Boolesches Retrieval

Weitere wichtige Retrievaloperatoren

Phrasenanfragen

Ein Auftreten einer Phrase in einem Dokument ist eine Sequenz von Einzelwörtern.

Phrase Queries oder **Wortphrasenanfragen** suchen nach Auftreten von Phrasen in Dokumenten. Separatoren und Stoppwörter werden dabei (oft) nicht betrachtet.

Beispiel: Phrasenanfragen

- Unsere Anfrage sei die Phrase “stanford university”.
- Daher soll
  The inventor Stanford Ovshinsky never went to university
  kein Treffer sein.
- Das Konzept der Wortphrasenanfrage wird von vielen Benutzern leicht verstanden.
- Etwa **10% aller Webanfragen** enthalten Phrasen.

Es genügt nun offenbar nicht mehr, nur die docIDs im invertierten Index zu speichern, wenn auch Phrasenanfragen verarbeitet werden sollen.

- **Warum ist das so?**
- **Wie kann man das reparieren?**

Wortpaarindexe

Wortpaarindexe indexieren jedes **aufeinanderfolgende Paar von Termen** in einem Dokument als Phrase.

Das Dokument

Friends, Romans, Countrymen

generiert zum Beispiel die beiden Wortpaare "**friends romans**" und "**romans countrymen**", die beide einen (komplexen) Term des Vokabulars bilden.

Mit einem solchen Index können Phrasen aus zwei Wörtern einfach beantwortet werden. Komplexere Phrasen erfordern aber zusätzlichen Aufwand.

Phrasen mit mehr als zwei Wörtern

Eine lange Phrase wie **“stanford university palo alto”** kann als folgende Boolesche Anfrage repräsentiert werden.

> **“stanford university”** AND **“university palo”** AND **“palo alto”**

Das Ergebnis der Anfrage ist aber nur eine **Obermenge der Phrasenanfrage** (warum?). Wir müssen die Treffer noch filtern, um die Teilmenge zu bestimmen, die die Vier-Wort-Phrase tatsächlich enthält.

Grundsätzlich kann man mit entsprechendem Aufwand auch längere Sequenzen als Paare indexieren und so die Anfrageausführung vereinfachen.

Probleme mit Wortpaarindexen

Warum werden Wortpaarindexe selten verwendet?

- Falsch-positive Ergebnisse, die eine Filterung der Ergebnisse erforderlich machen
- Index kann sehr groß werden, da das Vokabular sehr groß werden kann

Positionsindexe

Ein **Positionsindex** besteht wie ein invertierter Index aus einem Vokabular und einer Positionsliste. Er speichert dabei zusätzlich für jeden Term **t** aus dem Vokabular seine **Positionen** für jedes Dokument, in dem er auftritt, in der Form

$$
\begin{aligned}
&t, \mathrm{DFreq} : \langle \\
&\quad \mathrm{DocId}_1, \mathrm{TFreq}_1 : \langle\, \mathrm{pos}_{11}, \ldots, \mathrm{pos}_{1n_1} \,\rangle \\
&\quad \ldots \\
&\quad \mathrm{DocId}_m; \mathrm{TFreq}_m : \langle\, \mathrm{pos}_{m1}, \ldots, \mathrm{pos}_{mn_m} \,\rangle \\
&\rangle
\end{aligned}
$$

Dabei sind

- DFreq: Die **Dokumenthäufigkeit**, d.h. die Anzahl der Dokumente, in denen **t** vorkommt.
- DocId_i : Der Identifikator des i-ten Dokuments, in dem **t** vorkommt.
- TFreq_i : Die Anzahl der Positionen, an denen **t** in Dokument DocId auftritt.
- $\mathrm{Pos}_{i1}, \ldots, \mathrm{pos}_{in_i}$: Die Positionen in aufsteigender Reihenfolge, an denen **t** in Dokument DocId auftritt.

