Automatically Discovering, Extracting and Modeling Web Sources for Information Integration

Craig A. Knoblock
University of Southern California

Joint work with
J. L. Ambite, K. Lerman, A. Plangprasopchok, and T. Russ, USC
C. Gazen and S. Minton, Fetch Technologies
M. Carman, University of Lugano
Abundance of Data, Limited Knowledge
Motivation

- **Problem**
  - Web sources and services are designed for people, not machines
  - Limited or no description of the information provided by these sources
  - This makes it hard, if not impossible to find, retrieve and integrate the vast amount of structured data available
    - *Weather sources, geocoders, stock information, currency converters, online stores, etc.*

- **Approach**
  - Start with an some initial knowledge of a domain
    - *Sources and semantic descriptions of those sources*
  - Automatically
    - *Discover related sources*
    - *Determine how to invoke the sources*
    - *Learn the syntactic structure of the sources*
    - *Build semantic models of the source*
    - *Validate the correctness of the results*
Automatically Discover and Model a Source in the Same Domain
Current Conditions Data

Seed (wunderground.com)

Washington, District of Columbia
Local Time: 1:07 PM EST — Set My Timezone
Tropical Weather: Invest 96 (North Atlantic)

Current Conditions
Eckington Pi, NE, Washington, District of Columbia (PWS)
Updated: 1:06 PM EST on November 25, 2008

- **Temp**: 48.8°F (9.3°C)
- **Humidity**: 41%
- **Wind**: 15.0 mph / 24.1 km/h
- **Visibility**: 10.0 miles
- **Dew Point**: 24°F (-4°C)
- **Wind Chill**: 3.6 mph from the WSW
- **Pressure**: 29.78 in / 1008.4 hPa (Steady)
- **Clouds**: Mostly Cloudy

Target (unisys.com)

Latest Observation for Washington, DC (20502)

- **Site**: KDCA (Washington/Nati, VA)
- **Time**: 4 PM EST 25 NOV 08
- **Temp**: 45 F (7°C)
- **Dewpt**: 22 F (-6 C)
- **Rel Hum**: 40%
- **Winds**: W at 7 Knt
- **Wind chill**: 41 F
- **Pressure**: 1010.1 mb (29.84 in)
- **Visibility**: 10 mi
- **Weather**: partly cloudy

Partial Mapping of Values
Approach

discovery

unisys

invocation & extraction

Background knowledge

Seed URL

http://wunderground.com

unisys(Zip,Temp,…) :- weather(Zip,…,Temp,Hi,Lo)

source modeling

semantic typing

unisys(Zip,Temp,…) unisys(Zip,Temp,Humidity,…)

sample input values

“90254”

sample values

patterns

domain types

definition of known sources

USC
Outline

- Discovering related sources
- Automatically invoking the sources
- Constructing syntactic models of the sources
- Determining the semantic types of the data
- Building semantic models of the sources
- Experimental Results
- Related Work
- Conclusions
Outline

• Discovering related sources
• Automatically invoking the sources
• Constructing syntactic models of the sources
• Determining the semantic types of the data
• Building semantic models of the sources
• Experimental Results
• Related Work
• Conclusions
Source Discovery

- Sources providing similar functionality are annotated with "similar" tags on the social bookmarking site del.icio.us
• **Goal**
  - Leverage user-generated tags on the social bookmarking site del.icio.us to discover sources similar to the seed

• **Approach**
  - Gather a corpus of `<user, source, tag>` bookmarks from del.icio.us
  - Use probabilistic modeling to find hidden topics in the corpus
  - Rank sources by similarity to the seed within topic space

![Diagram](Diagram.png)
Source Discovery Results

- Manually evaluated the top-ranked 100 sources
  - Number of relevant sources providing same functionality as the seed
    - Weather domain: weather conditions (wunderground seed)
    - Geospatial domain: geocodes of addresses (geocode.us seed)

The top-ranked 100 sources become the *target sources* we will try to model
Outline

- Discovering related sources
- Automatically invoking the sources
- Constructing syntactic models of the sources
- Determining the semantic types of the data
- Building semantic models of the sources
- Experimental Results
- Related Work
- Conclusions
To invoke the target source, we need to locate the form and submit it with appropriate input values

