The **lxto** System

Applications in Business Intelligence & the Semantic Web

Georg Gottlob

Oxford University

ESWC Innsbruck 2007
Talk Outline

• Motivation: Need of Web data extraction
• The LiXto Approach (+ short demo)
• Theoretical Results
• Lixto Suite + Applications
• Creating RDF Bases
• Research Issues: Web feature extraction etc.
Electronic Commerce

E-procurement, quality assurance:

- IVECO
- Quality norm 06865 for tires
- OEM PORTAL
- Pirelli
- Michelin
- Avon
- Dunlop
- Firestone
The Wall

Problem: Make web contents accessible to electronic data processing

WEB
HTML pages
layout

Corporate edp apps
structured data,
Databases,
XML
Solutions:

Semantic Web? Yes, but…

Intelligent agents? Yes, but…

Web wrapping
Web information extraction

Goal: Make web contents accessible to electronic data processing
Web information extraction

**Goal:** Make web contents accessible to electronic data processing

Wrappers: **HTML** $\rightarrow$ select $\rightarrow$ extract $\rightarrow$ annotate $\rightarrow$ **XML**
<table>
<thead>
<tr>
<th>Brand</th>
<th>Model</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goodyear</td>
<td>Eagle NCT 5</td>
<td>£44.10</td>
</tr>
<tr>
<td>DUNLOP</td>
<td>SP Sport 01</td>
<td>£44.30</td>
</tr>
<tr>
<td>Bridgestone</td>
<td>Turanza FD 21</td>
<td>£46.50</td>
</tr>
<tr>
<td>Michelin</td>
<td>Primacy</td>
<td>£56.40</td>
</tr>
<tr>
<td>Goodyear</td>
<td>Eagle NCT 5</td>
<td>£44.20</td>
</tr>
<tr>
<td>Pirelli</td>
<td>P6000</td>
<td>£44.70</td>
</tr>
<tr>
<td>DUNLOP</td>
<td>SP Sport 01</td>
<td>£46.40</td>
</tr>
</tbody>
</table>

*Prices include postage, packing and VAT within mainland UK.

***Note: These tyres are subject to a delivery period of up to 14 days.*
<table>
<thead>
<tr>
<th>Brand</th>
<th>Profile Size</th>
<th>Speed</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goodyear***</td>
<td>FALG UCT G</td>
<td>105/65 R15 01V</td>
<td>mytyres.co.uk only £44.10</td>
</tr>
<tr>
<td>Dunlop***</td>
<td>SP SPORT 01</td>
<td>195/65 R15 91H</td>
<td>mytyres.co.uk only £44.30</td>
</tr>
<tr>
<td>Pirelli***</td>
<td>P6000 Powergy</td>
<td>195/65 R15 91H</td>
<td>mytyres.co.uk only £44.70</td>
</tr>
<tr>
<td>Continental***</td>
<td>EcoContact GP</td>
<td>195/65 R15 91H runout,</td>
<td>mytyres.co.uk only £46.00</td>
</tr>
<tr>
<td>Bridgestone***</td>
<td>Turanza ER 31</td>
<td>195/65 R15 91H</td>
<td>mytyres.co.uk only £46.50</td>
</tr>
<tr>
<td>Michelin***</td>
<td>Pilot PRIMACY</td>
<td>195/65 R15 91H</td>
<td>mytyres.co.uk only £56.40</td>
</tr>
<tr>
<td>Goodyear***</td>
<td>FALG UCT G</td>
<td>195/65 R15 01V</td>
<td>mytyres.co.uk only £44.20</td>
</tr>
<tr>
<td>Pirelli***</td>
<td>P6000 Powergy</td>
<td>195/65 R15 01V</td>
<td>mytyres.co.uk only £44.70</td>
</tr>
<tr>
<td>Dunlop***</td>
<td>SP SPORT 01</td>
<td>195/65 R15 91H</td>
<td>mytyres.co.uk only £46.40</td>
</tr>
<tr>
<td>Pirelli***</td>
<td>P6000 Powergy</td>
<td>195/65 R15 01V</td>
<td>mytyres.co.uk only £46.70</td>
</tr>
</tbody>
</table>

10 tyres from 118 were displayed. (1 - 10)

Prices includes postage, packing and VAT within mainland UK.