Proximity-Queries (Nachbarschaftsanfragen)

Proximity-Queries oder **Nachbarschaftsanfragen** stellen eine verallgemeinerte Form der Phrase Queries dar.

Bei Proximity Queries wird nicht die genaue Wortsequenz gesucht, sondern Textstellen, in denen die angegebenen Einzelwörter einen bestimmten **Maximalabstand nicht überschreiten**. Manchmal wird auch von der konkreten Reihenfolge der Einzelwörter abstrahiert.

Beispiel:

Die Anfrage

employment /3 place

sucht nach Dokumenten, die **employment** und **place** im Abstand von höchstens drei Wörtern enthalten.

Employment agencies that place healthcare workers are seeing growth ist also ein Treffer.

Employment agencies that help place healthcare workers are seeing growth ist dagegen kein Treffer.

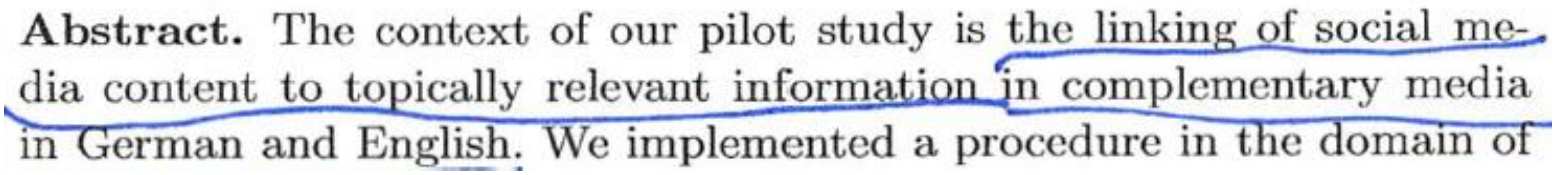
Rechtschreibkorrektur / Spelling Correction

Verschiedene Ansätze zur Rechtschreibkorrektur

Es gibt zwei mögliche Ansätze für die Rechtschreibkorrektur

- Korrektur von **Dokumenten** vor der Indizierung;
- Korrektur von **Anfragen**.

Im Folgenden wird nur der zweite Ansatz betrachtet, der erste wird nur in Sonderfällen eingesetzt (z.B. bei der Suche in Dokumenten, die aus einem OCR-System kommen).



Abstract. The context of our pilot study is the linking of social media content to topically relevant information in complementary media in German and English. We implemented a procedure in the domain of

OCR mit ABBYY FineReader 8.3

```
Abstract. Thc context of our pilot study is the linking of social me-^
^dia Content to topicallv relevant informatinn jn rnmnlementarv media
m Uerman and English. We implementcd a proccdure ^n thc domain ot
```

Methodische Ansätze der Rechtschreibkorrektur

Man unterscheidet folgende Ansätze zur Rechtschreibkorrektur:

- **Korrektur isolierter Terme** (isolated-term correction);
  - Jeder einzelne Anfrageterm wird separat behandelt.
  - Diese Methode kann keine fehlerhafte Anfrage, die aus korrekten Termen besteht, erkennen.
  - Beispiel: „Ein Stern **v**iel vom Himmel“ wird nicht als falsch erkannt
- **Kontext-sensitive Korrektur** (context-sensitive correction).
  - Es wird die gesamte Anfrage behandelt;
  - Basis: Größe der Anfrageergebnisse für mögliche Korrekturen, Biword-Statistik, Korrekturverhalten der Nutzer, . . .
  - Beachte: Es findet in der Regel keine grammatikalische Prüfung statt.

