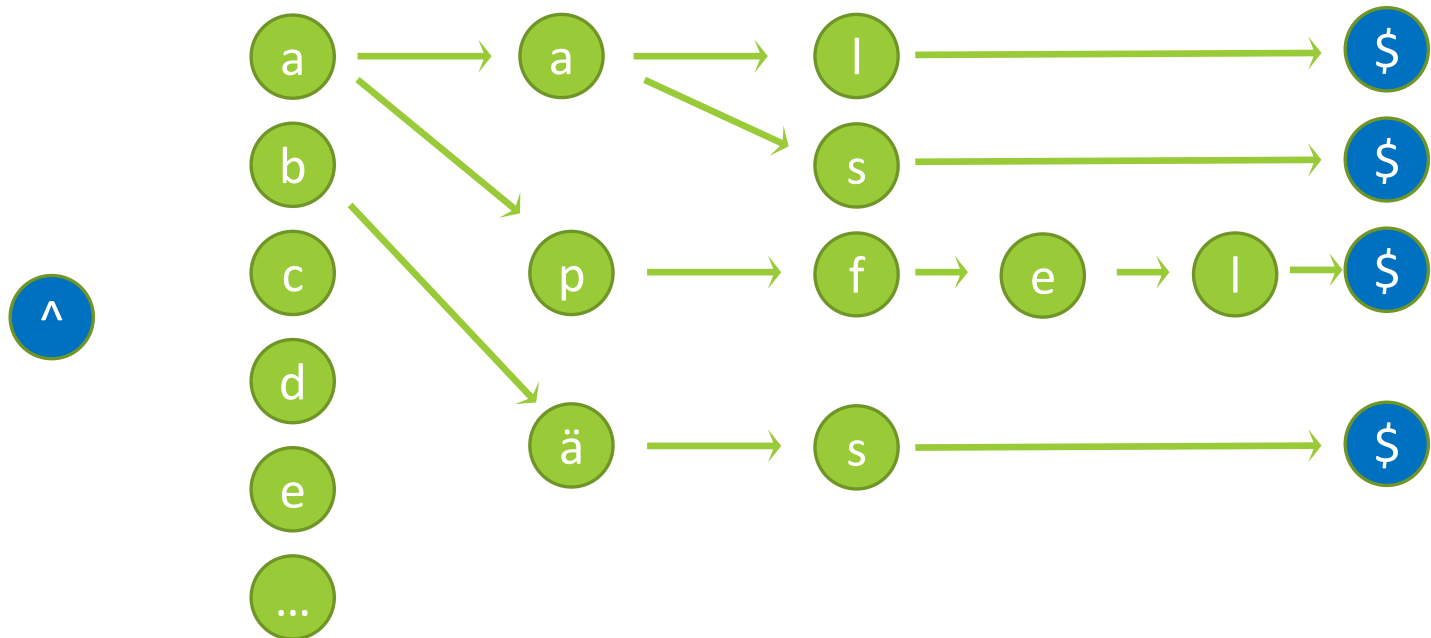
Getrocknetes Wasser, das vom Himmel
fällt und Tiere ohne Flügel, die trotzdem
fliegen können?