1. Locate the form
2. Try different data type combinations as input
   - For weather, only one input - location, which can be zipcode or city
3. Submit Form
4. Keep successful invocations
Invoke the Target Source with Possible Inputs

http://weather.unisys.com

Weather conditions for 20502

Unisys Weather

Latest Observation for Washington, DC (20502)

Partly Cloudy

Site: KDCA (Washington/Dulles, VA)

Time: 4 PM EST 25 Nov 08

Temp: 45°F (7°C)

Dew Pt: 22°F (-6°C)

Hum: 40%

Winds: W 7 knt

Wind chill: 41°F

Pressure: 1010.1 mb (29.84 in)

Visibility: 10 mi

Skies: partly cloudy

Weather:

Alerts

No alerts

Forecast Summary

<table>
<thead>
<tr>
<th>Day</th>
<th>Weather</th>
<th>Hi</th>
<th>Lo</th>
<th>Hi</th>
<th>Lo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wednesday</td>
<td>Sunny</td>
<td>45</td>
<td>32</td>
<td>52</td>
<td>35</td>
</tr>
<tr>
<td>Thursday</td>
<td>Sunny</td>
<td>52</td>
<td>35</td>
<td>52</td>
<td>35</td>
</tr>
<tr>
<td>Friday</td>
<td>Rainy</td>
<td>45</td>
<td>35</td>
<td>48</td>
<td>35</td>
</tr>
<tr>
<td>Saturday</td>
<td>Sunny</td>
<td>48</td>
<td>35</td>
<td>45</td>
<td>35</td>
</tr>
<tr>
<td>Sunday</td>
<td>Sunny</td>
<td>45</td>
<td>35</td>
<td>45</td>
<td>35</td>
</tr>
<tr>
<td>Monday</td>
<td>Sunny</td>
<td>45</td>
<td>35</td>
<td>45</td>
<td>35</td>
</tr>
<tr>
<td>Tuesday</td>
<td>Sunny</td>
<td>45</td>
<td>35</td>
<td>45</td>
<td>35</td>
</tr>
</tbody>
</table>

Detailed forecast from National Weather Service

District of Columbia-Armed Forces, Alexandria, including the cities of Washington, Alexandria, Falls Church 300 PM EST Tue Nov 25 2008

Detailed forecast from National Weather Service

© Unisys Corp. 2009

For questions and information on this server, NDDAPORT and WXP, contact Dan Victor at dvenor@sys.unisys.com

For sales information on Unisys weather solutions, contact Robert Benedict at robert.benedict@unisys.com

Last modified February 7, 2007
Form Input Data Model

- Each domain has an input data model
  - Derived from the seed sources
  - Alternate input groups
- Each domain has sample values for the input data types

<table>
<thead>
<tr>
<th>PR-Zip</th>
<th>PR-CityState</th>
<th>PR-City</th>
<th>PR-StateAbbr</th>
</tr>
</thead>
<tbody>
<tr>
<td>20502</td>
<td>Washington, DC</td>
<td>Washington</td>
<td>DC</td>
</tr>
<tr>
<td>32399</td>
<td>Tallahassee, FL</td>
<td>Tallahassee</td>
<td>FL</td>
</tr>
<tr>
<td>33040</td>
<td>Key West, FL</td>
<td>Key West</td>
<td>FL</td>
</tr>
<tr>
<td>90292</td>
<td>Marina del Rey, CA</td>
<td>Marina del Rey</td>
<td>CA</td>
</tr>
<tr>
<td>36130</td>
<td>Montgomery, AL</td>
<td>Montgomery</td>
<td>AL</td>
</tr>
</tbody>
</table>
• Discovering related sources
• Automatically invoking the sources
• **Constructing syntactic models of the sources**
• Determining the semantic types of the data
• Building semantic models of the sources
• Experimental Results
• Related Work
• Conclusions
• Goal:
  • Model Web sources that generate pages dynamically in response to a query

• Approach:
  • Given two or more sample pages, derive the page template
  • Use the template to extract data from the pages
Inducing Templates

• Template: a sequence of alternating **slots** and **stripes**
  • stripes are the common substrings among all pages
  • slots are the placeholders for data
• Induction: Stripes are discovered using the Longest Common Subsequence algorithm

