Humboldt-Universität Informatik

Humboldt Universit t zu Berlin
Institut f r Informatik

XTRACT – ein Überblick

VL XML, XPath, XQuery: Neue Konzepte für Datenbanken

Jörg Pohle, pohle@informatik.hu-berlin.de
Daniel Apelt, apelt@informatik.hu-berlin.de

Überblick

- Überblick über unsere Motivation
- Motivation der Verfasser
- Grundsätzlicher Aufbau
- Vorstellung der einzelnen Module und deren Komplexität
- Mögliche Probleme und Fehler

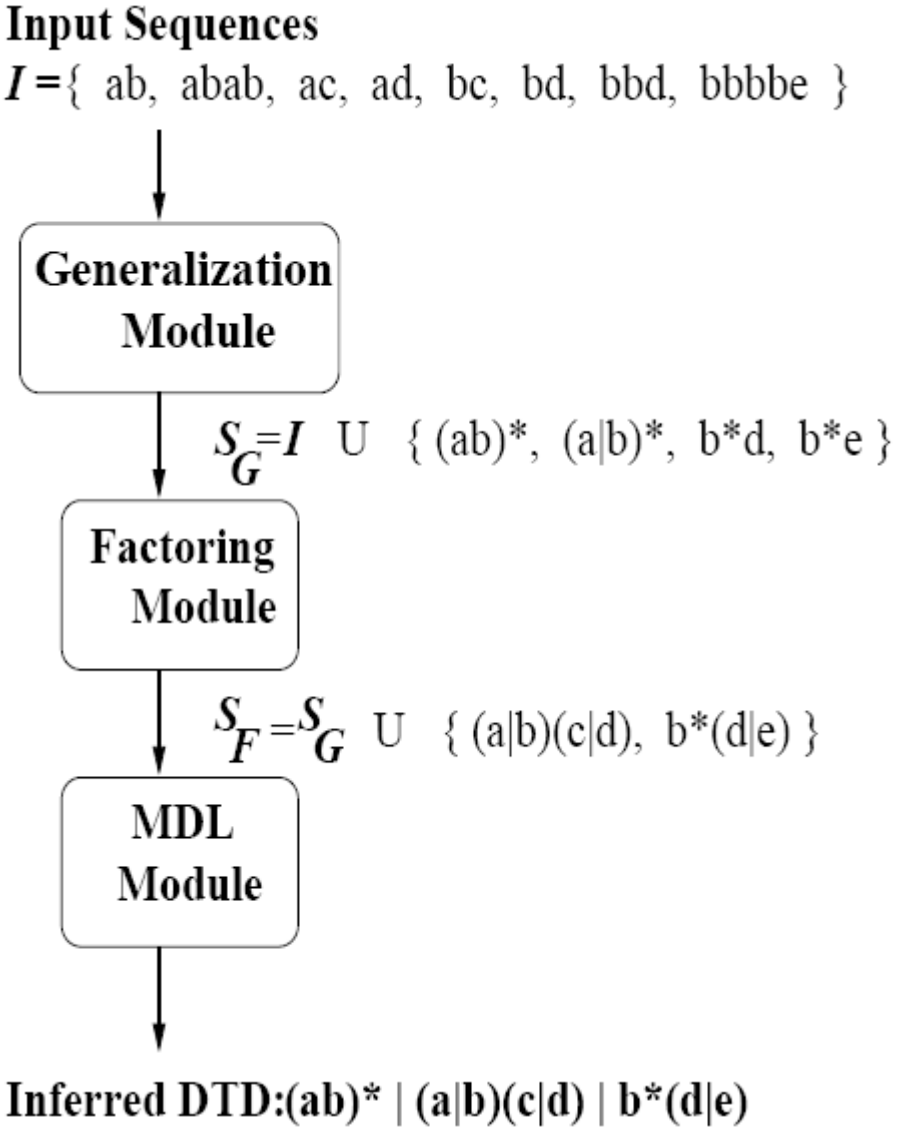
Motivation des Paper

- viele XML-Dokumente für die keine DTD existiert
- unterschiedliche Dokumente aus verschiedenen Quellen zusammenführen, deswegen gemeinsame DTD (Webservices)
- Einschränkungen: nur Struktur der Elemente, keine Attribute

Restriktionen in Bezug auf die DTD:

R1: The DTD should be concise (i.e., small in size).
R2: The DTD should be precise (i.e, not cover too many sequences not contained in *I*).

