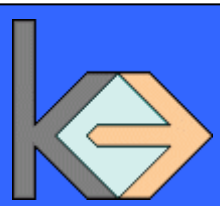
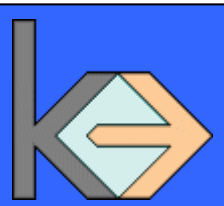


Efficient Substructure Discovery from Large Semi-structured Data

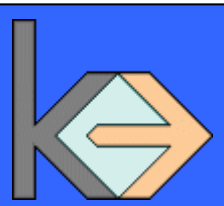
FREQT

Content



- Motivation
- Basic Definitionts
- Tree Matching
- Tree Enumeration
- Occurrence Lists
- Pruning and Experimental Results

Motivation



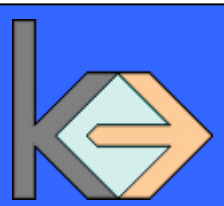
- Größe Menge von semi-structured Data:

Web Pages, XML Datei

- Gesucht ist : rules, patterns
- Modell: labeled ordered trees
- Frequency von subtrees in ein Wald

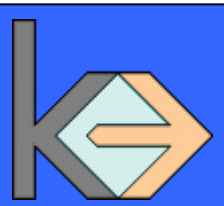
- => Freqt

Definitionen



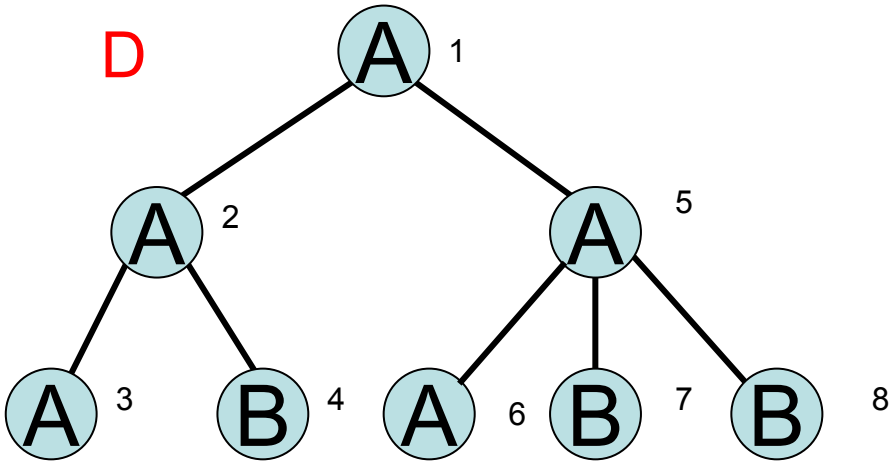
- Labeled Ordered Tree $T=(V,E,L,L,v_0, \prec)$
  - L ist Alphabet für labels
  - L ist Belegung von Labels zu Knoten
  - $\prec$ ist die Kinderordnungsrelation
  - V,E, v_0 sind die übrige Notationen für Knoten- und Kantenmenge und für Root

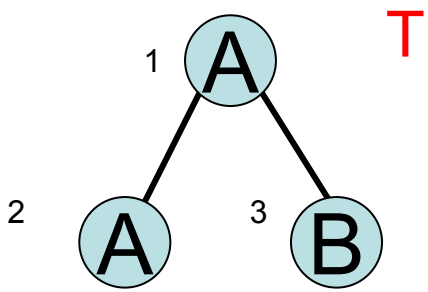
NF und Tree Matching



- Normalform: die pre-order Travesierung- v_0 ist 1 und der rechteste Knot v_{k-1} ist k
- Matchfunktion: für pattern tree T, data tree D und Knoten v ϵ T und v´ ϵ D existiert ein MF, wenn:
  1. die Elternrelation ist erhalten
  2. die Geschwisterrelation ist erhalten
  3. v und v´ haben gleiches Label

Frequency



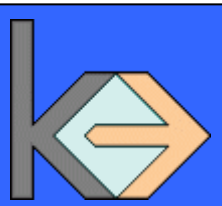


Pattern Tree T can be matched to subtrees in D : (2,3,4) , (5,6,7) and (5,6,8)

Occerence of the root of T : 2 (2,5)

Frequency of T in D : 2/8

FREQT



Algorithm FREQT

Input: A set $\mathcal{L}$ of labels, a data tree D on $\mathcal{L}$, and a *minimum support* $0 < \sigma \leq 1$.

