technische universität
dortmund

Item Sets that Compress

Arno Siebes Jilles Vreeken Matthijs van Leeuwen

Niels Ackermann

TU Dortmund
Fakultät für Informatik
Lehrstuhl 8

28.04.2009

Grundlagen

- Verkaufsgeschäft
  - Viele Transaktionen
  - Riesige Datenbank
  - Viele Daten, kaum Informationen
- Itemsets
  $\{\{I_1\}\{I_1, I_2\}\{I_1, I_2, I_3\}\{I_1, I_2, I_3, I_4\}\}$

Das Ziel

- Normalerweise: Finden von frequent item sets
- Problem: Redundanzen der Item Sets $\{\{I_1\}\{I_1, I_2\}, ...\}$
- **Ziel**: Kompression der Datenbank *db*

Die Idee

Item Sets sind interessant:

- gute Beschreibung der *db*
- somit gute Kompression der *db*
- ~~gute Kompression der frequent item sets~~

Die Konkurrenz: Closed Item Sets

- Closed Item Sets reduzieren Item Sets

Closed Item Set

$\nexists$ item set J:

- $I \subset J$ und
- $supp_{db}(I) = supp_{db}(J)$

Beispiel

- Item Sets: $\{\{I_1 : 3\}\underbrace{\{I_1, I_2 : 2\}\{I_1, I_2, I_3 : 2\}}\{I_1, I_2, I_3, I_4 : 1\}\}$
- Closed Item Sets:$\{\{I_1 : 3\}\{I_1, I_2, I_3 : 2\}\{I_1, I_2, I_3, I_4 : 1\}\}$

Kompression

Kompression

- Reduktion des Speicherbedarfs
- hier: verlustfrei

Kompression

1. von item sets mit item sets
2. Codierung in Bits

Kolmogorow Komplexität

- Darzustellende Zeichenkette
- Programm erzeugt Zeichenkette
- Kolmogorow Komplexität: Länge des kürzesten Programms

$\longrightarrow$ Strukturiertheit einer Zeichenkette
$\longrightarrow$ Kompression, wenn Kolmogorow Komplexität $<$ Länge Zeichenkette

Beispiel

tausend 0en

```
1 Program Nullfolge(int n){
2    for(int i=0; i<n; i++){
3       System.out.println("0");
4 }
```

Minimum Despription Length 1/3

- Modell $H \in \mathcal{H}$ kodiert Menge von Beispielen mit eindeutigem Präfix
- $\mathcal{H}$ Hypothese

Beispiel

Nachricht	n1	n2	n3	n4
Code	0	10	110	111

10001111100100 $\rightarrow$ 10 0 0 111 110 0 10 0
$\rightarrow$ n2 n1 n1 n4 n3 n1 n2 n1

Minimum Despription Length 2/3

Beispielcode ist optimal für W'keit(n1)= 1/2, W'keit(n2)=1/4, W'keit(n3)=1/8,W'keit (n4)=1/8

Shannon Entropie

Wähle Länge L(i) von n_i:

$$L(i) = -log_2 W'keit(n_i)$$

so ist

$$\text{Länge} \geq -\sum W'keit(n_i) log_2(W'keit(n_i))$$

Minimum Despription Length 3/3

Wähle das Modell mit der kürzesten Länge!

Güte der Kompression

$L(H) + L(D|H) \rightarrow \min$

- L(H): Länge der Beschreibung
- L(D|H): Länge der Daten D codiert mit H

Coding Item Set C

- besteht aus Item Sets
- Singleton Items enthalten
  $\longrightarrow$ alles codierbar
- Reihenfolge bestimmt Wahl beim codieren

Coding Item Set - Bedingung

- gültige Kodierung beschreibt jede Transaktion
- Überdeckung ist überschneidungsfrei

C(t) covers t:

for each t $\in db$ $\exists$ subset C(t) $\subseteq$ C:

1. $t = \bigcup_{c_i \in C(t)} c_i$
2. $\forall c_i, c_j \in C(t) : c_i \neq c_j \rightarrow c_i \cap c_j = \emptyset$

Codierungsschema CS

Paar (C,S):

