technische universität
dortmund

Mining conjunctive sequential patterns

Chedy Raïssi • Toon Calders • Pascal Poncelet

Manh Linh Nguyen
TU Dortmund
Fakultät für Informatik
Lehrstuhl 8

10.05.2009

Inhalt

1. Einleitung
2. Mining von frequent sequences
3. Verdichtete Repräsentation
4. Äquivalenzklassen von Sequenzen
5. Conjunctive sequential patterns
6. CSP Miner Algorithmus
7. Experimente

Einleitung

- Aufgabe: Finden von frequent sequences
- Problem:
  - Menge der gefundenen frequent sequences zu groß
  - viele Redundanzen
- Lösungsvorschlag: Mining von verdichteter Repräsentation

Mining von frequent sequences 1/4

- Endliche Menge von Items $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$
- S = $\langle it_1, it_2, \ldots, it_m \rangle$ heißt Sequenz über $\mathcal{I}$ wenn:
  - $\langle it_1, it_2, \ldots, it_m \rangle$ ist geordnet
  - it_j ist itemset über $\mathcal{I}$
- Beispiel: $\mathcal{I} = \{a, b, c, d\}$, S = $\langle (ab)(c)(ab)(de) \rangle$

Mining von frequent sequences 2/4

- $\mathcal{T}(\mathcal{I})$ ist Menge aller möglichen Sequenzen über $\mathcal{I}$.
- (SID, T) mit SID $\in$ {1, 2, …} als Identifikator und T $\in \mathcal{T}(\mathcal{I})$ heißt Transaktion.
- Beispiel: *(1, ‹(a)(b)›), (2, ‹(ab)(c)›)*

- Eine endliche Menge $\mathcal{D}$ von (SID, T) heißt Datenbank von Sequenzen über $\mathcal{I}$.
- Es gilt:

$$(SID_1, T) \neq (SID_2, T) \in \mathcal{D} \rightarrow SID_1 \neq SID_2$$

$\mathcal{D}$	S_1	*(a, b, c, d)(a, c)*
	S_2	*(a, d)(c)*

Mining von frequent sequences 3/4

- **Inklusion:**

  S´ = ‹$is´_1, is´_2, \ldots, is´_n$› heißt *Subsequenz* von S = ‹$is_1, is_2, \ldots, is_m$›

  (S´ ≤ S) wenn $i_1 < i_2 < , \ldots , < i_n$ existieren, für die gilt:

  $$is´_1 \subseteq is_{i1}, is´_2 \subseteq is_{i2}, \ldots, is´_n \subseteq is_{in}$$

- Beispiel: ‹*(a)(c)(d)*› ist Subsequenz von ‹*(ab)(c)(ab)(de)*›

  aber ‹*(a)(c)*› ist keine Subsequenz von ‹*(c)(a)*›

- **Support:**
  - *Support(S,D)* = $|\{(SID, T) \in D | S \leq T\}|$
  - *Häufigkeit* von S in *D*: $f_s^D = \frac{Support(S,D)}{|D|}$

Mining von frequent sequences 4/4

- Aufgabe: Sei eine Datenbank $\mathcal{D}$ und Grenzwert σ gegeben.
  Extrahiere alle S in $\mathcal{D}$ mit: $f_s^{\mathcal{D}} \geq \sigma$
  Also die Menge $FSeqs(\mathcal{D}, \sigma) := \{S \mid f_s^{\mathcal{D}} \geq \sigma\}$.