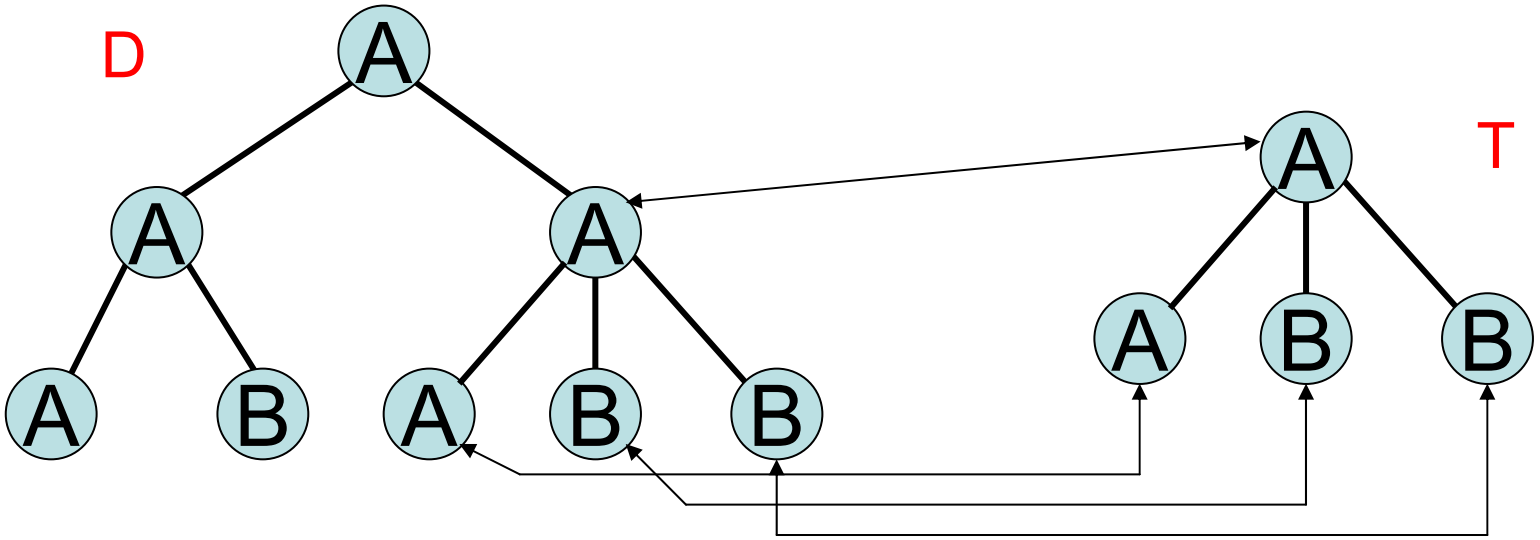
Output: The set $\mathcal{F}$ of all σ-frequent patterns in D.

1. Compute the set $\mathcal{C}_1 := \mathcal{F}_1$ of σ-frequent 1-patterns and the set RMO_1 of their rightmost occurrences by scanning D; Set $k := 2$;
2. While $\mathcal{F}_{k-1} \neq \emptyset$, do:

   2 (a) $\langle \mathcal{C}_k, RMO_k \rangle :=$ Expand-Trees$(\mathcal{F}_{k-1}, RMO_{k-1})$; Set $\mathcal{F}_k := \emptyset$.

