Multimedia-Datenbanken im SS 2010 "Effiziente Algorithmen und Datenstrukturen II"

Dr.-Ing. Marcin Grzegorzek

05.07.2010

Inhalte und Termine

1. Einführung in MMDB

1.1 Grundlegende Begriffe

1.2 Suche in einem MMDBS

1.3 MMDBMS-Anwendungen 27.04.2010

2. Prinzipien des Information Retrievals

2.1 Einführung

2.2 Information-Retrieval-Modelle

2.3 Relevance Feedback

2.4 Bewertung von Retrieval-Systemen

2.5 Nutzerprofile 03.05.2010

Inhalte und Termine

3. Einführung in Multimedia-Retrieval

3.1 Besonderheiten der Verwaltung und des Retrievals	
3.2 Ablauf des Multimedia-Information-Retrievals	
3.3 Daten eines Multimedia-Retrieval-Systems	
3.4 Feature	
3.5 Eignung verschiedener Retrieval-Modelle	
3.6 Multimedia-Ähnlichkeitsmodell	10.05.2010

4. Feature-Transformationsverfahren

4.1 Diskrete Fourier-Transformation	11.05.2010
4.2 Diskrete Wavelet-Transformation	17.05.2010
4.3 Karhunen-Loeve-Transformation	
4.4 Latent Semantic Indexing und Singulärwertzerlegung	31.05.2010

Inhalte und Termine

5. Distanzfunktionen

5.1 Eigenschaften und Klassifikation	
5.2 Distanzfunktionen auf Punkten	07.06.2010
5.3 Distanzfunktionen auf Binärdaten	
5.4 Distanzfunktionen auf Sequenzen	
5.5 Distanzfunktionen auf allgemeinen Mengen	08.06.2010

6. Ähnlichkeitsmaße

6.1 Einführung	
6.2 Distanz versus Ähnlichkeit	
6.3 Grenzen von Ähnlichkeitsmaßen	21.06.2010
6.4 Konkrete Ähnlichkeitsmaße	
6.5 Aggregation von Ähnlichkeitswerten	
6.6 Umwandlung von Distanzen in Ähnlichkeitswerte und Normierung	
6.7 Partielle Ähnlichkeit	22.06.2010

Inhalte und Termine

7. Effiziente Algorithmen und Datenstrukturen

7.1 Hochdimensionale Indexstrukturen	28.06.2010
7.2 Algorithmen zur Aggregation von Ähnlichkeitswerten	05.07.2010

8. Anfragebehandlung

8.1 Einführung

8.2 Konzepte der Anfragebehandlung

8.3 Datenbankmodell

8.4 Sprachen

9. Zusammenfassung

Klausur Multimedia-Datenbanken

- Dienstag, den 13.07.2010, 14:00 Uhr, Raum F314
- Anmeldung über KLIPS (KLIPS-Nr.: 403510071314) bis zum 12.07 (obwohl Belegfrist bis zum 20.07)

Overview

Algorithmen zur Aggregation von Ähnlichkeitswerten

Overview

Algorithmen zur Aggregation von Ähnlichkeitswerten

Aggregation - Einführendes Beispiele

- Gesucht sind alle Bilder, die zu einem vorgegebenen Photo bezüglich Farbverteilung und Textur ähnlich sind. Für jedes Ergebnisbild müssen also zwei Ähnlichkeitswerte anhand einer geeigneten Aggregatfunktion kombiniert werden.

- Gesucht sind alle Bilder, die zu mehreren Anfragebildern ähnlich sind. In diesem Fall müssen pro Ergebnisbild mehrere Ähnlichkeitswerte aggregiert werden.

Anforderungen

An eine Aggregation agg, die Ähnlichkeitswerte für ein Objekt aggregiert, werden bestimmte Forderungen gestellt:

1. *Ähnlichkeitswerte*. Die Funktion muss mehrere Ähnlichkeitswerte aus dem Intervall $[0, 1]$ auf einen Wert aus dem Intervall $[0, 1]$ abbilden:

$$\mathrm{agg} : [0, 1]^n \rightarrow [0, 1]$$

2. *Monotonie.* Wenn die Eingangswerte nicht sinken, dann sinkt auch der aggregierte Ähnlichkeitswert nicht:

$$x_1 \leq y_1 \wedge \ldots \wedge x_n \leq y_n \Rightarrow \mathrm{agg}(x_1, \ldots, x_n) \leq \mathrm{agg}(y_1, \ldots, y_n)$$

Anforderungen

3. *Strikte Monotonie*. Wenn alle Eingangswerte wachsen, dann muss auch der entsprechende, aggregierte Ähnlichkeitswert wachsen:

   $$x_1 < y_1 \wedge \ldots \wedge x_n < y_n \Rightarrow \mathrm{agg}(x_1, \ldots, x_n) < \mathrm{agg}(y_1, \ldots, y_n)$$

4. *Stetigkeit*. Die Aggregatfunktion soll bezüglich der Eingangswerte stetig sein, also keine abrupten Sprünge aufweisen.