***Please note: these tyres are subject to a delivery period of up to 1 week.
<?xml version="1.0" encoding="UTF-8"?>
<document>
  <record>
    <number>409449118</number>
    <item>98 Degrees - Notebook - New</item>
    <picture/>
    <price>2.99</price>
    <currency>$</currency>
    <bids>-</bids>
  </record>
  <record>
    <number>413171469</number>
    <item>Notebook - Compaq Presario 1207</item>
    <price>730.00</price>
    <currency>AU $</currency>
    <bids>-</bids>
  </record>
  [...]
</document>
Function \( f: \) HTML Parse tree \( \Rightarrow \) Subtrees

Leaves of subtrees are among leaves of orig. tree
HTML page = tree

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html> <body>
<h1>People @ DBAI</h1>
<table border="1" cellpadding="3" cellspacing="1">
  <tr> <td>Georg Gottlob</td> <td>gottlob@dbai.tuwien.ac.at</td> <td>18420</td> </tr>
  <tr> <td>Christoph Koch</td> <td>koch@dbai.tuwien.ac.at</td> <td>18449</td> </tr>
</table>
</body> </html>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html> <body>
<h1>People @ DBAI</h1>
<table border="1" cellspacing="1" cellpadding="3">
<tr> 
<td>Georg Gottlob</td> 
<td>gottlob@dbai.tuwien.ac.at</td> 
<td>18420</td> 
</tr> 
<tr> 
<td>Christoph Koch</td> 
<td>koch@dbai.tuwien.ac.at</td> 
<td>18449</td> 
</tr> 
</table> 
</body> </html>
### Predicate \textit{employee}