   2 (b) For each pattern $T \in \mathcal{C}_k$, do the followings: Compute $freq_D(T)$ from $RMO_k(T)$, and then, if $freq_D(T) \geq \sigma$, then $\mathcal{F}_k := \mathcal{F}_k \cup \{T\}$.
3. Return $\mathcal{F} = \mathcal{F}_1 \cup \cdots \cup \mathcal{F}_{k-1}$.

FREQT: Beispiel

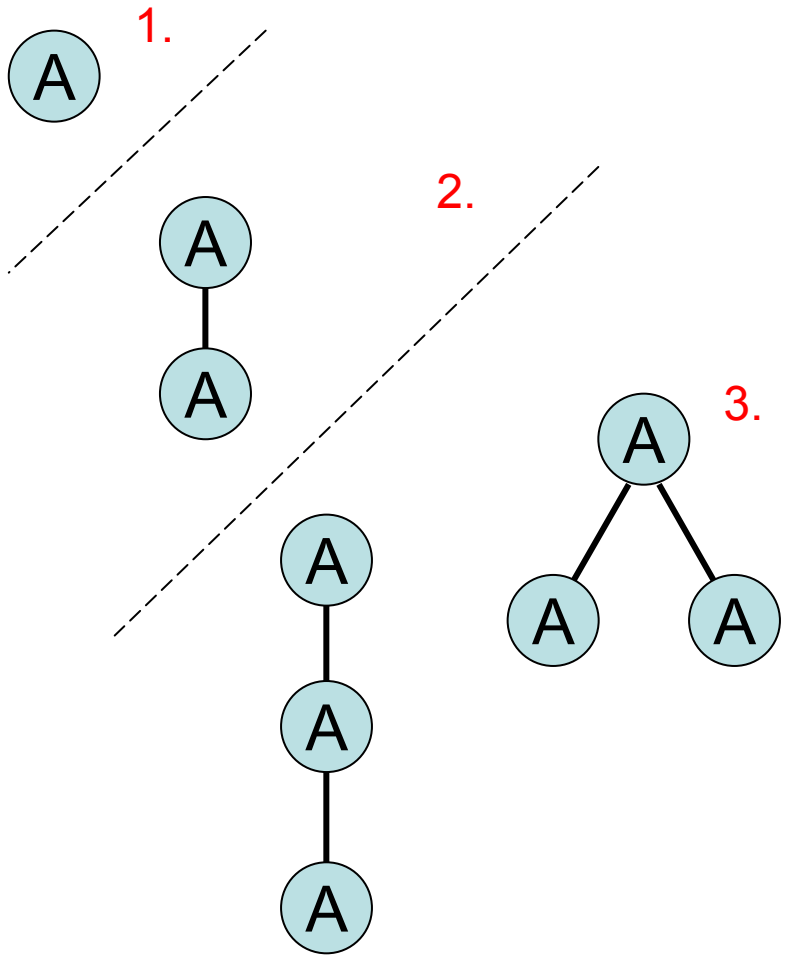


D = ((1,A) (2,A) (3,A) (3,B) (2,A) (3,A) (3,B) (3,B))

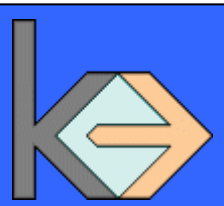
T = ((1,A) (2,A) (2,B) (2,B))

Enumeration

- Fang mit Menge von Bäume, die haben nur ein Knot an
- Mit jede Iteration ist ein Baum expandiert mit zufügen von ein neues Knot
  - Rightmost Expansion



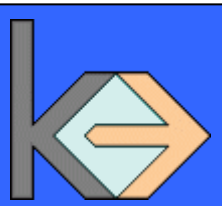
Enumeration



Algorithm Expand-Trees($\mathcal{F}, RMO$)