Sample Page 1

<img src="images/Sun.png" alt="Sunny"><br>
<font face="Arial, Helvetica, sans-serif">
  <small><b>Temp: 72F (22C)</b></small><br>
  Site: <b>KSMO (Santa_Monica_Mu, CA)</b><br>
  Time: <b>11 AM PST 10 DEC 08</b>
</font>

Sample Page 2

<font face="Arial, Helvetica, sans-serif">
  <small><b>Temp: 37F (2C)</b></small><br>
  Site: <b>KAGC (Pittsburgh/Alle, PA)</b><br>
  Time: <b>2 PM EST 10 DEC 08</b>
</font>
Data Extraction with Templates

- To extract data: Find data in slots by locating the stripes of the template on unseen page:

Unseen Page

- Image: Sunny
- Font: Temp: 71F (21C)
- Small: Site: KCQT (Los_Angeles_Dow, CA)
- Time: 11 AM PST 10 DEC 08

Induced Template

- Image: Sunny
- Font: Temp: ☀ (°)
- Small: Site: ☀ (°, °)
- Time: ☀ 10 DEC 08

Extracted Data

| Sun | Sunny | 71F | 21C | KCQT | Los_Angeles_Dow | CA | 11 AM PST |
Extracting Lists

- Approach:
  - Assume items in a list are formatted using an “item” template
  - Search for “item” templates, using the DOM structure to reduce complexity

Sample Page

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FRIDAY</strong></td>
<td><strong>SATURDAY</strong></td>
</tr>
<tr>
<td>Sun</td>
<td>Rain</td>
</tr>
<tr>
<td>Sunny</td>
<td>Rainy</td>
</tr>
<tr>
<td>65</td>
<td>60</td>
</tr>
<tr>
<td>52</td>
<td>48</td>
</tr>
</tbody>
</table>

Template

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FRIDAY</strong></td>
<td><strong>SATURDAY</strong></td>
</tr>
<tr>
<td>Sun</td>
<td>Rain</td>
</tr>
<tr>
<td>Sunny</td>
<td>Rainy</td>
</tr>
<tr>
<td>65</td>
<td>60</td>
</tr>
<tr>
<td>52</td>
<td>48</td>
</tr>
<tr>
<td>Column</td>
<td>Invocation 1</td>
</tr>
<tr>
<td>--------</td>
<td>--------------</td>
</tr>
<tr>
<td>1</td>
<td>Unisys Weather: Forecast for Washington, DC (20502) [0] 2</td>
</tr>
<tr>
<td>2</td>
<td>Washington,</td>
</tr>
<tr>
<td>3</td>
<td>DC</td>
</tr>
<tr>
<td>4</td>
<td>20502</td>
</tr>
<tr>
<td>5</td>
<td>20502)</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Images/PartlyCloudy.png <strong>Image URL</strong></td>
</tr>
<tr>
<td>15</td>
<td>Partly Cloudy <strong>Good Field</strong></td>
</tr>
<tr>
<td>16</td>
<td>45</td>
</tr>
<tr>
<td>17</td>
<td>Temp: 45F (7C) <strong>Too Complex</strong></td>
</tr>
<tr>
<td>18</td>
<td>45F</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>217</td>
<td>45</td>
</tr>
<tr>
<td>218</td>
<td>MOSTLY SUNNY. HIGHS IN THE MID 40S.</td>
</tr>
</tbody>
</table>
• Discovering related sources
• Automatically invoking the sources
• Constructing syntactic models of the sources
• Determining the semantic types of the data
• Building semantic models of the sources
• Experimental Results
• Related Work
• Conclusions
Semantic Typing of Extracted Data

• **Goal:**
  • Assign semantic types to extracted data

• **Approach:** Leverage background knowledge to semantically type extracted data
  • Learn models of content from samples of known semantic types
  • Use learned models to recognize semantic types of extracted data
Learning Patterns to Recognize Semantic Types

- We developed a domain-independent token-level language to represent the structure of data as patterns
  - Token is a string or a general type
    - 90202 is a specific token
    - 5DIGIT number is a general type
  - Pattern is a sequence of tokens
    - E.g., Phone numbers

<table>
<thead>
<tr>
<th>Sample values</th>
<th>Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>310 448–8714</td>
<td></td>
</tr>
<tr>
<td>310 448–8775</td>
<td>[310 448 – 4DIGIT]</td>
</tr>
<tr>
<td>212 555–1212</td>
<td>[3DIGIT 3DIGIT – 4DIGIT]</td>
</tr>
</tbody>
</table>

- Efficiently learn patterns from examples of semantic types
- Score the match between a type (patterns) and data
Weather Data Types

Sample values

- **PR-TempF**
  88 F
  57°F
  82 F ...