- Beispiel: $\mathcal{I} = \{a, b, c, d\}$, $\sigma = ⅔$
  Ergebnis: $FSeqs(\mathcal{D}, \sigma) =$
  $\{\langle(a)\rangle, \langle(b)\rangle, \langle(c)\rangle, \langle(d)\rangle, \langle(ab)\rangle, \langle(ad)\rangle, \langle(a)(c)\rangle, \langle(d)(c)\rangle\}$

$\mathcal{D}$	S_1	$(a, b, c, d)(a, c)$
	S_2	(a, b)
	S_3	$(a, d)(c)$

Verdichtete Repräsentation 1/3

Verdichtete Repräsentation:

- Subkollektion aller frequent itemsets/sequences.
- Enthält die vollständigen Informationen

Itemsets	Sequence patterns
Closed itemsets	Closed sequential patterns
Non-derivable itemsets *(Nicht-ableitbare Itemsets)*	**Conjunctive sequential patterns**

Verdichtete Repräsentation 2/3

- Sequenz S heißt *closed*, wenn:

  !∃ Supersequenz S´ von S mit: *Support(S)* = *Support(S´)*

- Nicht-ableitbare Itemsets:
  - Itemset I heißt *ableitbar*, wenn *Support(I)* mit Hilfe der *Supports* seiner Teilmengen vollständig bestimmt werden kann.
  - *Ableitbare* Itemsets als Redundanz von Informationen

Verdichtete Repräsentation 3/3

- Konstruktion von verdichteten Repräsentation: Ableitung der Häufigkeiten.

- Problem: Sei $\mathcal{C} = \{f_{S1}, f_{S2}, \ldots, f_{Sn} \in [0,1]\}$ *für* $S_1, \ldots, S_n$ gegeben.
- Frage: Sequenz S $!\in \{S_1, S_2, \ldots, S_n\}$. Welche Ableitung für f_S ?

- Eine Datenbank $\mathcal{D}$ heißt *konsistent* mit $\mathcal{C} \Leftrightarrow f_{Si}^{\mathcal{D}} = f_{Si}$
- Beste Grenzintervall für f_s :

$$[LB_C(S), UB_C(S)] := [\min\{f_S^{\mathcal{D}} \mid \mathcal{D} \textit{ konsistent} \text{ mit } \mathcal{C}\},$$
$$\max\{f_S^{\mathcal{D}} \mid \mathcal{D} \textit{ konsistent} \text{ mit } \mathcal{C}\}]$$

Äquivalenzklassen von Sequenzen 1/3

- **Definition**: Sei $\mathcal{S}$ eine Menge von Sequenzen.
  2 Sequenzen T_1 und T_2 heißt *$\mathcal{S}$-äquivalent* ($T_1 \equiv_S T_2$), wenn:
  $$\forall S \in \mathcal{S}: S \leq T_1 \Leftrightarrow S \leq T_2$$

- *IN, OUT* sind Mengen von Sequenzen:
  $\mathcal{E}(IN, OUT) := \{T \in \mathcal{T} \mid \forall S \in IN : S \leq T, \forall S \in OUT : S \not\leq T\}$

- Beispiel: $\mathcal{I} = \{a, b, c\}$, S = $\{\langle(a)(b)(c)\rangle, \langle(b)(c)\rangle, \langle(a)(b)\rangle\}$
  - $\langle(abc)(abc)(abc)\rangle \equiv_S \langle(a)(bc)(abc)\rangle$
  - *aber* $\langle(b)(c)\rangle \not\equiv_S \langle(a)(bc)\rangle$

Äquivalenzklassen von Sequenzen 2/3

Beispiel für $\mathcal{E}(IN, OUT) : I = \{a, b, c\}$

- $IN = \{\langle(a)(b)(c)\rangle, \langle(ac)\rangle\}, OUT = \{\langle(bc)\rangle\}$
  ➔ $\mathcal{E}(IN, OUT)$ enthält z. b. $\langle(ac)(b)(c)\rangle, \langle(ac)(c)(ab)(c)\rangle$
  aber nicht $\langle(abc)\rangle$.

- $IN = \{\langle(ab)(ac)\rangle\}, OUT = \{\langle(\ c)\rangle\}$ ➔ $\mathcal{E}(IN, OUT) = \emptyset$.

- $IN = \{\langle(a)(c)\ \rangle, \langle(b)\rangle\}, OUT = \{\{\langle(a)(b)\rangle, \langle(b)(c)\rangle\}$ ➔ $\mathcal{E}(IN, OUT) = \emptyset$.

