RECHTSCHREIBKORREKTUR UND ANDERE APPROXIMATIVE SUCHMETHODEN

MASTERSEMINAR SUCHMASCHINEN UND RAG

SOMMERSEMESTER 2025

STEFAN LANGER, CIS, UNIVERSITÄT MÜNCHEN

Übersicht

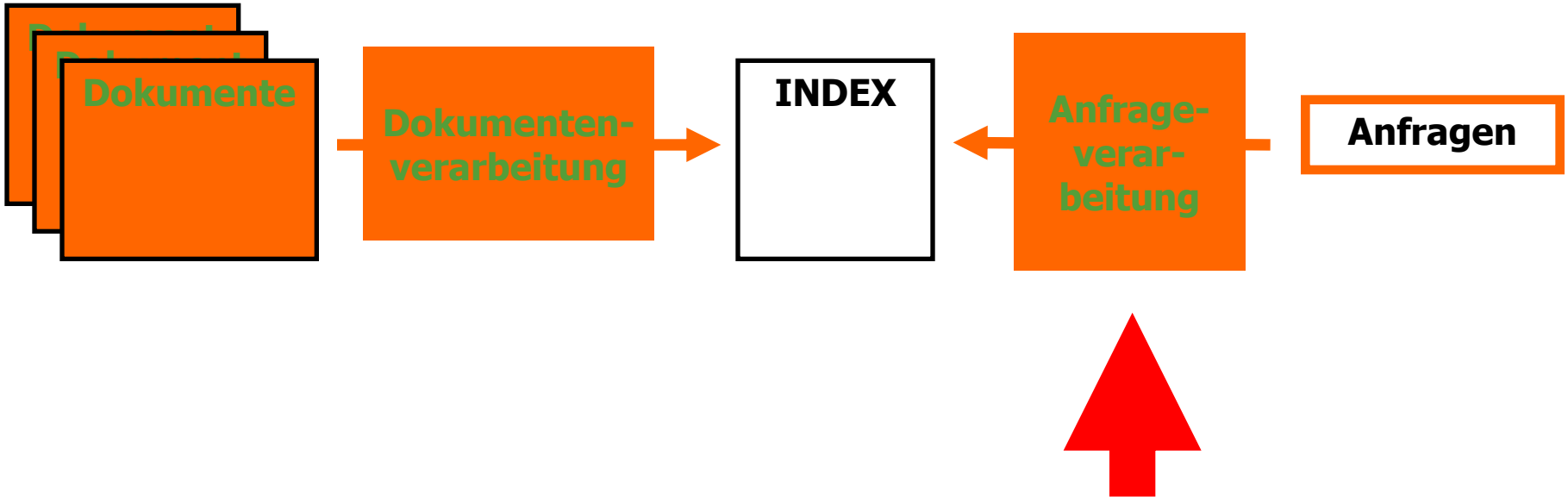
Spellchecking

Vervollständigung

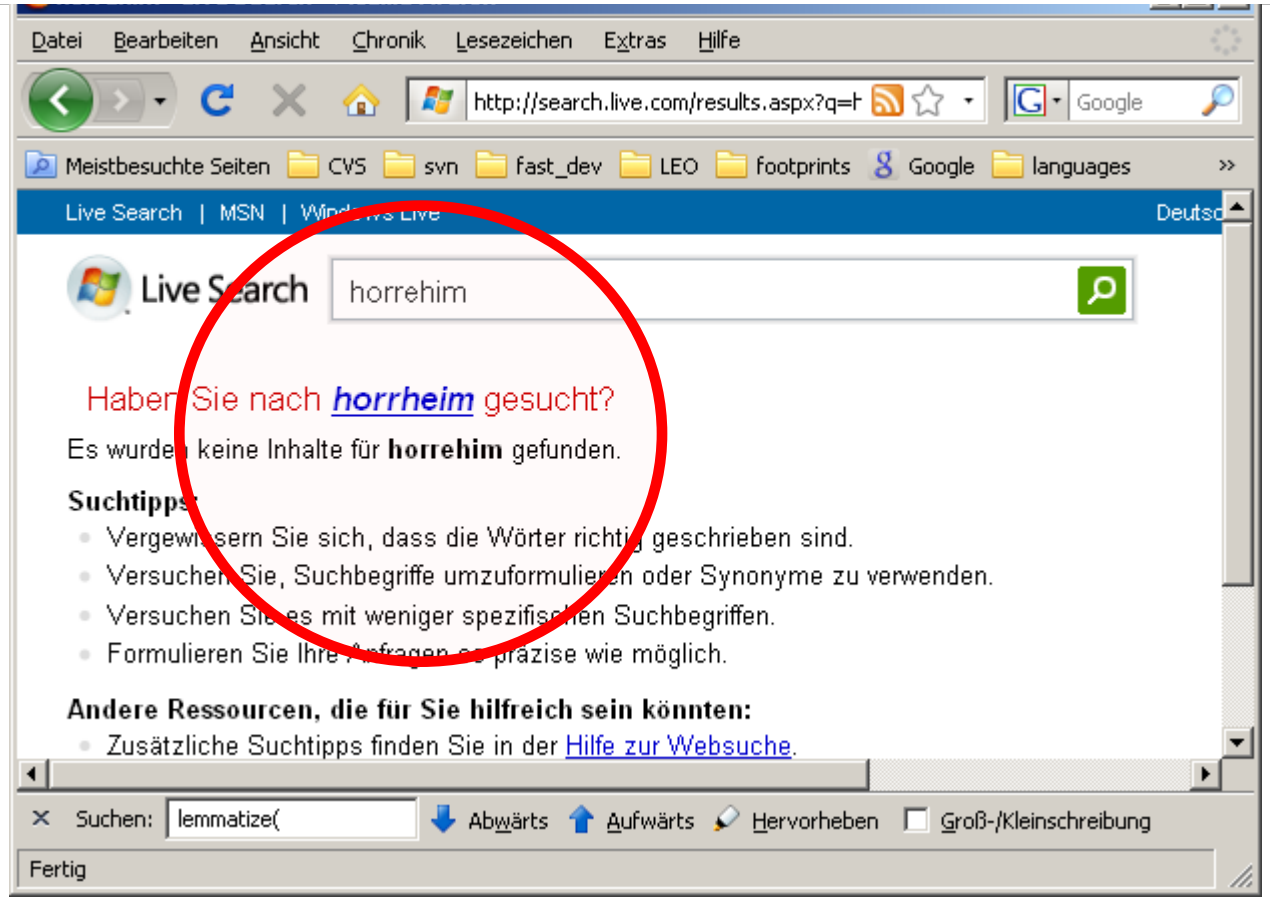
Phonetische Suche

Transliteration

Schematische Architektur einer Suchmaschine



Beispiel


Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://search.live.com/results.aspx?q=h
Google
Meistbesuchte Seiten CVS svn fast_dev LEO footprints Google languages
Live Search | MSN |
Live Search
horrehim
Haben Sie nach horrheim gesucht?
Es wurden keine Inhalte für horrehim gefunden.
Suchtipps:
Vergewissern Sie sich, dass die Wörter richtig geschrieben sind.
Versuchen Sie, Suchbegriffe umzuformulieren oder Synonyme zu verwenden.
Versuchen Sie es mit weniger spezifischen Suchbegriffen.
Formulieren Sie Ihre Anfragen so präzise wie möglich.
Andere Ressourcen, die für Sie hilfreich sein könnten:
Zusätzliche Suchtipps finden Sie in der Hilfe zur Websuche.
Suchen: lemmatize(
Abwärts Aufwärts Hervorheben Groß-/Kleinschreibung
Fertig

Rechtschreibkorrektur - Grundidee

Korrigiere Fehlschreibungen basierend auf einer Liste von korrekten Wörtern

Horrehem

Horrheim

Horrem

Horrend

Welches ist die richtige Korrektur?

→Abstandsmaß

Welches ist die korrekt Wortliste?