Architektur



Generalization

Ziel: vorhandene Muster in jeder Eingabe-sequenz mit regulären Ausdrücken ersetzen und diese hinzufügen zu S_G

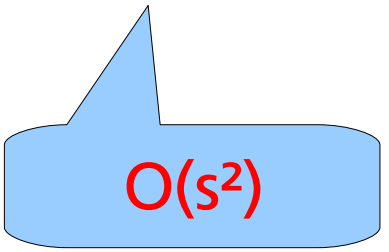
bbbbe → b*e
abab → (ab)*
abab → (a|b)*

```
Generalize(I)
   DiscoverSeqPattern(s, r)
   DiscoverOrPattern(s, d)
                  Partition(s, d)
```

Generalization

Komplexität

```
procedure Generalize(I)
begin
   for each sequence s in I
      add s to S_G
      for r := 2, 3, 4
         s' := DiscoverSeqPattern(s,r)
         for d := 0.1 · |s'|, 0.5 · |s'|, |s'|
            s'' := DiscoverOrPattern(Partition(s',d),d)
            add s'' to S_G
end
```



Generalization

Komplexität

```
procedure Generalize(I)
begin
    for each sequence s in I
        add s to S_G
        for r := 2, 3, 4
            s' := DiscoverSeqPattern(s,r)
            for d := 0.1 · |s'|, 0.5 · |s'|, |s'|
                s'' := DiscoverOrPattern(Partition(s',d),d)
                add s'' to S_G
end
```

$O(s^2 + s^2)$

Generalization

Komplexität

```
procedure Generalize(I)
begin
    for each sequence s in I
        add s to S_G
        for r := 2, 3, 4
            s' := DiscoverSeqPattern(s,r)
            for d := 0.1 · |s'|, 0.5 · |s'|, |s'|
                s'' := DiscoverOrPattern(Partition(s',d),d)
                add s'' to S_G
end
```

$O(3 * (2 * s^2))$

Generalization

Komplexität

```
procedure Generalize(I)
begin
    for each sequence s in I
        add s to S_G
        for r := 2, 3, 4
            s' := DiscoverSeqPattern(s,r)
            for d := 0.1 · |s'|, 0.5 · |s'|, |s'|
                s'' := DiscoverOrPattern(Partition(s',d),d)
                add s'' to S_G
end
```

$O(s^3 + (6 * s^2))$

Generalization

Komplexität

```
procedure Generalize(I)
begin
    for each sequence s in I
        add s to S_G
        for r := 2, 3, 4
            s' := DiscoverSeqPattern(s,r)
            for d := 0.1 · |s'|, 0.5 · |s'|, |s'|
                s'' := DiscoverOrPattern(Partition(s',d),d)
                add s'' to S_G
end
```

$O(3 * (s^3 + 6 * s^2))$

Generalization

Komplexität

```
procedure Generalize(I)
begin
    for each sequence s in I
        add s to S_G
        for r := 2, 3, 4
            s' := DiscoverSeqPattern(s,r)
            for d := 0.1 · |s'|, 0.5 · |s'|, |s'|
                s'' := DiscoverOrPattern(Partition(s',d),d)
                add s'' to S_G
end
```

$O(i * (3 * s^3 + 18 * s^2))$

Generalization

Komplexität

```
procedure Generalize(I)
begin
   for each sequence s in I
      add s to S_G
      for r := 2, 3, 4
         s' := DiscoverSeqPattern(s,r)
         for d := 0.1 · |s'|, 0.5 · |s'|, |s'|
            s'' := DiscoverOrPattern(Partition(s',d),d)
            add s'' to S_G
end
```