Äquivalenzklassen von Sequenzen 3/3

- **Lemma 1**: Sei $\mathcal{S}$ eine Menge von Sequenzen.
  $\equiv_S$ ist eine Äquivalenzrelation auf Menge $\mathcal{T}$, dann: $|\mathcal{T} / \equiv_S| \geq 2^{|\mathcal{S}|}$.

  Für jede Transaction $T \in \mathcal{T}$ gilt:

  $$[T]_S = \mathcal{E}(\{S \in \mathcal{S} \mid S \leq T, S \in \mathcal{S} \mid S \not\leq T\})$$

- $\mathcal{E}(IN, OUT) = \emptyset \Leftrightarrow$ keine mögliche Erzeugung von Sequenzen:
  - support Sequenzen von IN
  - und support kein Sequenzen von OUT

- **Lemma 2**: Sei S eine nicht leere Sequenz, dann ist $\mathcal{E}(\{S' \mid S' < S\}, \{S' \mid S' \not< S\})$ nicht leer.

Untergrenze der Häufigkeit von Sequenzen

- **Theorem 1**: Sei S eine Sequenz, $\mathcal{C} = \{f_{S1}, f_{S2}, \ldots, f_{Sn} \in [0,1]\}$ *für* $S_1, \ldots, S_n$ mit $S_1, \ldots, S_n$ ist echte Subsequenz von S. Wenn eine Datenbank existiert, die *konsistent* mit $\mathcal{C}$, dann $LB_C(S) = 0$.

$D =$		
	S_1	*(a)(b)(c)*
	S_2	*(a)(b)(c)*
	S_3	*(c)(a)*
	S_4	(b)(c)

$D' =$		
	S_1	*(b)(c)(a)(c)(a)(b)*
	S_2	*(b)(c)(a)(c)(a)(b)*
	S_3	*(c)(a)*
	S_4	(b)(c)

- Folge: Sequenz S mit f_S von *0* ist ableitbar.

  Konzept von *nicht-ableitbaren Itemsets* nicht anwendbar für sequential patterns.

Conjunctive sequence patterns 1/5

- **Definition**: C heißt conjunctive sequence pattern (CSP), wenn:
  - $C \subseteq \mathcal{T}$
  - $S \neq S' \in C \rightarrow S$ und S' unvergleichbar: $S' \not\leq S$ und $S \not\leq S'$

- $\mathcal{C}$ ist Menge aller CSPs.
- $T \in \mathcal{T}$ heißt Support für ein CSP wenn $\forall S \in C: S \leq T$.
- $Support_D(C) := |\{(SID, T) \in D | T \text{ Support } C\}|$
- $C_1, C_2 \in \mathcal{C}$: C_1 heißt Teilmuster von C_2 ($C_1 \leq C_2$)

  $\Leftrightarrow \forall S_1 \in C_1: \exists S_2 \in C_2 : S_1 \leq S_2$

Conjunctive sequence patterns 2/5

- **Lemma 3**: (Anti-monotonie) Seien $C_1, C_2 \in \mathcal{C}$.
  Wenn $C_1 \leq C_2$, dann $\forall\ D$: $Support_D(C_1) \geq Support_D(C_2)$.

- **Beispiel**:
  $C_1 = \{\langle b\rangle, \langle (c)\rangle\}$; $C_2 = \{\langle (a)(b)\rangle, \langle (a, c)\rangle\}$; $C_3 = \{\langle (a)(b)\rangle, \langle (c)\rangle\}$

$D =$	S_1	$(a)(b)(a, c)$
	S_2	$(a, b)(c)$
	S_3	$(c)(a)$

- $C_1 \prec C_3 \prec C_2$
- $Support_D(C_1) = 2$, $Support_D(C_2) = 1$, $Support_D(C_3) = 1$,