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>  
  <body>
    <h1>People @ DBAI</h1>
    <table border="1" cellpadding="3" cellspacing="1">
      <tr>
        <td>Georg Gottlob</td>
        <td>gottlob@dbai.tuwien.ac.at</td>
        <td>18420</td>
      </tr>
      <tr>
        <td>Christoph Koch</td>
        <td>koch@dbai.tuwien.ac.at</td>
        <td>18449</td>
      </tr>
    </table>
  </body>
</html>
```
<html> <body>
<h1>People @ DBAI</h1>
<table border="1" cellpadding="3" cellspacing="1">
<tr> 
<td>Georg Gottlob</td>
<td>gottlob@dbai.tuwien.ac.at</td>
<td>18420</td>
</tr>
<tr> 
<td>Christoph Koch</td>
<td>koch@dbai.tuwien.ac.at</td>
<td>18449</td>
</tr>
</table>
</body> </html>
Desiderata of Tree Wrapping Languages

• Clear semantics.
• Good trade-off between complexity and expressiveness.
• Easy to use as wrapper programming language.
• Suitable for being incorporated into visual tools.
Expressiveness Yardstick: MSO

• MSO captures exactly the essence of data extraction:
  ✓ Define sets of nodes of a document
  ✓ Re-label tree nodes by monadic predicates
  ✓ Declare some as irrelevant

Expressiveness, complexity, semantics well understood:
  ✓ MSO over trees: perfect logical semantics
  ✓ MSO over trees: high expressive power (tree automata)
  ✓ MSO queries over trees: low data complexity

• Drawbacks: - hard to use, no visual specification,
  - high query complexity (bad scalability).
Ordered Trees as finite structures

html
  ↓
body
  ↓
h1
table
  ↓
tr
td
td
td
td
td
  ↓
Georg Gottlob
gottlob@dbai.tuwien.ac.at
18420
Christoph Koch
koch@dbai.tuwien.ac.at
18449

firstchild

label_{h1}()
label_{td}()
...
root()
leaf()

td
  ↓
gottlob@dbai.tuwien.ac.at
18420
Christoph Koch
koch@dbai.tuwien.ac.at
18449

nextsibling

MSO over Trees

Extract from a binary tree all roots of sub-trees with an odd number of leaves:

\[ \exists S \forall x [ \ S(u) \ & \ (leaf(x) \rightarrow S(x)) \ & \ \forall x,y,z ((firstchild(x,y) \ & \ nextsibling(y,z)) \rightarrow (S(x) \leftrightarrow \neg(S(y) \leftrightarrow S(z)))) ] \]
Logic

Database theory

DB programming

Application design
Logic

Database theory

DB programming

Application design

MSO

Monadic Datalog
Logic

Database theory

DB programming

Application design

MSO

II

Monadic Datalog
Logic

Database theory

DB programming

Application design

MSO

‖

Monadic Datalog

Elog
Logic

Database theory

DB programming

Application design

MSO

\|

Monadic Datalog

\n
Elog
Logic

Database theory

DB programming

Application design

MSO

∥

Monadic Datalog

∩

Elog

Lixto Visual Wrapper
Logic

Database theory

DB programming

Application design

MSO

Monadic Datalog

Elog

Lixto Visual Wrapper

LiXto Suite
Monadic Datalog as a Wrapping Language

entry(X) :- root(R), firstchild(R,U), label[html](U),
           firstchild(U,V), label[body](V),
           firstchild(V,W), label[table](W),
           firstchild(W,X), label[tr](X).
entry(X):- entry(Y), nextsibling(Y,X).
name(X) :- entry(E), firstchild(E, X), label[td](X).
email(X) :- name(N), nextsibling(N, X), label[td](X).
phone(X) :- email(M), nextsibling(M, X), label[td](X).
Monadic Datalog as a Wrapping Language

entry(X) :- root(R), firstchild(R,U), label[html](U),
           firstchild(U,V), label[body](V),
           firstchild(V,W), label[table](W),
           firstchild(W,X), label[tr](X).

entry(X) :- entry(Y), nextsibling(Y,X).

name(X) :- entry(E), firstchild(E,X), label[td](X).

email(X) :- name(N), nextsibling(N,X), label[td](X).

phone(X) :- email(M), nextsibling(M,X), label[td](X).
Monadic Datalog as a Wrapping Language

entry(X) :- root(R), firstchild(R,U), label[html](U),
           firstchild(U,V), label[body](V),
           firstchild(V,W), label[table](W),
           firstchild(W,X), label[tr](X).
entry(X):- entry(Y), nextsibling(Y,X).

name(X) :- entry(E), firstchild(E, X), label[td](X).
email(X) :- name(N), nextsibling(N, X), label[td](X).
phone(X) :- email(M), nextsibling(M, X), label[td](X).
Monadic Datalog as a Wrapping Language

entry(X) :- root(R), firstchild(R,U), label[html](U),
         firstchild(U,V), label[body](V),
         firstchild(V,W), label[table](W),
         firstchild(W,X), label[tr](X).

entry(X) :- entry(Y), nextsibling(Y,X).

name(X) :- entry(E), firstchild(E, X), label[td](X).

email(X) :- name(N), nextsibling(N, X), label[td](X).

phone(X) :- email(M), nextsibling(M, X), label[td](X).
Monadic Datalog as a Wrapping Language

entry(X) :- root(R), firstchild(R,U), label[html](U),
            firstchild(U,V), label[body](V),
            firstchild(V,W), label[table](W),
            firstchild(W,X), label[tr](X).

entry(X) :- entry(Y), nextsibling(Y,X).

name(X) :- entry(E), firstchild(E, X), label[td](X).

email(X) :- name(N), nextsibling(N, X), label[td](X).

phone(X) :- email(M), nextsibling(M, X), label[td](X).