Index

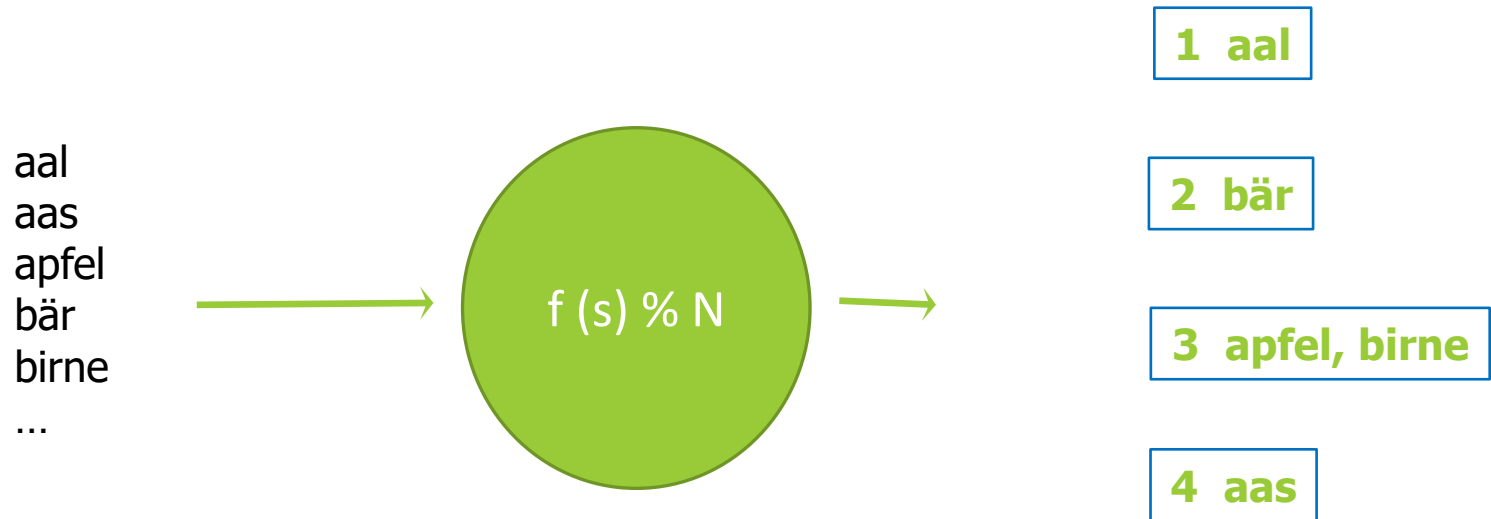
schöne.d1
männer.d1
...
himmel.d1/d2
wasser.d2
fliegen.d2

Trie

aal
aas
apfel
bär
...



Hash



Hash function examples: md5, sha-1,

Terminologie

Terminologie	Erklärung	Elasticsearch
Index	Index zum Speichern der Informationseinheiten (z.B. Tokens, Wortbigramme, Buchstaben N-Gramme)	Index, (Shard)
Feld	Teil eines Dokuments mit Label. Jedes indizierte Element gehört zu einem Feld	Field, property
Schema	Welche Felder existieren im Index, und welches Format haben sie (String, Integer, Boolean etc.)	Mapping, mapping type
Tokenisierung	Nach welchen Kriterien werden Token erzeugt	analyzer, tokenizer
Stemming	Linguistisch motivierte Trunkierung von Strings	stemming / stemmer
Facetten, Aggregationen	Unterteilung von Suchergebnissen aufgrund der Inhalte bestimmter Felder	aggregations
Filter	Teil einer Anfrage, der nicht zum Ranking verwendet wird	Filter

Retrieval mit boolschen Ausdrücken

Verknüpfung von Suchtermen mit UND/ODER/NICHT

Dokumenten-Ids für jede Teilquery

Bilde Schnittmengen (UND) /Differenzmengen (NICHT) / bzw. Vereinigungsmengen (ODER)

Effiziente Algorithmen verfügbar

- S. Abschnitt 1.3 im IR-Buch

Boolsche Operatoren

UND (AND): beide Teilausdrücke müssen in einem Dokument auftreten

Das ist der Default bei allen mir bekannten Websuchmaschinen

Beispiel: *UND("HAMLET","OPHELIA")*

ODER (OR): einer der Teilausdrücke muss im Dokument auftreten

Beispiel: *ODER("Streichholz","Zündholz")*

NICHT (NOT): negiere den dahinterstehenden Ausdruck

Beispiel: *UND("HAMLET",NOT("OPHELIA"))*

Kombinationen: z. B. ANDNOT = AND NOT

Komplexität Schnittmengenbildung

Die Komplexität der Schnittmengenbildung für einen Index (Mengen nach Größe sortiert):

- Naiver Ansatz: Vergleiche alle Elemente aus Menge 1 mit allen aus Menge 2: $O(n^2)$
- Verbesselter naiver Ansatz: Sortiere die Elemente aus Menge 1 oder schreibe sie in einen Hash/Dictionary. Lookup von allen Elementen aus Menge 2 in Menge 1 (binäre Suche oder Hash-Lookup): $O(n \log n)$ (sortieren)
- Index-Ansatz: lege die Mengen bereits nach Dokument-ID sortiert ab. Arbeite die sortierten Listen nach Dokument-ID ab. Linear – $O(N)$

Suche von Termfolgen und Nähe im durchsuchten Dokument

Beispiele für Operatoren:

PHRASE("Rot","Grün")

NAH(Rot,Grün)

NAH(Rot,Grün,<ABSTAND>)

GEORDNETES_NAH(Rot,Grün)

Lucene: Abfrage von Tokens

text:dalloway → "text" : "Mrs. **Dalloway** would by the flowers ..."