Korrektur isolierter Terme

Grundannahmen:

- Es existiert eine **Liste der korrekten Wörter** bzw. Terme.
  - Möglichkeit: Wähle das vorhandene Term-Vokabular des invertierten Index.
    Dies kann problematisch sein (warum?).
  - Alternativen: Standardwörterbuch, domänenspezifische Wörterbücher, gewichtetes Term-Vokabular
- Es gibt eine Methode zur **Berechnung der Distanz** zwischen einem korrekten und einem fehlerhaften Wort.
  - Es wird das „korrekte“ Wort mit der kleinsten Distanz zu dem inkorrekten Anfrageterm gewählt.
  - Beispiel: Statt „Infromation“ wähle „Information“
  - Distanz zählt notwendige Änderungen (Einfügen, Löschen, Ändern von Zeichen), um Wort A in Wort B zu konvertieren

Gewichtete Edit-Distanz

Man kann die Edit-Distanz verfeinern, indem man das **Gewicht einer Grundoperation** in Abhängigkeit von den behandelten Zeichen definiert.

Beispiel:

- Es ist wahrscheinlicher, dass bei der Tastatureingabe das Zeichen **m** mit **n** verwechselt wird als mit **e**.
- Die Ersetzung von **m** durch **n** erhält daher eine kleinere Edit-Distanz als die Ersetzung durch **e**.

Diese Gewichte werden **statistisch** aus einer großen Menge von Fehlerkorrekturen geschätzt.

Einsatz der Edit-Distanz zur Rechtschreibkorr.

Aufgabe:

Gegeben seien eine Menge S von Strings (entsprechend den Termen im Vokabular) und ein Anfragestring q. Gesucht sind die Strings $s \in S$ mit minimaler Edit-Distanz von q.

Probleme

- Erschöpfende Suche, d.h. Berechung der Edit-Distanz zu q für alle $s \in S$ und Selektion der s mit minimaler Edit-Distanz, ist in der Regel zu aufwändig und skaliert nicht.
- **Heuristiken** sind erforderlich, um eine effiziente Suche zu bewirken.
  Beispiel:
  Betrachte nur die $s \in S$, die mit dem gleichen Zeichen beginnen wie q.

Spelling Corrector von Peter Norvig

Einen völlig anderen Weg geht der von Peter Norvig vorgeschlagene **Spelling Corrector**. Er beruht auf einer **statistischen Analyse** großer Textcorpora.

http://norvig.com/spell-correct.html

Spelling Corrector von Peter Norvig

Das Verfahren berechnet die **Wahrscheinlichkeit** P(c|w), dass Term c gemeint ist, wenn Term w geschrieben wurde, mit dem Satz von Bayes als

$$P(c|w) = \frac{P(w|c) \cdot P(c)}{P(w)}$$

Hierbei sind

- P(c) die Wahrscheinlichkeit, dass c in einem Text vorkommt (das **Sprachmodell**); je häufiger c ist, desto höher ist P(c).
- P(w|c) die Wahrscheinlichkeit, dass w geschrieben wird, wenn c gemeint ist (das **Fehlermodell**); sie ist im wesentlichen proportional zur gewichteten Editdistanz von w und c.
- P(w) die Wahrscheinlichkeit, dass w in einem Text vorkommt; weil w fest ist, spielt sie keine Rolle.

Würde man direkt P(c|w) berechnen, würde man Fehlermodell und Sprachmodell in einer Wahrscheinlichkeit mischen; isoliert sind beide viel besser zu kontrollieren.

Das Verfahren wählt dann das c, das P(c|w) maximiert. Norvigs Verfahren berechnet diese Wahrscheinlichkeit nicht explizit, sondern wählt zunächst die Korrektur mit minimaler Edit-Distanz; gibt es mehrere davon, wird die mit maximalem P(c) ausgewählt.

Kontext-sensitive Rechtschreibkorrektur

Beispielanfrage:

Ein Stern viel vom Himmel

Gewünscht wird ein Korrekturvorschlag wie

Ein Stern fiel vom Himmel

Alle Terme der Anfrage sind einzeln korrekt. Eine Korrektur kann also nur unter Berücksichtigung des **Kontextes** erfolgen.