Monadic Datalog as a Wrapping Language

entry(X) :- root(R), firstchild(R,U), label[html](U),
  firstchild(U,V), label[body](V),
  firstchild(V,W), label[table](W),
  firstchild(W,X), label[tr](X).

entry(X):- entry(Y), nextsibling(Y,X).

name(X) :- entry(E), firstchild(E, X), label[td](X).

email(X) :- name(N), nextsibling(N, X), label[td](X).

phone(X) :- email(M), nextsibling(M, X), label[td](X).
entry(X) :- root(R), firstchild(R,U), label[html](U),
    firstchild(U,V), label[body](V),
    firstchild(V,W), label[table](W),
    firstchild(W,X), label[tr](X).
entry(X):- entry(Y), nextsibling(Y,X).
name(X) :- entry(E), firstchild(E, X), label[td](X).
email(X) :- name(N), nextsibling(N, X), label[td](X).
phone(X) :- email(M), nextsibling(M, X), label[td](X).

<?xml version="1.0"?>
<peopledb>
  <entry> <name>Georg Gottlob</name>
    <email>gottlob@dbai.tuwien.ac.at</email>
    <phone>18420</phone>
  </entry>
  <entry> <name>Christoph Koch</name>
    <email>koch@dbai.tuwien.ac.at</email>
    <phone>18449</phone>
  </entry>
</peopledb>
The ELOG Programming Language

Monadic Datalog enriched with several practical primitives for handling objects on Web pages:

- Expressing a tree-path at once
- Extracting data items from plain text
- Following URLs and extracting from several pages
- Keeping track of the pattern hierarchy
- Access to ontological databases
ELOG Program for eBay pages

tableseq(S, X) ← document("www.ebay.com/", S), subseq(S, (.body, []), (.table, []), (.table, []), X),
  before(S, X, (.table, [(elementtext, item, [])]), 1, 1, _, _), after(S, X, (.hr, []), 1, 1, _, _)

record(S, X) ← tableseq(_, S), subelem(S, .table, X)

itemnum(S, X) ← record(_, S), subelem(S, *td, X), notbefore(S, X, (.td, []), maxint)

itemdes(S, X) ← record(_, S), subelem(S, (*td, *content, [(a, 0)], X)

price(S, X) ← record(_, S), subelem(S, (*td, [(elementtext, Y, 1)]), X), valuta(Y)

bids(S, X) ← record(_, S), subelem(S, *td, X), before(S, X, (.td, []), 1, 30, Y, _), price(S, Y)

currency(S, X) ← price(_, S), subtext(S, Y, X), valuta(Y)

pricewc(S, X) ← price(_, S), subtext(S, [0 – 9]+, X)
The LiXto Visual Wrapper

- Web
  - Similarly structured pages
  - Example page(s)

- Extraction Module

- Visual Wrapper Generator

- Extraction-program

- XML

Further processing:
- Tracking changes
- Delivering (email, sms)
  ... (Infopipesystem)
SHORT DEMO

Paris Restaurants
Some theoretical results...
Monadic Datalog: Expressive Power and Complexity

Theorems [G. & Koch LICS 2002, J ACM 2004]:

Monadic Datalog = MSO (over trees)

A data extraction task is definable in MSO if and only if it is feasible via a monadic datalog program.

Monadic Datalog has complexity: \( O(|\text{data}| \times |\text{program}|) \)

Good scaling behavior!
Expressive power of LiXto

Elog⁻: Monadic kernel of Elog

Theorems [G., Koch PODS’02, SOFSEM’06]

ELOG⁻ expresses monadic datalog

All of ELOG⁻ is graphically programmable via LiXto

Corollary:

LiXto expresses all MSO wrapping tasks.
The LiXto Suite...
Extraction vs. Querying and Transforming

**DATA EXTRACTION**
- selection
- annotation
- hierarchical (re)grouping

**DATA TRANSFORMATION**
- combination (joins, etc.)
- querying (XSLT, XQUERY)
- personalization

HTML → XML
XML → XSLT
XSLT → RDF
Extraction vs. Querying and Transforming

DATA EXTRACTION
- selection
- annotation
- hierarchical (re)grouping

DATA TRANSFORMATION
- combination (joins, etc.)
- querying (XSLT, XQUERY)
- personalization

HTML
XML
XSLT
RDF

liXto Visual Wrapper
liXto Transformation Server
Product Architecture

Transformation Server

LiXto Extraction Engine

wrappers
WEB ETL

FOR DATA WAREHOUSING
Internal Sources

DBMS

SAP

Extract

Data Warehouse

cleaned data

Transformation

Loading

ETL

Data Marts

OLAP

Data Mining

Data Usage

Data Storage

Data Integration

Data Sources

Reporting
Architecture Overview

Involved Components:

- Lixto TS
- BI Tool
- Oracle 9
B2B Process Integration

Goal:
Automate portal-based quality management tasks

Numbers:
• 9 major automotive OEM portals
• ~50 supply plants

Project duration: ongoing
Web-based B2B processes
Process

OEM | Fiat | Ford | GM | BMW

Automatic Extraction

Email

Intranet

Enterprise Portal

R/3 QM
Automate portal applications

1. Login
Automate portal applications

2. Navigation
Automate portal applications

1. Fill Forms
2. Click
3. Fill Forms
4. Click
Automate portal applications

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<PRRReport>
  <PRRNumber>20030521-1234567</PRRNumber>
  <SupplierName>KRONO SYSTEMS GMBH</SupplierName>
  <Status>NR</Status>
  <LastActivity>21-MAY-2003</LastActivity>
  
  <!-- PRR Aktivitäten -->
  
  <!-- Screen shot of portal activity page -->
  
  <!-- SAP logo -->
</PRRReport>
```
Creating RDF repositories...
Generating a publication ontology