Conjunctive sequence patterns 3/5

- Sei $P \subseteq \mathcal{C}$.
  - $\lceil P \rceil := \{S \in P \mid !\exists\, S' \in P : S' \prec S\}$ ist ein CSP
  - $\downarrow P := \{S \in \mathcal{T} \mid \exists\, S' \in P : S \leq S'\}$

- $C_1 \rightarrow C_2$ wenn $C_1 \leq C_2$
- $!\exists\, C_3 : C_1 \prec C_3 \prec C_2$ dann:
  - C_2 heißt direkte Spezialisierung von C_1
  - C_1 heißt direkte Verallgemeinerung von C_2

Conjunctive sequence patterns 4/5

- S = ‹$i_1, i_2, \ldots, i_m$›:

  Menge der direkten Verallgemeinerungen von S ist:

$$dg(S) := \bigcup_{\substack{j=1 \\ |I_j|>1}}^{n} \left\{ \left\langle I_1, \ldots, I_{j-1}, I_j \setminus \{i\}, I_{j+1}, \ldots, I_n \right\rangle \mid i \in I_j \right\} \cup \bigcup_{\substack{j=1 \\ |I_j|=1}}^{n} \left\{ \left\langle I_1, \ldots, I_{j-1}, I_{j+1}, \ldots, I_n \right\rangle \mid i \in I_j \right\}$$

- Beispiel: S = ‹*(a)(bc)…(a)(c)(bd)…(cd)*›

  S_1 = ‹*(a)(bc)…(a)(c)(d)…(cd)*›

  S_2 = ‹*(a)(bc)…(a)(bd)…(cd)*›

Conjunctive sequence patterns 5/5

- **Lemma 4**: Sei $C \in \mathcal{C}$.

  Menge der direkten Verallgemeinerungen von C ist:

$$\left\{ \left\lceil \left(C \setminus S\right) \cup \ dg(S) \right\rceil \mid S \in \ C \right\}$$

  und Menge der direkten Spezialisierungen von C ist:

$$\left\{ C \cup \left\{ S \right\} \mid S \notin \ C, dg(S) \subseteq \ C \right\}$$

- Erlaube die Erzeugung von Menge aller Muster, von allgemein bis spezifisch, unter Berücksichtigung der Anti-Monotonie von Support.

Begrenzung von Support und Möbius-Inversion 1/3

- **Theorem 2**: Die partielle Ordnung ($\mathcal{C}$, ≤) bildet einen Verband
  Also ∀ $C_1, C_2 \in \mathcal{C}$: ∃⌈↓C_1 ∩ ↓C_2⌉ und ∃⌈↓C_1 ∪ ↓C_2⌉

- s(C´)= *Support(C´, D)*
  und $a(C') = A(C') := |\{T \in D \mid \forall C'' \in \downarrow C : T \models C'' \Leftrightarrow C'' \leq C'\}|$
  Lemma 5: ∀ C´ ≤ C:

$$s(C') = \sum_{C' \lesssim C''} a(C'')$$

Begrenzung von Support und Möbius-Inversion 2/3

- **Lemma 6**: *Möbius-Inversion*
  ∀ *D* und C´, C´´ ≤ C: ∃ *μ*(• , •):

$$a(C') = \sum_{C' \leq C''} \mu(C', C'')\, s(C'')$$

- **Theorem 4**: ∀C´≤ C und ganze Zahl s_C:
  ∃ *D* mit ∀C´≤ C: *Support(C´, D)* = s_C

$$\sum_{C' \leq C''} \mu(C', C'')\, s(C'') = 0 \quad \forall C' : (\varepsilon(C', \downarrow C \setminus C') = \emptyset$$

$$\Leftrightarrow$$

$$\sum_{C' \leq C''} \mu(C', C'')\, s(C'') \geq 0 \quad \forall C' : (\varepsilon(C', \downarrow C \setminus C') \neq \emptyset$$

Begrenzung von Support und Möbius-Inversion 3/3

- **Beispiel**:

  $D = \{(S_1, \langle(a)(b)(c)\rangle)\}$, $\varepsilon = 1$

  $csp = \{\langle(a)\rangle; \langle(b)\rangle\} \rightarrow \quad P = \{ \{\langle\rangle\}, \{\langle(a)\rangle\}, \{\langle(b)\rangle\} \}$

  Verband auf $P \cup \{csp\}$

$$\varepsilon = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \mu = \begin{pmatrix} 1 & -1 & -1 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

1. $s(\{\langle\rangle\}) - s(\{\langle a\rangle\}) - s(\{\langle b\rangle\}) + s(\{\langle a\rangle , \langle b\rangle\}) \geq 0$
2. $s(\{\langle a\rangle\}) - s(\{\langle a\rangle , \langle b\rangle\}) \geq 0$
3. $s(\{\langle a\rangle\}) - s(\{\langle a\rangle , \langle b\rangle\}) \geq 0$
4. $s(\{\langle a\rangle , \langle b\rangle\}) \geq 0$

   $\rightarrow \quad s(\{\langle a\rangle , \langle b\rangle\}) = 1$

CSP Miner Algortihtmus

- Tiefensuche-Algorithmus
- Idee: Mischen von
  - Mining von Sequenzen
  - Erzeugung aller möglichen CSPs
- Eigenschaften:
  - Berechne Häufigkeiten von CSPs = Bestimme Kardinalzahl
  - Berechne Häufigkeitsintervall von CSP Y:
    $\{\langle\rangle\} \leq X \prec Y$ vorher bekannt → umgekehrte Reihenfolge

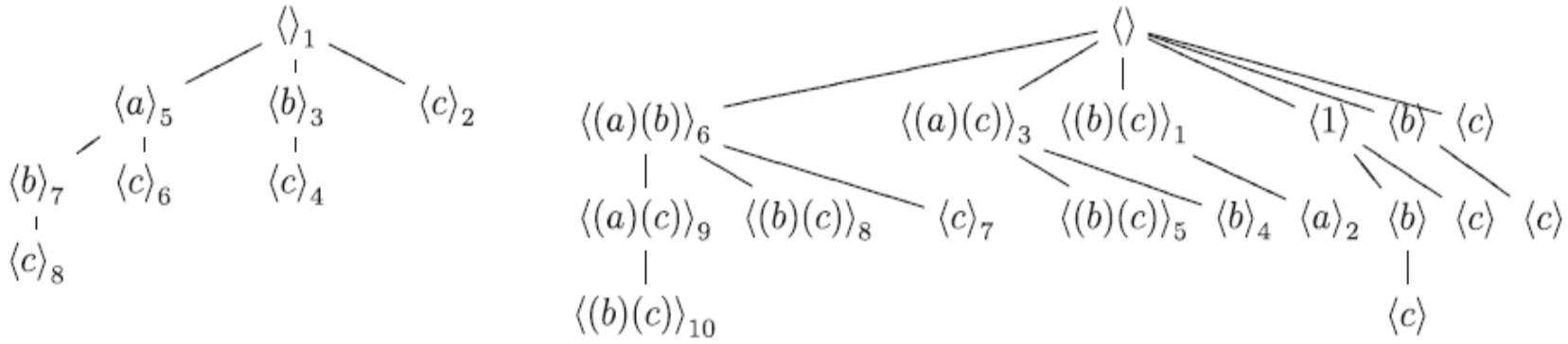
CSP Miner Algortihtmus

Daten: Datenbank von Sequenzen *D*; minimal Grenzwert der Häufigkeit ε

Ergebnis: Menge der CSPs : $\mathcal{C}$

1 begin
2 $k \leftarrow 1$;
3 $C \leftarrow \emptyset$;
4 $F_1 \leftarrow$ *mine_sequence(k)*;
5 $C \leftarrow F_1$;
6 **while** F_k *not empty* **do**
7 **foreach** $s \in F_k$ **do**
8 **foreach** $x <_{invprefix} s$ *with* $x \not\leq s$ **do**
9 $C \leftarrow$ *generate_CSP(x, s, C)*;
10 k++;
11 $F_k \leftarrow$ *mine_sequence(k)*;
12 return *C*;
13 end