- C: item set cover von db
- S: $db \rightarrow \mathcal{P}(C)$ mit S(t) covers t

Überdeckung

- Transaktion $\xrightarrow{Cover(C,t)}$ Code

Cover(C, t)

```
1 S := smallest element c of C in coding order for which c ⊆ t
2 if t \ S = ∅
3       then Res := {S}
4       else Res:= {S} ∪ Cover(C, t \ S)
5 return Res
```

Länge einer codierten Datenbank

- Häufigkeit eines Elements: $freq(c) = |\{t \in db \mid c \in S(t)\}|$
- Codelänge: $L(d) = -log(P(d))$
- W'keitsverteilung des Codes: $\forall c \in C : P(c) = \frac{freq(c)}{\sum_{d \in C} freq(d)}$

Grösse der Datenbank

$$L_{(C,S)}(db) = -\sum_{c \in C} freq(c) log(\frac{freq(c)}{\sum_{d \in C} freq(d)})$$

Beispiel: Kompression

db :

Transaktion	A	B	C	D	Item Set	Komprimiert
T_{100}	1	1	1	1	{A,B,C,D}	{X}
T_{200}	1	1	0	1	{A,B,D}	{Y}
T_{300}	1	0	1	0	{A,C}	{A,C}
T_{400}	1	0	0	1	{A,D}	{A,D}
T_{500}	1	0	0	0	{A}	{A}
T_{600}	0	0	0	1	{D}	{D}

- X:={A,B,C,D},Y:={A,B,D}
- $L_{(C,S)}(db) = 14,5$

Kompression

- **Naive Kompression**

Sortierung eines Codesets

- $\mathcal{I}$: singleton Items
- J: Item Sets

Standard($\mathcal{I}$, db)

- absteigend sortiert nach Häufigkeit

Cover-Order(J,db)

1. sortiere nach Länge
   - Längere nach vorne
2. sortiere nach Häufigkeit
   - Häufige nach vorne

Naive Kompression - Idee

- Greedy-Ansatz

Algo

1. Nimm Item Set: höchste Frequenz
   (wenn nicht eindeutig: Item Set: größte Menge)
2. Item Set $\rightarrow$ Code Set (nach Cover-Order)
3. Länge$_{neu}$ <Länge$_{alt}$?
   - ja: behalte
   - nein: weg

Naiver Kompressions Algorithmus

Naive-Compression($\mathcal{I}$, J, db)

1 CodeSet := Standard($\mathcal{I}$, db)
2 J := J $\setminus \mathcal{I}$
3 CanItems := Cover-Order(J, db)
4 while CanItems $\neq \emptyset$ do
5 cand := maximal element of CanItems
6 CanItems := CanItems $\setminus$ {cand}
7 CanCodeSet := CodeSet $\oplus$ {cand}
8 if $L_{CanCodeSet}(db) < L_{CodeSet}(db)$
9 then CodeSet := CanCodeSet
10 return CodeSet

Beispiel: Naive Kompression

- $CodeSet_{Start} = \{\{A\},\{D\},\{B\},\{C\}\}$
- CanItems := {{A,B,C,D},{A,B,D},...{A,D},{A,B},...,{C,D}}
- cand = {A,D} (höchste Frequenz: 3)
- prüfe ob CodeSet $\oplus$ {A,D} geringere Länge hat
  $L_{CanCodeSet}(db) = 22,3 < L_{CodeSet}(db) = 24,5$
  $\rightarrow$ Behalten von {A,D}
- nächster Kandidat: {A,B,D} (größte Menge mit Frequenz 2)

Schwächen des naiven Ansatzes

Beispiel

- $CS_1 = \{\{I_1, I_2\}, \{I_1\}, \{I_2\}, \{I_3\}\}$
- $CS_2 = \{\{I_1, I_2, I_3\}, \{I_1, I_2\}, \{I_1\}, \{I_2\}, \{I_3\}\}$
- $CS_3 = \{\{I_1, I_2, I_3\}, \{I_1\}, \{I_2\}, \{I_3\}\}$

- Annahme: $supp(\{I_1, I_2, I_3\}) = supp(\{I_1, I_2\}) - 1$
- Möglichkeit
  - $L_{CS_2}(db) > L_{CS_1}(db) > L_{CS_3}(db)$