HTML ➔ XML ➔ RDF

Extraction & Processing

Reasoning & Personalization

Lixto Suite (Vienna)

Triple Sesame

Personal Reader Framework (Hannover – Nicola Henze)
PPR Example: Extraction & Processing

HTML → XML → RDF

HTML Extraction → Unifying and Cleaning XML → Generating RDF

Lixto Visual Wrapper

Leveraging XML to RDF

Lixto Transformation Server
Lixto Transformation Server  PPR Screenshot
Rewerse Publication Ontology

Publications Ontology:
weekly crawled & created (Wednesday noon)
6 partners from REWERSE currently
~15,000 lines, ~2000 publications,
dynamic, changing, growing!
Research challenges in Data Extraction

- Interactive pattern learning:
  positive and negative examples, heuristics, theory.
- Auto-adaptation of Wrappers when layout changes.
- Extracting from PDF and similar sources.
- Optimization of Meta-Search.
- Better interaction with terminological reasoning.
- Generic wrappers: Feature extraction.
Generic Data Extraction...
Context of the problem

• automatic product retrieving
• product feature detection & extraction
• from web tabular data.
Context of the problem

- automatic product retrieving
- product feature detection & extraction
- from web tabular data.

TU Wien + Univ. Klagenfurt + Configworks + Lixto
Using Ontologies for Extracting Product Features from Web Pages*

Wolfgang Holzinger, Bernhard Krüpl, and Marcus Herzog

Database and Artificial Intelligence Group, Vienna University of Technology,
Favoritenstraße 9-11, A-1040 Wien, Austria
{holzing, kruepl, herzog}@dbai.tuwien.ac.at

Abstract. In this paper, we show how to use ontologies to bootstrap a knowledge acquisition process that extracts product information from tabular data on Web pages. Furthermore, we use logical rules to reason about product specific properties and to derive higher-order knowledge about product features. We will also explain the knowledge acquisition process, covering both ontological and procedural aspects. Finally, we will give an qualitative and quantitative evaluation of our results.
Typical Candidate
Typical Candidate