Ablauf des CSP Miner Algorithmus für $D = \{(S_1, \langle(a)(b)(c)\rangle)\}$



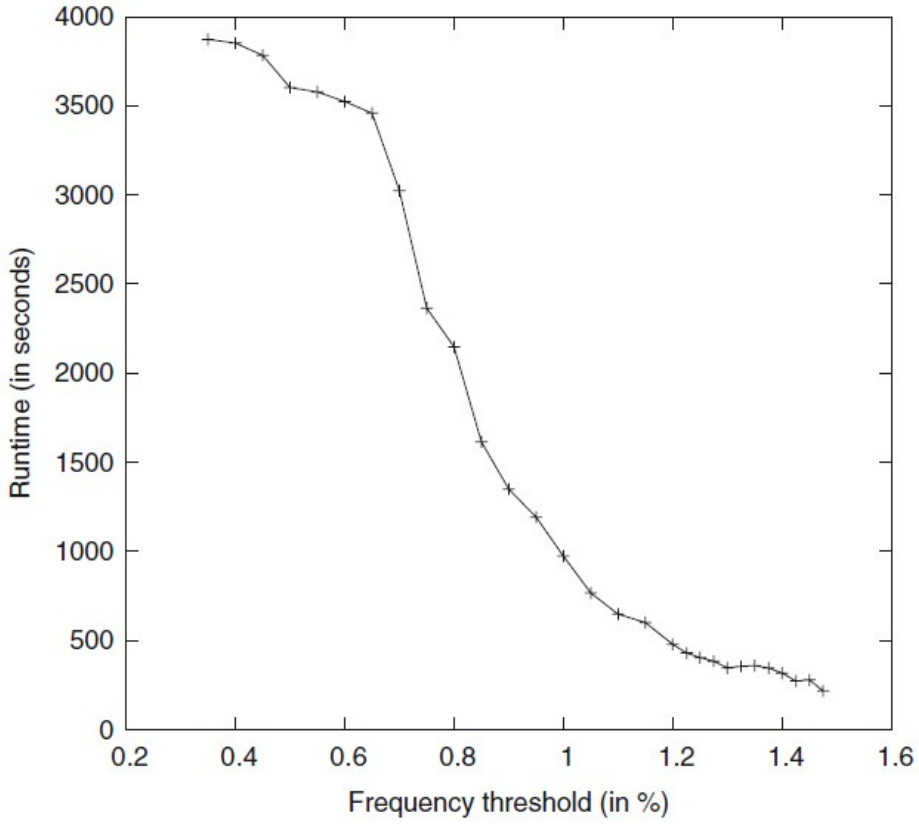
Erst für Sequenzen der Länge 1, dann der Länge 2