- **PR-Visibility**
  8.0 miles
  10.0 miles
  4.0 miles
  7.00 mi
  10.00 mi

- **PR-Zip**
  07036
  97459
  02102

Patterns

- **PR-TempF**
  [88, F]
  [2DIGIT, F]
  [2DIGIT, °, F]

- **PR-Visibility**
  [10, ., 0, miles]
  [10, ., 00, mi]
  [10, ., 00, mi, .]
  [1DIGIT, ., 00, mi]
  [1DIGIT, ., 0, miles]

- **PR-Zip**
  [5DIGIT]
Using the Patterns for Semantic Labeling

• Use learned patterns to map new data to types in the domain model
  • Score how well patterns associated with a semantic type describe a set of examples
    • Scoring considers:
      – Number of matching patterns
      – How specific the matching patterns are
      – How many tokens of the example are left unmatched
  • Output top-scoring types
<table>
<thead>
<tr>
<th>Column</th>
<th>4</th>
<th>18</th>
<th>25</th>
<th>15</th>
<th>87</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>PR-Zip</td>
<td>PR-TempF</td>
<td>PR-Humidity</td>
<td>PR-Sky</td>
<td>PR-Sky</td>
</tr>
<tr>
<td>Score</td>
<td>0.333</td>
<td>0.68</td>
<td>1.0</td>
<td>0.325</td>
<td>0.375</td>
</tr>
<tr>
<td>Values</td>
<td>20502</td>
<td>45F</td>
<td>40%</td>
<td>Partly Cloudy</td>
<td>Sunny</td>
</tr>
<tr>
<td></td>
<td>32399</td>
<td>63F</td>
<td>23%</td>
<td>Sunny</td>
<td>Partly Cloudy</td>
</tr>
<tr>
<td></td>
<td>33040</td>
<td>73F</td>
<td>73%</td>
<td>Sunny</td>
<td>Rainy</td>
</tr>
<tr>
<td></td>
<td>90292</td>
<td>66F</td>
<td>59%</td>
<td>Partly Cloudy</td>
<td>Sunny</td>
</tr>
<tr>
<td></td>
<td>36130</td>
<td>62F</td>
<td>24%</td>
<td>Sunny</td>
<td>Partly Cloudy</td>
</tr>
</tbody>
</table>
Outline

• Discovering related sources
• Automatically invoking the sources
• Constructing syntactic models of the sources
• Determining the semantic types of the data
• Building semantic models of the sources
• Experimental Results
• Related Work
• Conclusions
Inducing Source Definitions

- Step 1: classify input & output semantic types

```
source1($zip, lat, long) :- centroid(zip, lat, long).
source2($lat1, $long1, $lat2, $long2, dist) :- greatCircleDist(lat1, long1, lat2, long2, dist).
source3($dist1, dist2) :- convertKm2Mi(dist1, dist2).
```

New Source 4

- Known Source 1
- Known Source 2
- Known Source 3

```
zipcode
```

```
distance
```

```
source4( $startZip, $endZip, separation)
```
Generating Plausible Definition

- Step 1: classify input & output semantic types
- Step 2: generate plausible definitions

source1($zip, lat, long) :-
    centroid(zip, lat, long).

source2($lat1, $long1, $lat2, $long2, dist) :-
    greatCircleDist(lat1, long1, lat2, long2, dist).

source3($dist1, dist2) :-
    convertKm2Mi(dist1, dist2).

source4($zip1, $zip2, dist) :-
    source1(zip1, lat1, long1),
    source1(zip2, lat2, long2),
    source2(lat1, long1, lat2, long2, dist2),
    source3(dist2, dist).
Top-down Generation of Candidates

Start with empty clause & generate specialisations by
1. Adding one predicate at a time from set of sources
2. Checking that each definition is:
   - Not logically redundant
   - Executable (binding constraints satisfied)

\[
\text{source5}(\_,\_,\_,\_).
\]

source5(\$zip1,\$dist1,zip2,dist2)

\[
\begin{align*}
\text{source5}(\text{zip1},\_,\_,\_) & : - \text{source4}(\text{zip1},\text{zip1},\_). \\
\text{source5}(\text{zip1},\_,\text{zip2},\text{dist2}) & : - \text{source4}(\text{zip2},\text{zip1},\text{dist2}). \\
\text{source5}(\_,\text{dist1},\_,\text{dist2}) & : - <(\text{dist2},\text{dist1}).
\end{align*}
\]
Invoke and Compare the Definition