→ Datenquellen und Filterung

Levenshtein-Distanz

Einfügen:

- Waser → Wasser

Löschen:

- Wasssser → Wasser

Ersetzen:

- Waszer → Wasser

Levenshtein-Algorithm

```
int LevenshteinDistance(char s[1..m], char t[1..n])
  declare int d[0..m, 0..n]

   for i from 0 to m
      d[i, 0] := i
   for j from 0 to n
      d[0, j] := j

   for i from 1 to m
      for j from 1 to n
      {
         if s[i] = t[j] then cost := 0
                        else cost := 1
         d[i, j] := minimum(
                           d[i-1, j] + 1,     // deletion
                           d[i, j-1] + 1,     // insertion
                           d[i-1, j-1] + cost   // substitution
                         )
      }
   return d[m, n]
```

Levenshtein-Tabelle

		K	A	T	Z	E
	0	1	2	3	4	5
K	1	0	1	2	3	4
A	2	1	0	1	2	3
D	3	2	1	1	2	3
S	4	3	2	2	2	3
E	5	4	3	3	3	**2**

Levenshtein - Probleme

Keine Distanzunterschiede zwischen unterschiedlichen Buchstabenpaaren (z.B. *t*-*d* / *t*-*x*) – kann aber integriert werden

- WLD: weighted Levenshtein distance

Vertauschung von Buchstaben erhält keine spezielle Beachtung

- Alternative: Damerau-Levenshtein distance)

Levenshtein – zusätzliche Heuristiken

Korrigiere nie Wörter zu Wörter mit anderen Anfangsbuchstaben

Korrigiere keine Wörter unter N Zeichen (N=3 oder 4)

Achte auf problematische Wörter in der Liste der korrekten Wörter

worehouse → warehouse nicht *whorehouse*

Rechtschreibkorrektur – woher kommt die Wortliste

Frequenzliste aus

- Dokumenten
- Suchtermen

Korrekte Wörter/Wortformen

- Wörterbuch

Ausnahmelisten

- Positive Ausnahmeliste
- Negative Ausnahmeliste

Anfragevervollständigung

Anforderungen
- Schnell!
- Relevant
- search as you type?

Schlage während des Tippens mögliche vollständige Anfragen vor

Datenquellen
- Anfragen
- Dokumente

Matching
- Infix
- Prefix
- ….

Ranking

Ähnliche Ranking-Herausforderungen wie bei der Suche

Andere approximative Suchmethoden

Normalisierung

- Eur. Sprachen: Diakritika u.a. nicht-ASCII-Buchstaben
- Arabisch: Vokalzeichen
- CJK: Half-Width ; Full Width;

Transliteration, Transkription

Phonetische Suche durch phonetische Normalisierung

- z.B. Soundex, Kölner Phonetik

N-Gram-Suche

- Bsp. *Immobilien %im|imm|mmo|mob|obi|bil|ili|lie|ien|en%*

Phonetische Suche – Beispiel Soundex

Retain the first letter of the name and drop all other occurrences of a, e, i, o, u, y, h, w.

Replace consonants with digits as follows (after the first letter):

- b, f, p, v => 1
- c, g, j, k, q, s, x, z => 2
- d, t => 3
- l => 4
- m, n => 5
- r => 6

Two adjacent letters (in the original name) with the same number are coded as a single number; also two letters with the same number separated by 'h' or 'w' are coded as a single number, whereas such letters separated by a vowel are coded twice. This rule also applies to the first letter.

Continue until you have one letter and three numbers. If you run out of letters, fill in 0s until there are three numbers.

Buchstaben-N-Gram-Suche

N ist normalerweise 2-4.

Tri-Gram-Suche

- Bsp. *Immobilien*
  %im|imm|mmo|mob|obi|bil|ili|lie|ien|en%

Anwendungsgebiete:

- CJK-Sprachen ohne Tokenisierung

- Namens- und Entitätensuche

- Suche in Buchstabenfolgen (z.B. DNA)

Lucene (Elasticsearch)

NGramTokenizer(int minGram, int maxGram)
Creates NGramTokenizer with given min and max n-grams.