Experimente

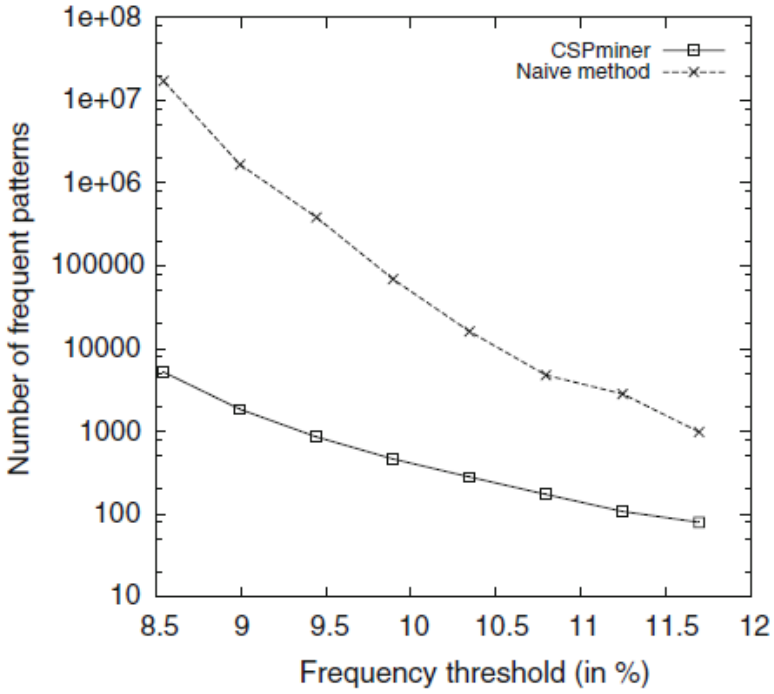
- Hardware: Core-Duo 2.16 GHz MacBook Pro mit 2Gb-RAM
- Software:
  - Betriebssystem: MAC OS x 10.5.2
  - Implementierung: C++ basiert auf DMTL Bibliothek und SPADE-Algorithmus
- Testdaten:

Data set	# sequences	# items	# it/trans.	# trans./sequence
$C_{200}T_{2,5}S_{10}I_{10k}$	200K	10K	2,5	10
UNIX User Data	11,116	2,016	1	39

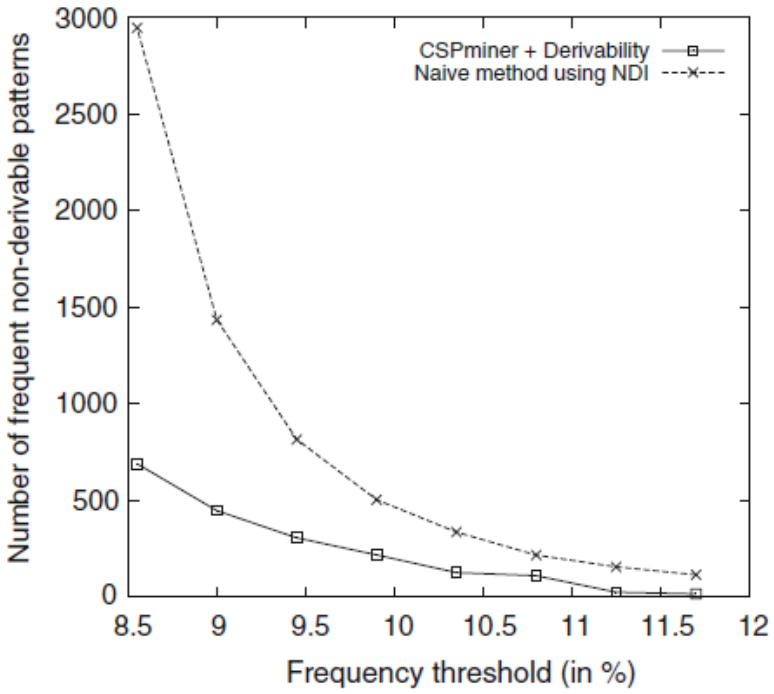
Ergebnis für $C_{200}T_{2,5}S_{10}I_{10k}$



Ergebnis für Unix User Data



(a) Frequent patterns



(b) Frequent non-derivable patterns

CSP Miner vs. Naive Methode bei Unix User Data

Frequency (%)	# CSP	# Conj.	# NDCSP	# NDconj.	CSP ratio (%)	Conj. ratio (%)
11.69	80	984	16	115	20	88.31
11.24	108	2,840	25	156	23.14	94.50
10.79	174	4,784	110	217	36.78	95.46
10.34	282	16,146	126	336	44.68	97.91
9.89	466	68,991	218	503	53.21	99.27
9.44	858	385,107	306	816	64.29	99.78
8.99	1,856	1,677,387	447	1,435	75.92	99.91
8.54	5,211	17,127,924	690	2,947	86.75	99.98

Danke für die Aufmerksamkeit!

Gibt es noch Fragen?