<table>
<thead>
<tr>
<th>General</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>5.8 oz</td>
</tr>
<tr>
<td>Width</td>
<td>3.3 in</td>
</tr>
<tr>
<td>Depth</td>
<td>0.9 in</td>
</tr>
<tr>
<td>Height</td>
<td>2.2 in</td>
</tr>
<tr>
<td>Body material</td>
<td>Stainless steel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Main Features</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor resolution</td>
<td>3.2 megapixels</td>
</tr>
<tr>
<td>Optical sensor type</td>
<td>CCD</td>
</tr>
<tr>
<td>Effective sensor resolution</td>
<td>3,200,000 pixels</td>
</tr>
<tr>
<td>Gross sensor resolution</td>
<td>3,300,000 pixels</td>
</tr>
<tr>
<td>Optical sensor size</td>
<td>1/2.7 in</td>
</tr>
<tr>
<td>Light sensitivity</td>
<td>ISO 50, ISO 100, ISO 400</td>
</tr>
</tbody>
</table>
Essential Problem

map table contents to product model

• textual content in table cells
• table layout
• background knowledge
Containment forms
Context

<table>
<thead>
<tr>
<th>Battery</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>200 g</td>
</tr>
<tr>
<td>Size</td>
<td>11x15x8 mm</td>
</tr>
</tbody>
</table>
## Containment forms

### Context

<table>
<thead>
<tr>
<th>Battery</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Weight</strong></td>
<td>200 g</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>11x15x8 mm</td>
</tr>
</tbody>
</table>
Containment forms
Context

<table>
<thead>
<tr>
<th>Battery</th>
</tr>
</thead>
</table>
| Weight  | 200 g  
| Size    | 11x15x8 mm  

○ ○
○ ○
○ ○
Containment forms
Context

| Battery |  
|---|---|
| Weight | 200 g |
| Size | 11x15x8 mm |
Containment forms
Context

<table>
<thead>
<tr>
<th>Battery</th>
<th>200 g</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>11x15x8 mm</td>
</tr>
</tbody>
</table>
Containment forms
Context

• cells are basic building blocks
• cells contain isolated words / phrases
• tabular arrangements group cells - multiple levels possible
• Proposal: these groupings determines which parts form a semantical unit
Dual Interpretation

a HTML <table> structure with 52 <tr> elements, each containing 1 or 2 <td>

a digital camera named “Powershot 116a”, manufactured by “Canon”, 5 Megapixel sensor, etc...
Dual Interpretation

- two distinct ontologies:
  - table
  - product
Dual Interpretation

- two distinct ontologies:
  - table
  - product
A table consists of rows, columns, and cells.
Table Ontology

A table consists of rows, columns, and cells.

<table>
<thead>
<tr>
<th>coll 1</th>
<th>coll 2</th>
<th>r1</th>
</tr>
</thead>
<tbody>
<tr>
<td>width</td>
<td>2 in</td>
<td>r1</td>
</tr>
<tr>
<td>height</td>
<td>1 in</td>
<td>r2</td>
</tr>
<tr>
<td>depth</td>
<td>1.5 in</td>
<td>r3</td>
</tr>
</tbody>
</table>
Table Ontology

A table consists of rows, columns, and cells.

<table>
<thead>
<tr>
<th>col1</th>
<th>col2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>width</td>
<td>2 in</td>
<td>r1</td>
</tr>
<tr>
<td>height</td>
<td>1 in</td>
<td>r2</td>
</tr>
<tr>
<td>depth</td>
<td>1.5 in</td>
<td>r3</td>
</tr>
</tbody>
</table>

- Table contains Row
- Row contains Column
- Column contains Cell
- Cell has stringValue
- stringValue is xsd:string
Product Ontology: Basic Structure

- simple model
- product is a collection of attributes
- attributes have
  - a keyword
  - a value
Value-Types & Spotters

**ProductType**
- 1024 x 768
- 1 cm x 1 cm x 1 cm

**FractionType**
- 1/500
- 1:1.8

**UnitType**
- 7.5 lbs.
- 4.4”

**Spotters** are small procedures that can recognize a value type in a given text.

Value-Types are recognized via *regular expressions* and have distinct OWL concepts each.