- Step 1: classify input & output semantic types
- Step 2: generate plausible definitions
- Step 3: invoke service & compare output

source4($zip1, $zip2, dist):-
  source1(zip1, lat1, long1),
  source1(zip2, lat2, long2),
  source2(lat1, long1, lat2, long2, dist2),
  source3(dist2, dist).

source4($zip1, $zip2, dist):-
  centroid(zip1, lat1, long1),
  centroid(zip2, lat2, long2),
  greatCircleDist(lat1, long1, lat2, long2, dist2),
  convertKm2Mi(dist1, dist2).

<table>
<thead>
<tr>
<th>$zip1</th>
<th>$zip2</th>
<th>dist (actual)</th>
<th>dist (predicted)</th>
</tr>
</thead>
<tbody>
<tr>
<td>80210</td>
<td>90266</td>
<td>842.37</td>
<td>843.65</td>
</tr>
<tr>
<td>60601</td>
<td>15201</td>
<td>410.31</td>
<td>410.83</td>
</tr>
<tr>
<td>10005</td>
<td>35555</td>
<td>899.50</td>
<td>899.21</td>
</tr>
</tbody>
</table>
Approximating Equality

Allow flexibility in values from different sources

- **Numeric Types** like *distance*
  
  \[ 10.6 \text{ km} \approx 10.54 \text{ km} \]
  
  Error Bounds (eg. +/- 1%)

- **Nominal Types** like *company*
  
  Google Inc. \( \approx \) Google Incorporated
  
  String Distance Metrics
  (e.g. JaroWinkler Score > 0.9)

- **Complex Types** like *date*
  
  Mon, 31. July 2006 \( \approx \) 7/31/06
  
  Hand-written equality checking procedures.
Outline

• Discovering related sources
• Automatically invoking the sources
• Constructing syntactic models of the sources
• Determining the semantic types of the data
• Building semantic models of the sources
• Experimental Results
• Related Work
• Conclusions
Experiments: Source Discovery

- DEIMOS crawls social bookmarking site del.icio.us to discover sources similar to domain seeds:
  - Geospatial: geocoder.us
  - Weather: wunderground.com

- For each seed:
  - retrieve the 20 most popular tags users applied to this source.
  - retrieve other sources that users have annotated with that tags
  - 15 million source-user-tag triples for the domains.

- Compute similarity of resources to seed using model

- Evaluation:
  - Manually checked top-ranked 100 resources produced by model
    - same functionality if same inputs and outputs as seed
  - Among the 100 highest ranked URLs:
    - 20 relevant geospatial sources
    - 70 relevant weather sources.
Experiments: 
Source Invocation, Extraction and Semantic Typing

- **Invocation**: Recognize form input parameters and calling method
- **Extraction**: Learn extractor for resulting output
  → Then, DEIMOS can call websites programmatically as web services.
- **Semantic Typing**: automatically assign semantic types to extracted data

**Evaluation:**
- Success if extractor produces output table *and* at least one output column not part of the input can be typed
- Given top-ranked 100 URLs, DEIMOS generated
  - 2 semantically-typed geospatial sources
    Ex: ontok($Address, Longitude, Latitude, Street, StateAbbr)
  - 6 semantically-typed weather sources
    Ex. unisys($Zip, Sky, TempF, TempC, _, _, _)
Experiments: Semantic Modeling

**Semantic Modeling**: learn formal (Datalog) source descriptions based on background knowledge (known sources and types)

- Geospatial Domain

  - Background knowledge (seed source description):
    
    geocoder.us(Address, Street, City, StateAbbr, ZIP, Latitude, Longitude):-
    
    Address(Address, Street, City, StateAbbr, State, ZIP, CountryAbbr, Country, Latitude, Longitude)

  - Learned source descriptions:
    
    ontok($Address, Longitude, Latitude, _, _) :-
    
    geocoder.us(Address, _, _, _, _, Latitude, Longitude)

    geocoder.ca($Address, _, StateAbbr, Street, Latitude, _):-
    
    geocoder.us(Address, Street, _, StateAbbr, _, Latitude, _)