Finde alle Dokumente, die im Feld 'text' das Wort 'dalloway' enthalten.
Wenn das Feld ein Textfeld ist, wird die Anfrage durch den
Tokeniserer/Analyzer vorbehandelt, d.h. normalisiert.

Lucene: Phrasensuche / Proximity

text:"to the lighthouse" → "text" : "Tomorrow, we'll go to the lighthouse"

Suche im Feld 'text' die Wortsequenz 'to the lighthouse' (wird tokenisiert/normalisiert)

Zwischen den Wörtern können im Dokument keine weiteren Token stehen.

Proximity:

text:"to the lighthouse"~1 → "text" : "Tomorrow, we'll go to the red lighthouse"

Zwischen zwei der Wörter kann ein Token stehen (~ ist die Phrasen-Editierdistanz)

Lucene: Boolesche Suche

text:"to the lighthouse" OR text:dalloway

text:"Peter Walsh" AND text:dalloway

Lucene: Suche mit Platzhaltern

Wildcard ? : text:dall?way → dalloway, dallaway...

Wildcard * : text:lighthou*

Einschränkung: nicht in Phrasen, nicht als erstes Zeichen

Lucene: Unscharfe Suche

text:dalloway~

text:dalloway~1

Je höher der Wert nach der Tilde, desto unschärfer die Suche,
Defaultwert ist 1, Maximalwert 2 – Levenshtein-Distanz

Einschränkungen:

- nicht in Phrasen (dort ist es Phrasen-Editierdistanz),
- nicht als erstes Zeichen

Lucene query string syntax in elasticsearch

GET /_search

```
{ "query":  
  { "query_string" :  
    { "default_field" : "text", "query" : "this AND that OR thus" }  
  }  
}
```

Ranking / Scoring

Beim Ranking geht es darum, für eine gegebene Anfrage zu ermitteln, wie gut ein Dokument passt, um die Reihenfolge der Ergebnisse festzulegen.

Einige Parameter:

- Gewicht der Anfrage-Terme in der Query
- Gewicht der Dokumententerme
 - Abhängig von der Termfrequenz im Dokument
 - Abhängig von der Termfrequenz über alle Dokumente
 - Abhängig vom Feld, in dem ein Term auftritt (Dokumentenstruktur)
- Statischer Score des Dokuments / Felds

Rankingfunktionen

- (Okapi) BM25

$$\text{score}(D, Q) = \sum_{i=1}^n \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

- D: Dokument
- q: query-Terme
- k, b: Parameter (1.2, 0.75)

BM25F: + Felderinformation

Lucene scoring I

score(q,d) =

$$\text{coord}(q,d) \cdot \sum (\text{tf}(t \text{ in } d) \cdot \text{idf}(t)^2 \cdot t.\text{getBoost}() \cdot \text{norm}(t,d))$$

Wobei:

- $\text{tf}(t \text{ in } d) = \text{frequency}^{\frac{1}{2}}$
- $\text{idf}(t) = 1 + \log (\text{numDocs}/(\text{docFreq}+1))$
- $\text{coord}(q,d)$ Ist eine Bewertungsfaktor, der darauf basiert, wie viele der Anfrageterme im Dokument gefunden wurden. Standard = 1
- $t.\text{getBoost}()$ Ist ein Boost für term t in Anfrage q , wie in der Anfrage(manuell) spezifiziert
Bsp: *jakarta^4 apache*

Lucene scoring II

$\text{score}(q,d) =$

$$\text{coord}(q,d) \cdot \text{queryNorm}(q) \cdot \sum (\text{tf}(t \text{ in } d) \cdot \text{idf}(t)^2 \cdot t.\text{getBoost}() \cdot \text{norm}(t,d))$$

norm(t,d) encapsulates a few (indexing time) boost and length factors:

- **Document boost** - set by calling `doc.setBoost()` before adding the document to the index.
- **Field boost** - set by calling `field.setBoost()` before adding the field to a document.
- **lengthNorm** - computed when the document is added to the index in accordance with the number of tokens of this field in the document, so that shorter fields contribute more to the score. LengthNorm is computed by the similarity class in effect at indexing.

Lucene scoring III

In der Formel fehlt:

- Termnähe (proximity) als Faktor

Mögliche Option:

~ "sloppy phrase query": text:"Leonce Lena"~100

- → max Distanz 100, sortiert nach Distanz