1. $\mathcal{C} := \emptyset$; $RMO_{\text{new}} := \emptyset$;
2. For each tree $S \in \mathcal{F}$, do:
   - For each $(p, \ell) \in \{1, \ldots, d\} \times \mathcal{L}$, do the followings, where d is the depth of the rightmost leaf of S:
     - Compute the (p, ℓ)-expansion T of S;
     - $RMO_{\text{new}}(T) :=$ Update-RMO$(RMO(S), p, \ell)$;
     - $\mathcal{C} = \mathcal{C} \cup \{T\}$;
3. Return $\langle \mathcal{C}, RMO_{\text{new}} \rangle$;

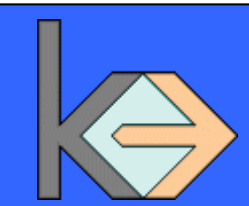
Updating Occurence Lists



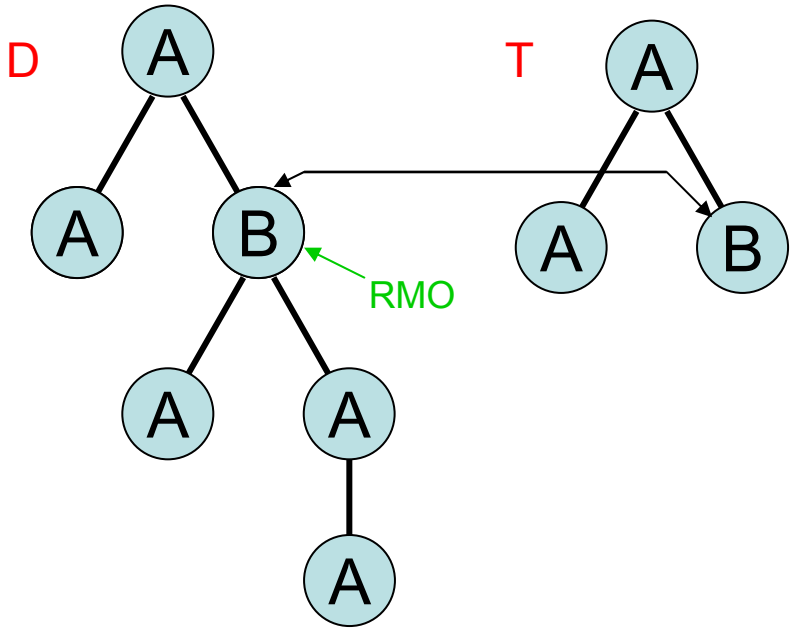
Algorithm Update-RMO(RMO, p, ℓ)

1. Set RMO_{new} to be the empty list ε and $check := null$.
2. For each element $x \in RMO$, do:
   (a) If $p = 0$, let y be the leftmost child of x.
   (b) Otherwise, $p \geq 1$. Then, do:
      – If $check = {\pi_D}^p(x)$ then skip x and go to the beginning of Step 2 (Duplicate-Detection).
      – Else, let y be the next sibling of ${\pi_D}^{p-1}(x)$ (the $(p-1)$st parent of x in D) and set $check := {\pi_D}^p(x)$.
   (c) While $y \neq null$, do the following:
      – If $L_D(y) = \ell$, then $RMO_{\text{new}} := RMO_{\text{new}} \cdot (y)$; /* Append */
      – $y := next(y)$; /* the next sibling */
3. Return RMO_{new}.