$O(3 * i * s^3 + 18 * i * s^2)$

Generalization

Komplexität

$|S_G| \leq 10 * |I|$

$O(3 * i * s^3 + 18 * i * s^2)$

```
procedure Generalize(I)
begin
    for each sequence s in I
        add s to S_G
        for r := 2, 3, 4
            s' := DiscoverSeqPattern(s,r)
            for d := 0.1 · |s'|, 0.5 · |s'|, |s'|
                s'' := DiscoverOrPattern(Partition(s',d),d)
                add s'' to S_G
end
```

Factoring

Ziel: Ausdrücke aus S_G kompakter gestalten und zu S_F hinzufügen

b*d, b*e	→ b * (d | e)
ac, ad, bc, bd	→ (a | b) (c | d)

Factoring

Komplexität

```
procedure FactorSubsets(S_G)
begin
     for each DTD D in S_G
        Compute score(D,S_G)
     ...
     [overlap(D,D')]
     for i:=1 to k
        ...
     for each DTD D in SeedSet
        ...
        while (S' is not empty)
           ...
        F:=Factor(S)
        S_F:=S_F U {F_1,...,F_m}
end
```

$O(|S_G| * s)$

Factoring

Komplexität

```
procedure FactorSubsets(S_G)
begin
     for each DTD D in S_G
        Compute score(D,S_G)
     ...
     [overlap(D,D')]
     for i:=1 to k
        ...
     for each DTD D in SeedSet
        ...
        while (S' is not empty)
           ...
        F:=Factor(S)
        S_F:=S_F U {F_1,...,F_m}
end
```

$O(|S_G| * s)$

$O(|S_G|^3)$

Factoring

Komplexität

```
procedure FactorSubsets(S_G)
begin
     for each DTD D in S_G
        Compute score(D,S_G)
     ...
     [overlap(D,D')]
     for i:=1 to k
        ...
     for each DTD D in SeedSet
        ...
        while (S' is not empty)
           ...
        F:=Factor(S)
        S_F:=S_F U {F_1,...,F_m}
end
```

$O(|S_G| * s)$

$O(|S_G|^3)$

$O(k * |S_G|)$

Factoring

Komplexität

```
procedure FactorSubsets(S_G)
begin
     for each DTD D in S_G
        Compute score(D,S_G)
     ...
     [overlap(D,D')]
     for i:=1 to k
        ...
     for each DTD D in SeedSet
        ...
        while (S' is not empty)
           ...
        F:=Factor(S)
        S_F:=S_F U {F_1,...,F_m}
end
```

$O(|S_G| * s)$

$O(|S_G|^3)$

$O(k * |S_G|)$

$O(|S_G|^2 * s)$

Factoring

Komplexität

```
procedure FactorSubsets(S_G)
begin
     for each DTD D in S_G
        Compute score(D,S_G)
     ...
     [overlap(D,D')]
     for i:=1 to k
        ...
     for each DTD D in SeedSet
        ...
        while (S' is not empty)
           ...
        F:=Factor(S)
        S_F:=S_F U {F_1,...,F_m}
end
```

$O(|S_G| * s)$

$O(|S_G|^3)$

$O(k * |S_G|)$

$O(|S_G|^2 * s)$

$O(|S_G|^2)$???

Factoring

Komplexität

```
procedure FactorSubsets(S_G)
begin
     for each DTD D in S_G
        Compute score(D,S_G)
     ...
     [overlap(D,D')]
     for i:=1 to k
        ...
     for each DTD D in SeedSet
        ...
        while (S' is not empty)
           ...
        F:=Factor(S)
        S_F:=S_F U {F_1,...,F_m}
end
```