Kompression

- **Pruning**

Lösung: Pruning

Pruning

1. Entfernen : Item Set
2. Prüfung : Bessere Kompression?
3. Enscheidung: Erhalt des Item Sets, oder nicht.

Prune Algorithmus

Prune-on-the-fly(CanCodeSet,CodeSet, db)

1 PruneSet :=
$\{J \in \text{CodeSet} \mid cover_{CanCodeSet}(J) < cover_{CodeSet}(J)\}$
2 PruneSet := Standard(PruneSet, db)
3 while PruneSet $\neq \emptyset$ do
4 cand := element of PruneSet with minimal cover
5 PruneSet := PruneSet $\setminus$ cand
6 PosCodeSet := CodeSet $\ominus$ cand
7 if $L_{PosCodeSet}(db) < L_{CanCodeSet}(db)$
8 then CanCodeSet := PosCodeSet
9 return CanCodeSet

Beispiel: Prune Algorithmus

- Menge nach zweitem Naive-Kompression-Schritt:
  $CanCodeSet = \{\{A, B, D\}, \{A, D\}, \{A\}, \{D\}, \{B\}, \{C\}\}$
  $CodeSet = \{\{A, D\}, \{A\}, \{D\}, \{B\}, \{C\}\}$
- $Prunset := \{\{A, D\}\}$
- $cand = \{A, D\}$
- $PosCodeSet = \{\{A, B, D\}, \{A\}, \{D\}, \{B\}, \{C\}\}$
- $L_{PosCodeSet}(db) = 17,7 < L_{CanCodeSet}(db) = 18$
  $\rightarrow$ Entfernen von {A,D}

Kompression

- **Noise Reduktion**

Noise

- Datenbank enthält ungewöhnliche Transaktionen
  - "Rauschen"
  - Unregelmäßigkeit in der Datenbank

ungewöhnliche Transaktion

- Standardcode $<$ Code nach Kompression
- $L_S(t) < L_{CS}(t)$

- Entfernen der Unregelmäßigkeiten

Noise($\mathcal{I}$,CodeSet, db)

1. Noise := $\emptyset$
2. foreach t $\in$ db do
3. if $L_S(t) < L_{CS}(t)$ then
4. Noise := Noise$\cup${t}
5. return Noise

Denoise($\mathcal{I}$,CodeSet, db)

1. Noise := Noise($\mathcal{I}$,CodeSet, db)
2. db := db $\setminus$ Noise
3. return db

Sanitize($\mathcal{I}$,CodeSet, db)

1. db := Denoise($\mathcal{I}$,CodeSet, db)
2. foreach J $\in$ CodeSet do
3. if $cover_{db}(J) = \emptyset$ then
4. CodeSet := CodeSet $\ominus$ J
5. return CodeSet