Update RMO : Beispiel

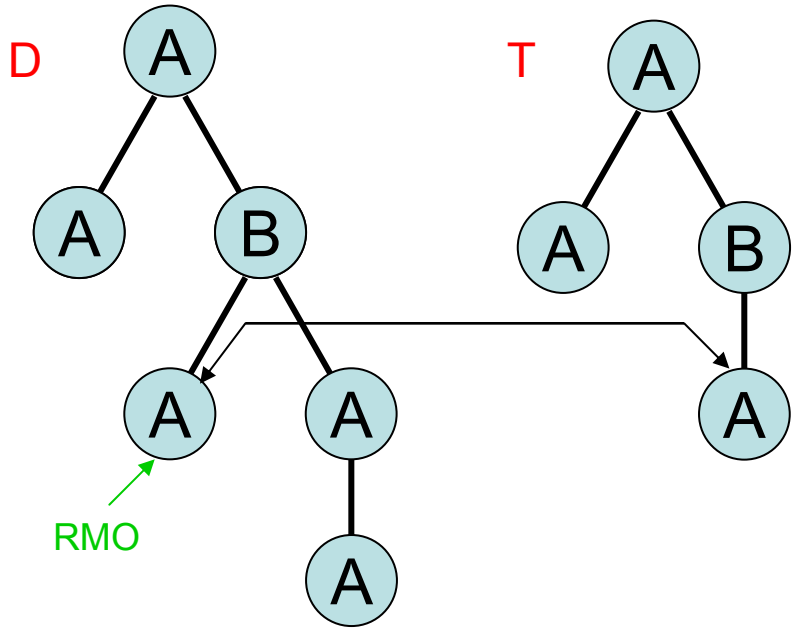


- Nicht alle Matchfunktionen sind gespeichert
  - wir brauchen nur der Rightmost Position
  - alles anders kann man leicht wieder erstellen



Update RMO : Beispiel

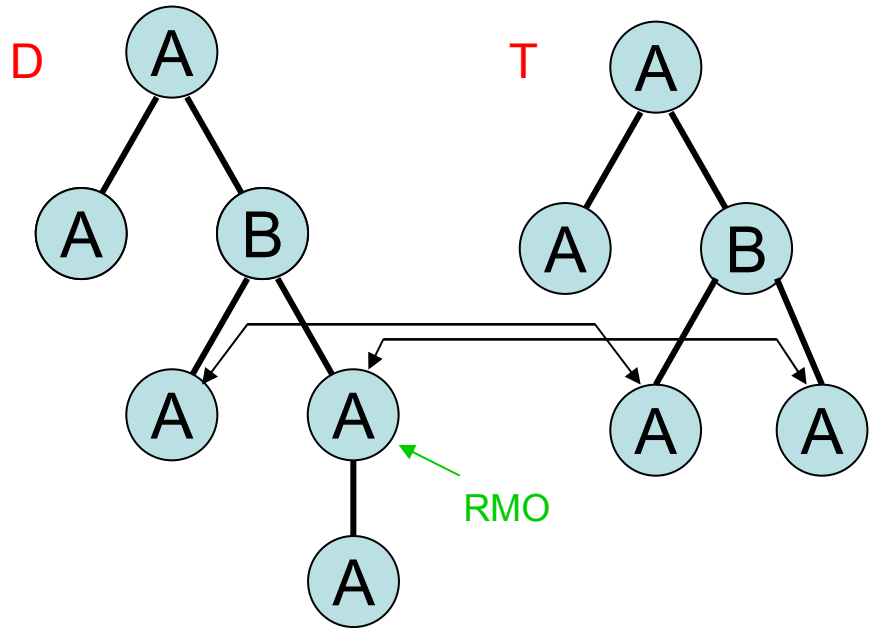
- Für Extension (0,A) ist der neue RMO gezeigt