$O(|S_G| * s)$

$O(|S_G|^3)$

$O(k * |S_G|)$

$O(|S_G| * (|S_G|^2 + |S_G|^2 * s))$

Factoring

Komplexität

$O(|S_G| * s + |S_G|^3 + k * |S_G| + |S_G|^3 * (1 + s))$

```
procedure FactorSubsets(S_G)
begin
     for each DTD D in S_G
        Compute score(D,S_G)
     ...
     [overlap(D,D')]
     for i:=1 to k
        ...
     for each DTD D in SeedSet
        ...
        while (S' is not empty)
           ...
        F:=Factor(S)
        S_F:=S_F U {F_1,...,F_m}
end
```

Factoring

Komplexität

$O((10i)^3 * (1 + s) + i * (s + k))$

```
procedure FactorSubsets(S_G)
begin
     for each DTD D in S_G
        Compute score(D,S_G)
     ...
     [overlap(D,D')]
     for i:=1 to k
        ...
     for each DTD D in SeedSet
        ...
        while (S' is not empty)
           ...
        F:=Factor(S)
        S_F:=S_F U {F_1,...,F_m}
end
```

Factoring

Komplexität

$O((10i)^3 * (1 + s) + i * (s + k))$

$|S_F| \leq 2 * |S_G| \leq 20 * |I|$

```
procedure FactorSubsets(S_G)
begin
     for each DTD D in S_G
        Compute score(D,S_G)
     ...
     [overlap(D,D')]
     for i:=1 to k
        ...
     for each DTD D in SeedSet
        ...
        while (S' is not empty)
           ...
        F:=Factor(S)
        S_F:=S_F U {F_1,...,F_m}
end
```

Minimum Description Length

Von mehreren Theorien, die den gleichen Sachverhalt beschreiben, ist die zu bevorzugen, die die einfachste (kürzeste in der Informationstheorie) ist.

Minimierung der Summe von:

a) benötigter Speicher (Bit), um die DTD zu beschreiben

$$for\ s \in S_F \rightarrow |s| \cdot \log(|\Sigma| + |M|)$$

b) Länge (Bit) der Daten, wenn sie mit der DTD codiert werden

$$for\ s \in S_F \rightarrow Kosten\ für\ Darstellung\ von\ i \in I\ mit\ s$$

MDL

Facility Location Problem

Menge von Standorten s_i mit Eröffnungskosten $c_i \geq 0$

Menge von Märkten m_j mit einem maximalen Preis P_j

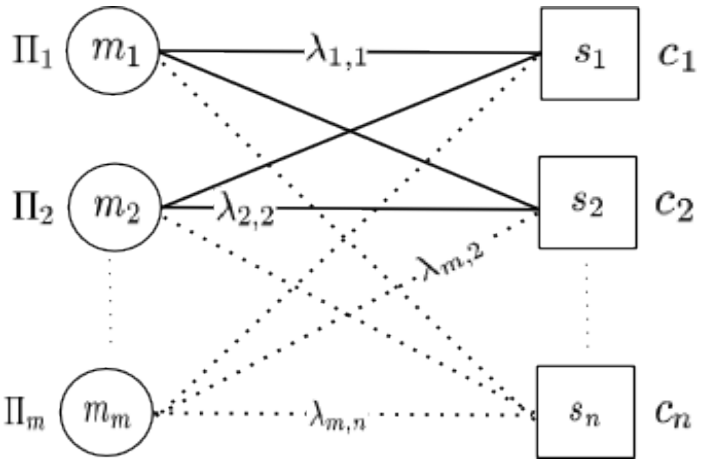
vollständiger, bipartiter Graph mit Kantengewichten d_{ij}

Lösung:

zu eröffnende Standorte:

$L \subset S := \{s_1, .., s_t\}$

$$H(L) = \sum_{i=1}^{m} \max_{x \in L} \left(P_i - d_{ij} \right) - \sum_{e \in L} c_e$$



Klaas Joeppen, Facility Location Game

MDL

Komplexität

- Kodierung „Part (A)“

$O(|S_F| * s)$

- Kodierung „Part (B)“

- DTDs mit minimal MDL-Kosten wählen

MDL

Komplexität

- Kodierung „Part (A)“ — $O(|S_F| * s)$
- Kodierung „Part (B)“ — $O(s^2 * i * |S_F|)$
- DTDs mit minimal MDL-Kosten wählen