Experimente

Die Hypothese

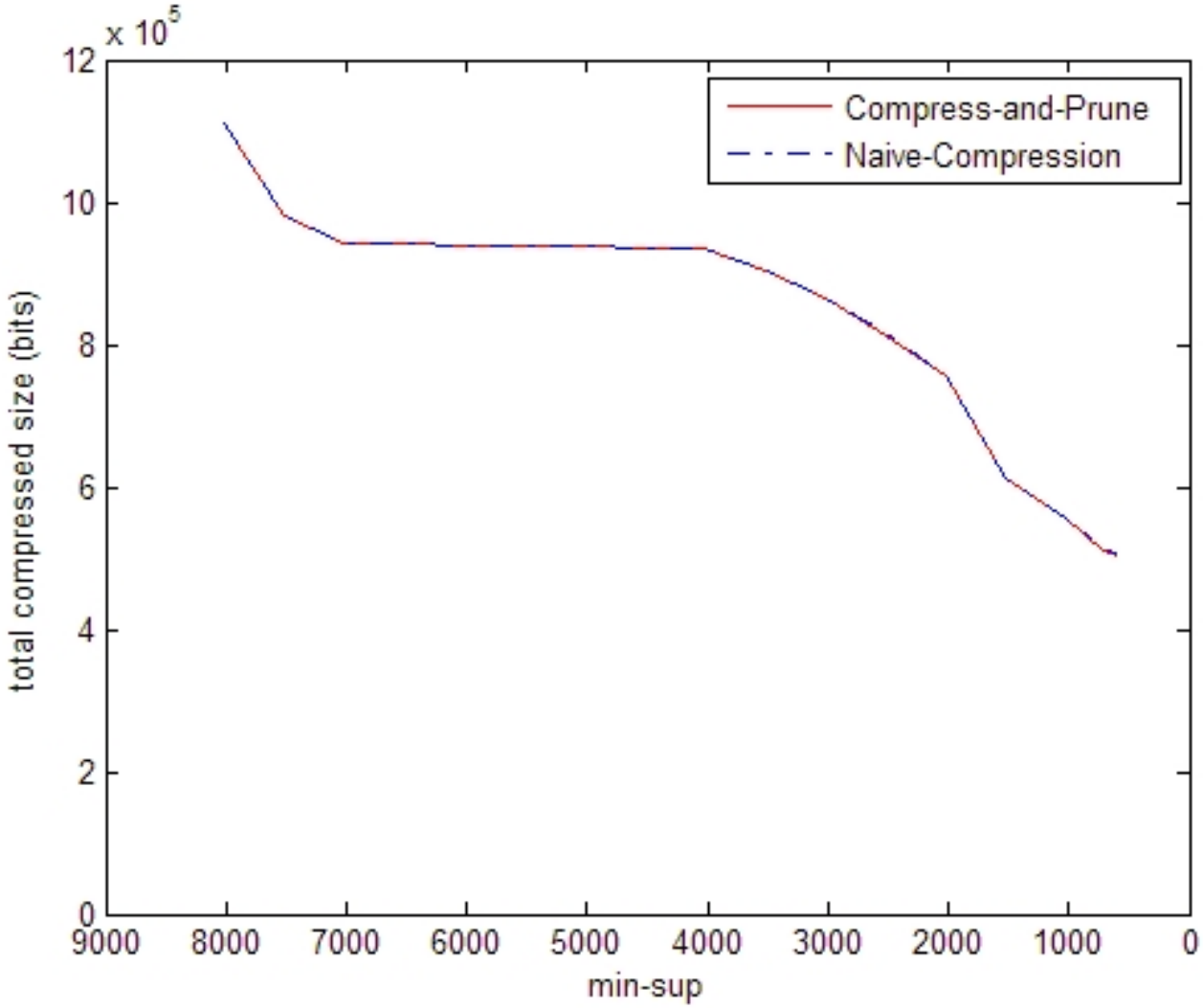
- Algorithmen reduzieren Anzahl der Item Sets
- Anzahl der Item Sets wird deutlich geringer

- Vergleich: Closed Item Sets
- Bester Algorithmus?