Anforderungen

5. *Idempotenz*. Eine Aggregation derselben Werte muss diesen Wert selbst ergeben:

$$\mathrm{agg}(a, \ldots, a) = a$$

6. *Unabhängigkeit von der Reihenfolge*. Das Resultat einer Aggregation ist unabhängig von der Reihenfolge der zu aggregierenden Ähnlichkeitswerte:

$$\mathrm{agg}(x_1, x_2, \ldots, x_n) = \mathrm{agg}(x_{p_1}, x_{p_2}, \ldots, x_{p_n})$$

wobei $[p_i]$ eine beliebige Permutation der Werte $[i]$ darstellt.

Generalisiertes Mittel

- Das generalisierte Mittel ist definiert wie folgt:

$$\mathrm{agg}^{\alpha}_{gm}(x_1, \ldots, x_n) = \left(\frac{x_1^{\alpha} + \ldots + x_n^{\alpha}}{n} \right)^{\frac{1}{\alpha}}$$

- Der Parameterwert α muss ungleich 0 sein.

- Folgende Spezialfälle ergeben sich:
  - $\alpha = 1$: arithmetisches Mittel
  - $\alpha = \infty$: maximaler Ähnlichkeitswert
  - $\alpha = -\infty$: minimaler Ähnlichkeitswert

Klassifikation

1. *Combiner-Algorithmen*: Ausgangspunkt dieser Algorithmen sind mehrere, nach Ähnlichkeitswerten absteigend sortierte Objektlisten, die anhand einer Aggregatfunktion zu einer solchen sortierten Liste vereinigt werden sollen.

2. *Kondensator-Algorithmen*: Ausgangspunkt für diese Algorithmen ist eine Liste von Objekten, bei der mehrere Listenobjekte zu jeweils einem neuen Listenobjekt aggregiert werden.

3. *Indexaggregation*: Bei diesen Algorithmen existieren keine Eingangslisten. Statt dessen wird innerhalb einer Indexstruktur die Aggregation ausgeführt, bevor eine sortierte Liste erstellt wird.

Combiner-Algorithmen

- Jeder Listeneintrag `e` enthält einen eindeutigen Identifikator `e.id` eines Medienobjekts zusammen mit einem Ähnlichkeitswerte `e.grade`.

- Wir beschränken uns der Einfachheit halber auf zwei Eingangslisten: `links` und `rechts`. Ein Objekt `o` hat einen Eintrag `o.lgrade` für den Ähnlichkeitswert der linken Liste und `o.rgrade` der rechten Liste.

- Der aggregierte Ähnlichkeitswert wird mit `o.grade` und der Identifikator mit `o.id` notiert. Einträge, denen noch kein Wert zugewiesen wurde, besitzen den Default-Wert `NULL`.

Combiner-Algorithmen

- Auf die Listenelemente kann in zwei verschiedenen Modi zugegriffen werden. Zum einen ist ein sequentieller Zugriff, also jeweils ein Zugriff auf den Beginn der Liste, möglich. Zusätzlich kann auf ein Medienobjekt über den dazugehörenden Identifikator direkt zugegriffen werden.

- Im Folgenden werden die beiden Zugriffsmodi mit *sequentiellem Zugriff* und mit *randomisiertem Zugriff* bezeichnet. Für die Zugriffe werden die Funktionen `getNext()` und `random()` verwendet.

Combiner-Algorithmen

- Die Ähnlichkeitswerte eines Medienobjekts aus den verschiedenen Listen sollen anhand einer monotonen Aggregatfunktion agg zu einem neuen Ähnlichkeitswert zusammengefasst werden.
- Von allen Ergebnisobjekten soll auf `k` Objekte mit den größten, aggregierten Ähnlichkeitswerten effizient zugegriffen werden können.
- Alternativ wird of auch ein sequentieller Zugriff auf die Objekte in der absteigenden Reihenfolge der aggregierten Ähnlichkeitswerte verlangt.
- Die beiden Arten der Ergebniszugriffe werden mit `top-k`- und mit `ranking`-Zugriff bezeichnet.

Combiner-Algorithmen

- TA-Algorithmus
- NRA-Algorithmus
- Stream-Combine-Algorithmus

Combiner-Algorithmen: TA-Algorithmus


```
01 procedure TA(liste links,liste rechts,funktion agg,liste top-k)
02   eintrag ol, or
03   real tau
04   repeat
05     ol = getNext(links)
06     or = getNext(rechts)
07     ol.rgrade = random(rechts, ol.id)
08     or.lgrade = random(links, or.id)
09     ol.grade = agg(ol.lgrade, ol.rgrade)
10     or.grade = agg(or.lgrade, or.rgrade)
11     aktualisiere top − k bzgl. ol und or
12     tau = agg(ol.lgrade, or.rgrade)
13   until |top − k| = k and ∀o ∈ top − k : o.grade ≥ tau
14   sortieretop − k − Elementenacho.grade
15 end procedure
```

Combiner-Algorithmen: TA-Algorithmus

Beispiel

links		**rechts**		**tau**	**top-k**	
id	**lgrade**	**id**	**rgrade**	**tau**	**id**	**lgrade+rgrade**
o3	0,9	o4	0,8	1,7	o4	1,4
o1	0,7	o2	0,7	1,4	o1	1,3
o4	0,6	o1	0,6	**1,2**	–	–
o2	0,2	o5	0,4	0,6	–	–
o5	0,1	o3	0,1	0,2	–	–