Hit-basierte Rechtschreibkorrektur

Ein einfaches (und nicht sehr effizientes) Verfahren zur kontext-sensitiven Rechtschreibkorrektur untersucht mögliche Ersetzungen für die **einzelnen Anfrageterme** durch Terme mit geringer Edit-Distanz und zählt die Anzahl der mit der so modifizierten Anfrage gefundenen Dokumente.

Beispiel:

- **Stern**: Stein, Stier, Adern
- **viel**: fiel, vier, kiel, vieh
- **vom**: vor, von, Rom
- **Himmel**: Hummel, Hammel

Das Verfahren durchläuft anschließend die folgenden Schritte:

- Ersetze in der Originalanfrage Q jeweils einen Term durch einen der gefundenen „ähnlichen“ Terme und erhalte eine modifizierte Anfrage Q0.
- Stelle alle so erhaltenen Anfrage Q0 und ermittle die Anzahl der Resultat-Dokumente.
- Wähle die Anfrage Q0 mit den meisten Treffern als die korrekte Anfrage.

Probleme der hit-basierten Rechtschreibkorrektur

Das Verfahren der hit-basierten Rechtschreibkorrektur ist nicht sehr effizient.

Bessere Alternative:

Wähle korrigierte Anfrage mit höchster geschätzter Wahrscheinlichkeit in einer großen Dokumentkollektion.

Wahrscheinlichkeit einer Wortsequenz

Wir möchten die Wahrscheinlichkeit schätzen, mit der die Wortsequenz

$w_1\ w_2 \ldots w_n$

in einer großen Dokumentkollektion vorkommt. Weil wir das nicht effizient vorberechnen können, zerlegen wir die Sequenz in ihre Bigramme (auf Wortebene) und nehmen Unabhängigkeit der Bigramm-Vorkommen an.

Damit erhalten wir

$$P(w_1\ w_2 \ldots w_n) = P(w_1) \cdot P(w_2|w_1) \ldots P(w_n|w_{n-1})$$

Dabei ist $P(w_k|w_i)$ die Wahrscheinlichkeit, dass in einem Text das auf Wort w_i direkt folgende Wort w_k ist; wir schätzen dies wieder durch Statistiken auf einem großen Korpus:

$$P(w_k|w_i) = \frac{\#\text{Vorkommen von } (w_i, w_k) + 1}{\#\text{Vorkommen von } w_i + 1}$$

Die beiden "+1"dienen dabei der Glättung, da Bigramme auftreten können, die nicht im Text vorkommen, aber trotzdem aus korrekten Wörtern bestehen und daher nicht ausgeschlossen werden dürfen.

Spelling Correction mit Wortsequenzen

Gegeben sei eine Anfrage. Wir generieren wie bei der hitbasierten Methode alle Kombinationen von Alternativen für die Terme der Anfrage und wählen die Sequenz mit der höchsten geschätzten Wahrscheinlichkeit.

Zur Verbesserung der Laufzeit kann man

- annehmen, dass nur ein Fehler pro Anfrage auftritt;
- nur Korrekturterme aus dem Vokabular erlauben, d.h. solche, die irgendwo im Text vorkommen.

Beispiel: Bigramm-Methode

Betrachte die Anfrage

„a stellar and versatile acress whose combination of sass and glamour …“

Wie man sieht, sollte für das Wort acress das Wort actress ersetzt werden, es käme aber u.a. auch across infrage.

Wir schätzen die Bigramm-Wahrscheinlichkeiten auf Basis eines großen Korpus (Corpus of Contemporary American English) und erhalten

- P(actress|versatile) = 0,000021
- P(across|versatile) = 0,000021
- P(whose|across) = 0,000006
- P(whose|actress) = 0,0010

Wir können nun die Wahrscheinlichkeiten folgender Wortsequenzen schätzen, wobei wir die Wahrscheinlichkeit des ersten Wortes (das bei beiden Sequenzen gleich ist) weglassen:

- P(versatile actress whose) $\propto$ 0,000021 · 0,0010 = 210 · 10^{-10}
- P(versatile across whose) $\propto$ 0,000021 · 0,000006 = 1 · 10^{-10}

Die erste Sequenz ist also wesentlich wahrscheinlicher und wird daher als Korrektur ausgewählt.