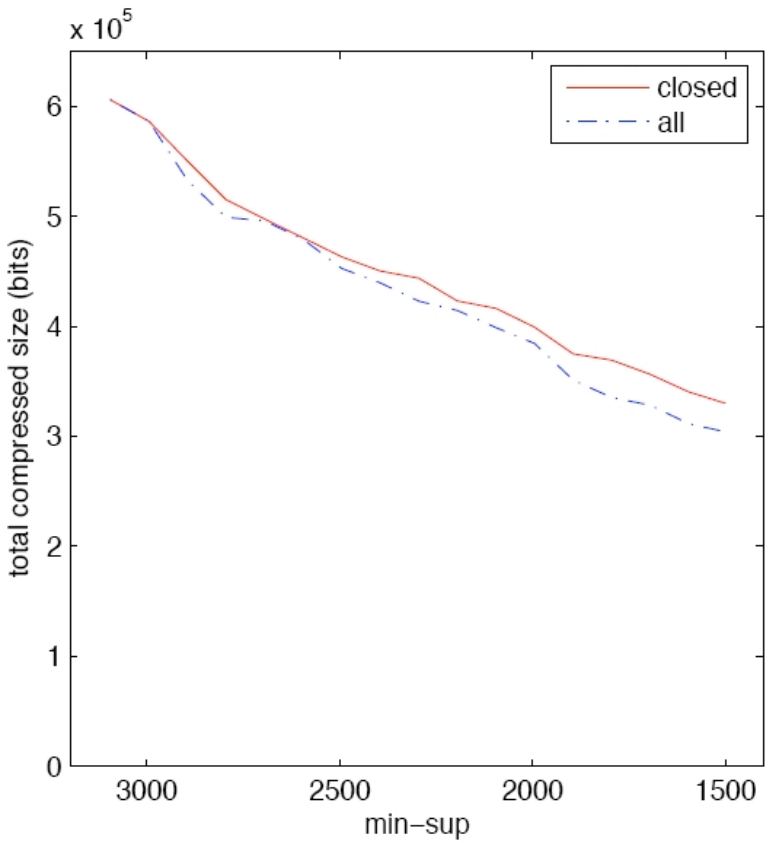
Die Experimente

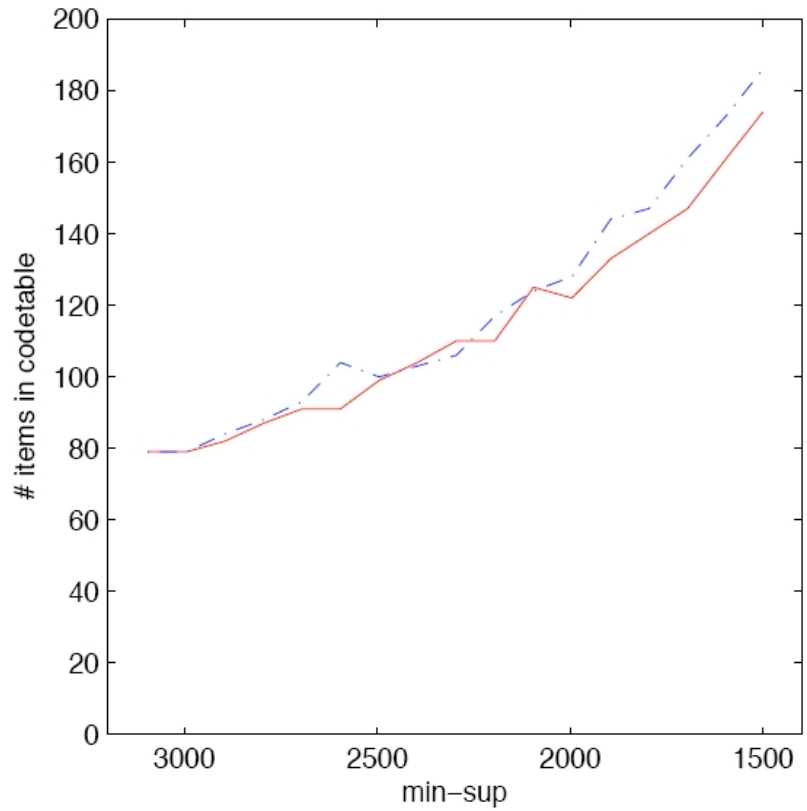
- alle Kombinationen
  - Naive Kompression
  - Naive Kompression & Pruning (nach jedem Schritt)
  - Naive Kompression + Sanitize (anschließend)
  - Naive Kompression & Pruning + Sanitize
- Testdatensätze: Pilze, Schach

Pruning


x 10^5
12
10
8
6
4
2
0
total compressed size (bits)
Compress-and-Prune
Naive-Compression
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
min-sup

Closed Item Sets VS All Item Sets





Ergebnisse

Source	Algo	C-Items	C-Table	Db	Compress
All	Naive	338	9901	425977	435878
All	Naive+Sanit	338	9901	425977	435878
All	Naive&Prune	282	8252	423487	431739
All	All	282	8252	423487	431739
Closed	Naive	149	4754	507691	512445
Closed	Naive+Sanit	149	4754	507691	512445
Closed	Naive&Prune	147	4631	507633	512246
Closed	All	147	4631	507633	512246

Zusammenfassung

- Naive Kompression erfolgreich
  - mehr Kompression mit allen Item Sets
- Pruning erfolgreich
- Sanitize unerfolgreich

Vielen Dank für Ihre Aufmerksamkeit

Gibt es noch Fragen?