Value-Types can be nested:
- 1 cm x 1 cm x 1 cm
- 1 kg = 2 lbs
Digicam Ontology

- specifies each keyword and value in detail
- different spellings, abbreviations
- external types

actual model has > 20 attributes
Basic Information Flow

HTML → table recognition algorithm(s) → fact pool

OWL reasoner

individuals and relations from Table Ontology

text content

classifications in Product Ontology

spotter algorithms
Example (1)

facts asserted by table recognition

these are all instances of concepts of the table ontology
Example (1)

- tbl is a **table**
- r1, r2, r3 is a **row**.
  - tbl **contains** r1,r2,r3
- col1, col2 is a **column**.
  - tbl **contains** col1,col2
- c1,c2,...,c6 is a **cell**.
  - c1 hasText “**Weight**”
  - c2 hasText “**8.5 oz**”
  - c3 hasText “**Width**”
  - r1 **contains** c1,c2
- r2 **contains** c3,c4
- r3 **contains** c5,c6
- col1 **contains** c1,c3,c5
- col2 **contains** c2,c4,c6
Example (2)

facts asserted by content spotters

1) c1 hasText “Weight”
Example (2)

facts asserted by content spotters

1) c1 hasText “Weight”

SPOT: “Weight” contains the keyword kw_weight
Example (2)

facts asserted by content spotters

1) cl hasText “Weight”

2) cl contains kw_weight
Example (2)

facts asserted by content spotters

1) cl hasText "Weight"

2) cl contains kw_weight
Example (2)

facts asserted by context spotters

1) cl hasText “Weight"

2) cl contains kw_weight

Product ontology says:

WeightKeyword ≡ Cell ∩ contains.WeightKeyword
Example (2)

facts asserted by content spotters

1) cl hasText “Weight”

2) cl contains kw_weight

3) cl a WeightKeyword
Example (2)

higher level classification
Example (2)

class higher level classification

- cl a WeightKeyword
- c2 a WeightValue
- rl contains cl,c2
Example (2)

higher level classification

- c1 a WeightKeyword
- c2 a WeightValue
- r1 contains c1,c2
Example (2)

higher level classification

- c1 a WeightKeyword
- c2 a WeightValue
- rl contains c1, c2

DEF: WeightAttribute \equiv contains\ WeightKeyword \cap contains\ WeightValue

⇒ rl a WeightAttribute
Example (3)

top level classification
Example (3)

top level classification

- rl a WeightAttribute
- r2 a SizeAttribute
- tbl contains rl,r2
Example (3)

top level classification

• r1 a **WeightAttribute**
• r2 a **SizeAttribute**
• tbl contains r1,r2

**DEF:** SimpleProduct \equiv \text{contains}.\text{WeightAttribute} \cap \text{contains}.\text{SizeAttribute}
Example (3)

top level classification

- r1 a WeightAttribute
- r2 a SizeAttribute
- tbl contains r1, r2

\[ \text{DEF: } \text{SimpleProduct} \equiv \text{contains.WeightAttribute} \cap \text{contains.SizeAttribute} \]

\[ \Rightarrow \text{tbl a SimpleProduct} \]
Empirical Results - I

• groundtruth table
• manually annotated tables
• tool for marking up concepts on a webpage
• compare manual recall + accuracy

We annotated 30 tables selected randomly out of about 2000
Empirical Results - I

- groundtruth table
- manually annotated tables
- tool for marking up concepts on a webpage
- compare manual recall + accuracy