USC
Experiments: Semantic Modeling (Weather)

Given background source descriptions:

• `wunderground($Zip, Humidity, TempF_hi, TempF_low, TempF_hinextday, Sky, PressureInches, WindDirection) :-
  weather(Zip, TempF_hi, TempF_low, TempF_hinextday, Humidity, Sky, PressureInches, WindDirection)
• `convertC2F($TempC, TempF) :- convertTemp($TempC, TempF)

DEIMOS learned descriptions for 2 sources:

• `unisys($Zip, Sky, TempF_hi, TempC, _, _, _) :-
  weather(Zip, TempF_hi, _, _, _, Sky, _, _),
  convertTemp($TempC, TempF)

• `timetemperature($Zip, _, Sky, _, _, TempF_low, TempF_hinextday, _) :-
  weather(Zip, _, TempF_low, TempF_hinextday, _, Sky, _, _)
Experiments: Discussion (I)

+ Sound: only learned correct source descriptions
  - Using both type and value comparison make it very unlikely that an attribute would be modeled incorrectly

~ 60% attributes mapped (3/5, 4/6, 4/7, 4/8)

+ Expressive: learned conjunctive source descriptions
  - Unisys: DEIMOS uses Fahrenheit to Celsius translation function

- Can’t learn attributes not present in background sources

- Dynamic sources: Rapidly changing values, update rates
  - cannot compare temperatures if seed, target invocations too distant
  - sites reported very different humidity values
- Extraction errors => missed types
  • Ex: “<font size='1'>FL”
    • *too many spurious tokens to be considered similar to “FL”*
  • Ex: 118.440470 vs. -118.440470:
    • *extractor missed – sign, not a longitude*
  • Mixed-value columns:
    • *variable number of data items returned for different inputs can sometimes fool extractor*
    • *Ex: weather advisory attribute appears for one input and not for others \(\rightarrow\) shift in columns \(\rightarrow\) mixed value columns*

- Semantic Typing errors
  • Ex: labeled time zone codes as WindDirection due to 3caps pattern learned (WSW vs PST)

⇒ Overall, promising results
• Discovering related sources
• Automatically invoking the sources
• Constructing syntactic models of the sources
• Determining the semantic types of the data
• Building semantic models of the sources
• Experimental Results
• Related Work
• Conclusions
Related Work

ILA & Category Translation (Perkowitz & Etzioni 1995)
Learn functions describing operations on internet

- Our system learns *more complicated* definitions
  - Multiple attributes, Multiple output tuples, etc.

iMAP (Dhamanka et. al. 2004)
Discovers complex (many-to-1) mappings between DB schemas

- Our system learns *many-to-many* mappings
- Our approach is more general
- We deal with problem of invoking sources
Related Work

- Metadata-based classification of data types used by Web services and HTML forms (Hess & Kushmerick, 2003)
  - Naïve Bayes classifier
  - No invocation of services

- Woogle: Metadata-based clustering of data and operations used by Web services (Dong et al, 2004)
  - Groups similar types together: Zipcode, City, State
  - Cannot invoke services with this information
Outline

• Discovering related sources
• Automatically invoking the sources
• Constructing syntactic models of the sources
• Determining the semantic types of the data
• Building semantic models of the sources
• Experimental Results
• Related Work
• Conclusions
Assumption: overlap between new & known sources
Nonetheless, the technique is widely applicable:

- Redundancy
- Scope or Completeness
- Binding Constraints
- Composed Functionality
- Access Time
Discussion

- Integrated approach to discovering and modeling online sources and services:
  - Discover new sources
  - How to invoke a source
  - Discovering the template for the source
  - Finding the semantic types of the output
  - Learning a definition of what the service does

- Provides an approach to generate source descriptions for the Semantic Web
  - Little motivation for providers to annotate services
  - Instead we can generate metadata automatically
Future Work

• Scalability!
  • Difficult to invoke sources with many inputs
    • Hotel reservation sites
  • Hard to learn sources that have many attributes
    • Some weather sources could have 40 attributes

• Learning beyond the domain model
  • Learn new semantic types
  • Learn new source attributes
  • Learn new source relations
  • Learn the domain and range of the sources