Update RMO : Beispiel

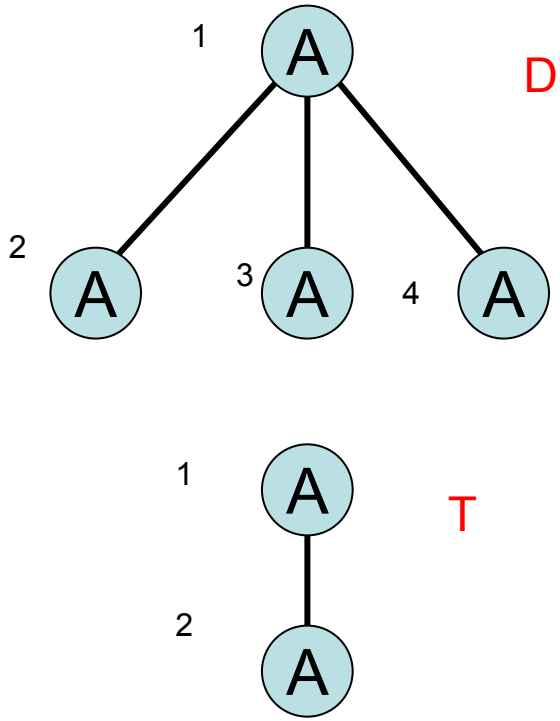
- Für Extension (0,A) ist der neue RMO gezeigt

- Für Extension (1,A) ist der neue RMO gezeigt

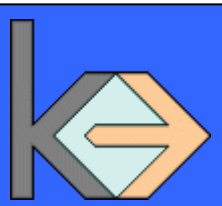


Duplicate-Detection

- Pattern Tree T hat 3 Matches
  - (1,2)
  - (1,3)
  - (1,4)
- Aber σ=1/4, nicht 3/4



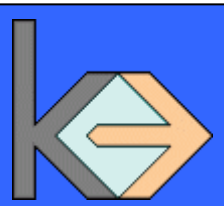
Pruning



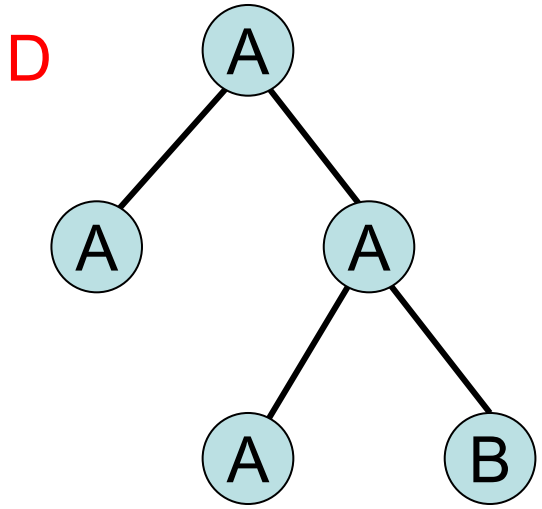
- Node Skip: Für Datenbaumknoten mit unfrequent Labels würden RMO-Update nicht angerufen

- Edge Skip: Kanten zwischen Knotenpaare, die in der 2. Iteration von Expand-Tree als nicht frequent bemerkt sind, kann man auch ignorieren

Pruning : Beispiel

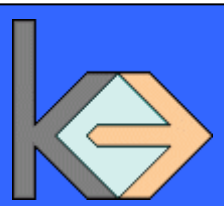


- Node-Skip :
  - σ = 0.3
  - σ (B) = 0.2
  - B ist ignoriert



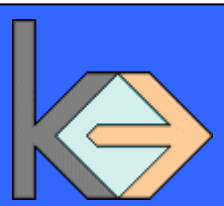
- Edge-Skip :
  - Kante (A,B) ist auch nicht frequent
  - Eine solche Extension ist ignoriert

Experimental Results



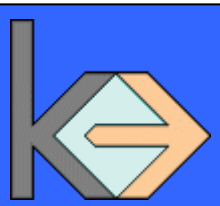
- Freqt, Freqt ohne Duplicate-Detection and Freqt mit Node-Edge-Pruning
- 2 Dataset von HTML Pages
- 2 threshold für Frequency
  - σ=10%, σ=2%

Experimental Results



- Bei σ=10% : 1 Pattern gefunden, gleiche Zeit für alle
- Bei σ=2% : Freqt ist 10 mal schneller als Freqt ohne Duplicate-Detection und 3 mal langsamer als Freqt mit Node-Edge-Skip

- Komplexität von Freqt : $O (k^2bLN)$



Danke für Ihre Aufmerksamkeit!