<table>
<thead>
<tr>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight 5.8 oz</td>
</tr>
<tr>
<td>Width 3.3 in</td>
</tr>
<tr>
<td>Depth 0.9 in</td>
</tr>
<tr>
<td>Height 2.2 in</td>
</tr>
<tr>
<td>Body material Stainless steel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Main Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor resolution 3.2 megapixels</td>
</tr>
<tr>
<td>Optical sensor type CCD</td>
</tr>
<tr>
<td>Effective sensor resolution 3,200,000 pixels</td>
</tr>
<tr>
<td>Gross sensor resolution 3,300,000 pixels</td>
</tr>
<tr>
<td>Optical sensor size 1/2.7 in</td>
</tr>
<tr>
<td>Light sensitivity ISO 50, ISO 100, ISO 400</td>
</tr>
</tbody>
</table>

We annotated 30 tables selected randomly out of about 2000
• groundtruth table
• manually annotated tables
• tool for marking up concepts on a webpage
• compare manual recall + accuracy

We annotated 30 tables selected randomly out of about 2000
Empirical Results - I

• ground truth
• manually annotated tables
• tool for marking up concepts on a webpage
• compare manual recall + accuracy

We annotated 30 tables selected randomly out of about 2000
Empirical Results - 2

<table>
<thead>
<tr>
<th>Ground Truth (per document)</th>
<th>Identified (per document)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keywords</td>
<td>Keywords</td>
</tr>
<tr>
<td>5.13</td>
<td>3.22</td>
</tr>
<tr>
<td>Values</td>
<td>6.78</td>
</tr>
<tr>
<td>4.82</td>
<td>4.02</td>
</tr>
<tr>
<td>Attributes</td>
<td>3.38</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Recall (per document)</th>
<th>Precision (per document)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keywords</td>
<td>Keywords</td>
</tr>
<tr>
<td>62.8%</td>
<td>65.8%</td>
</tr>
<tr>
<td>Values</td>
<td>59.3%</td>
</tr>
<tr>
<td>70.1%</td>
<td>41.6%</td>
</tr>
<tr>
<td>Attributes</td>
<td>92.4%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keywords</td>
</tr>
<tr>
<td>64.27%</td>
</tr>
<tr>
<td>Values</td>
</tr>
<tr>
<td>48.90%</td>
</tr>
<tr>
<td>Attributes</td>
</tr>
<tr>
<td>79.72%</td>
</tr>
</tbody>
</table>
Empirical Results - 2

<table>
<thead>
<tr>
<th>Ground Truth</th>
<th>Identified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keywords</td>
<td>5.13</td>
</tr>
<tr>
<td>Values</td>
<td>6.78</td>
</tr>
<tr>
<td>Attributes</td>
<td>4.82</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keywords</td>
<td>62.8%</td>
</tr>
<tr>
<td>Values</td>
<td>59.3%</td>
</tr>
<tr>
<td>Attributes</td>
<td>70.1%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keywords</td>
</tr>
<tr>
<td>Values</td>
</tr>
<tr>
<td>Attributes</td>
</tr>
</tbody>
</table>
Empirical Results - 2

<table>
<thead>
<tr>
<th>Average number of (per document)</th>
<th>Average number (per document)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ground truth</strong></td>
<td><strong>identified</strong></td>
</tr>
<tr>
<td>keywords</td>
<td>values</td>
</tr>
<tr>
<td>5.13</td>
<td>6.78</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>keywords</td>
</tr>
<tr>
<td>62.8%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>keywords</td>
</tr>
<tr>
<td>65.8%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>keywords</td>
</tr>
<tr>
<td>64.27%</td>
</tr>
</tbody>
</table>
Summary

- exploit dual interpretation of data present by making use of two ontologies

- top level strategy for bridging the gap: containment forms context

- mix procedures freely with declarative statements
summary

• **PRO**
  - declarative & procedural approaches combined
  - spotters, TR, etc nicely decoupled
  - „easy“ modifications by editing model

• **CON**
  - slow performance, reasoner optimization?
  - some concepts do not model well in OWL DL
  - rules should replace some algorithms
  - value ranges should be fuzzy
Thank You.