MDL

Komplexität

- Kodierung „Part (A)“ — $O(|S_F| * s)$
- Kodierung „Part (B)“ — $O(s^2 * i * |S_F|)$
- DTDs mit minimal MDL-Kosten wählen — $O(|S_F|^2 * \log |S_F|)$

MDL

Komplexität

$O(|S_F| * s + s^2 * i * |S_F| + |S_F|^2 * \log |S_F|)$

- Kodierung „Part (A)“

$O(|S_F| * s)$

- Kodierung „Part (B)“

$O(s^2 * i * |S_F|)$

- DTDs mit minimal MDL-Kosten wählen

$O(|S_F|^2 * \log |S_F|)$

MDL

Komplexität

$O(20i * s + s^2 * 20i^2 + 400i^2 * \log 20i)$

- Kodierung „Part (A)“

$O(|S_F|)$

- Kodierung „Part (B)“

$O(s^2 * i * |S_F|)$

- DTDs mit minimal MDL-Kosten wählen

$O(|S_F|^2 * \log |S_F|)$

Komplexität

Die Betrachtung der Komplexität wurde von den Autoren in den etwas komplizierteren Abschnitten weggelassen.

Gesamtbetrachtung:

- Generalization Subsystem: $O(3 * i * s^3 + 18 * i * s^2)$
- Factoring Subsystem: $O(1000 * i^3 * (1 + s) + i * (s + k))$
- MDL Subsystem: $O(20i * s + 20i^2 * s^2 + 400i^2 * \log 20i)$

- Gesamt: $O(3 * i * s^3 + 10^3 * i^3 * s + 20i^2 * s^2)$
  $= O(x * i * s * (i^2 + s^2 + i * s))$
  $= O(n^4)$

Probleme

Generalization: nur eine Auswahl von komplexen Mustern wird unterstützt (vgl. S. 15, unten)

MDL: Problem der Nicht-1-Mehrdeutigkeit wurde vergessen

Probleme

Partitionen: Zahl der Partitionen im Beispiel falsch (vgl. S. 18, mitte)

Factoring: score (D,S) – Mengenklammer falsch, "*" oder "·" (vgl. S. 20)

Probleme

Codierung (MDL): Kosten für Suchen des richtigen regulären Ausdrucks aus der veroderten Liste (DTD) werden nicht betrachtet, Auswahl der passenden DTD D_i aus ($D_1 \mid D_2 \mid \ldots \mid D_n$) für s

Komplexität: Betrachtungen unvollständig und teilweise mangelhaft, keine Gesamtbetrachtung

Literatur

Minos Garofalakis, Aristides Gionis, Rajeev Rastogi, S. Seshadri and Kyuseok Shim: XTRACT: Learning Document Type Descriptors from XML Document Collections

Christian Romberg: Untersuchung zur automatischen XML Schema-Ableitung, 2001 , Diplomarbeit
http://www.xml-und-datenbanken.de/sada/da-romberg.ps.gz

Matthias Brückner, Ableitung eines Schemata aus XML-Dokumenten, Hauptseminar 2002/03 Uni Rostock, Folien & Seminararbeit
http://wwwdb.informatik.uni-rostock.de/Lehre/Vorlesungen/hs_ws2002_2003/hs-brueckner.pdf
http://wwwdb.informatik.uni-rostock.de/Lehre/Vorlesungen/hs_ws2002_2003/folien_brueckner.ppt

Klaas Joeppen, Seminar über Algorithmen, Facility Location Game
http://www.inf.fu-berlin.de/lehre/SS06/SeminarAlgorithmen/5-FacilityLocation.ppt

Ralf Behrens, On the Complexity of Standard and Specialized DTD Parsing
http://www.is.informatik.uni-kiel.de/~hjk/wsgdboo/Paper/Behrens.ps

Vielen Dank für Eure Aufmerksamkeit