Beispiel entnommen aus http://web.stanford.edu/class/cs276/handouts/spell_correction_4_16_2013.ppt

Phonetische Korrektur

Neben der eigentlichen Rechtschreibkorrektur spielt auch die **phonetische Korrektur**, d.h. die Korrektur von Fehlern, die aufgrund des gleichen Klangs zweier Schreibweisen entstehen, eine Rolle.

Dies ist auch wichtig, wenn die korrekte Schreibweise etwa eines Namens nicht klar ist, zum Beispiel Chebyshev vs. Tchebycheff vs. Tschebyschow (Erfinder der Tschebyscheff-Ungleichung).

Korrekturalgorithmen, die auf dem sog. **Phonetic Hashing** basieren, sind als **SOUNDEX-Algorithmen** bekannt geworden. Sie konvertieren jeden Term (in Dokumenten und in der Anfrage) in eine reduzierte Form aus vier Zeichen.

Terme, die ähnlich ausgesprochen werden, werden auf die gleiche reduzierte Form abgebildet.

Funktionsprinzip des SOUNDEX-Algorithmus

1. Der erste Buchstabe eines Terms bleibt unverändert, wird aber als Großbuchstabe übernommen.
2. Ersetze alle Vorkommen der folgenden Buchstaben mit 0 (Null): 'A', E', 'I', 'O', 'U', 'H', 'W', 'Y'.
3. Ersetze die übrigen Buchstaben nach folgendem Schema:
   - B, F, P, V $\rightarrow$ 1
   - C, G, J, K, Q, S, X, Z $\rightarrow$ 2
   - D,T $\rightarrow$ 3
   - L $\rightarrow$ 4
   - M, N $\rightarrow$ 5
   - R $\rightarrow$ 6
4. Ersetze alle Paare von gleichen aufeinanderfolgenden Ziffern.
5. Lösche alle Nullen aus dem Ergebnis.
6. Gib die ersten vier Stellen des Ergebnisses zurück, ggf. nach Auffüllen mit Nullen; das Ergebnis hat dann die Form **Großbuchstabe Ziffer Ziffer Ziffer**.

Beispiel: SOUNDEX-Algorithmus

Betrachten wir nochmals unseren russischen Mathematiker. Wir erhalten mit SOUNDEX

- Chebyshev $\rightarrow$ C121
- Tchebycheff $\rightarrow$ T121
- Tschebyschow $\rightarrow$ T121

Man sieht, dass das Verfahren bis auf den Anfangsbuchstaben gut funktioniert. Es gibt Varianten von SOUNDEX, die auch solche Fälle unterstützen.

SOUNDEX in Datenbanksystemen

Viele Datenbanksysteme unterstützen SOUNDEX-Ähnlichkeit in SQL-Anfragen. Hier ist ein Beispiel (aus der Oracle-Dokumentation):

```
SELECT last_name, first_name
FROM hr.employees
WHERE SOUNDEX(last_name)=SOUNDEX('SMYTHE');


LAST_NAME   FIRST_NAME
---------   ----------
Smith       Lindsey
Smith       William
```

Auch die anderen Konzepte, die wir kennengelernt haben (z.B. Stemming, Stopwörter, Phrasen, Proximity, Thesauri) findet man in den Textkomponenten vieler Datenbanksysteme.

Schwachpunkte des Booleschen IR-Modells

Wesentliche Schwachpunkte des elementaren Booleschen IR Modells sind:

- Boolesche Anfragen werden schnell recht **komplex**.
- Die Retrieval-Strategie basiert auf einer binären Entscheidung, lässt also **kein Ranking** zu.

Es gibt Erweiterungen des Booleschen Modells, die manche der Schwachpunkte relativieren oder neue Funktionalität hinzufügen.