Learning Semantic Descriptions of Web Information Sources

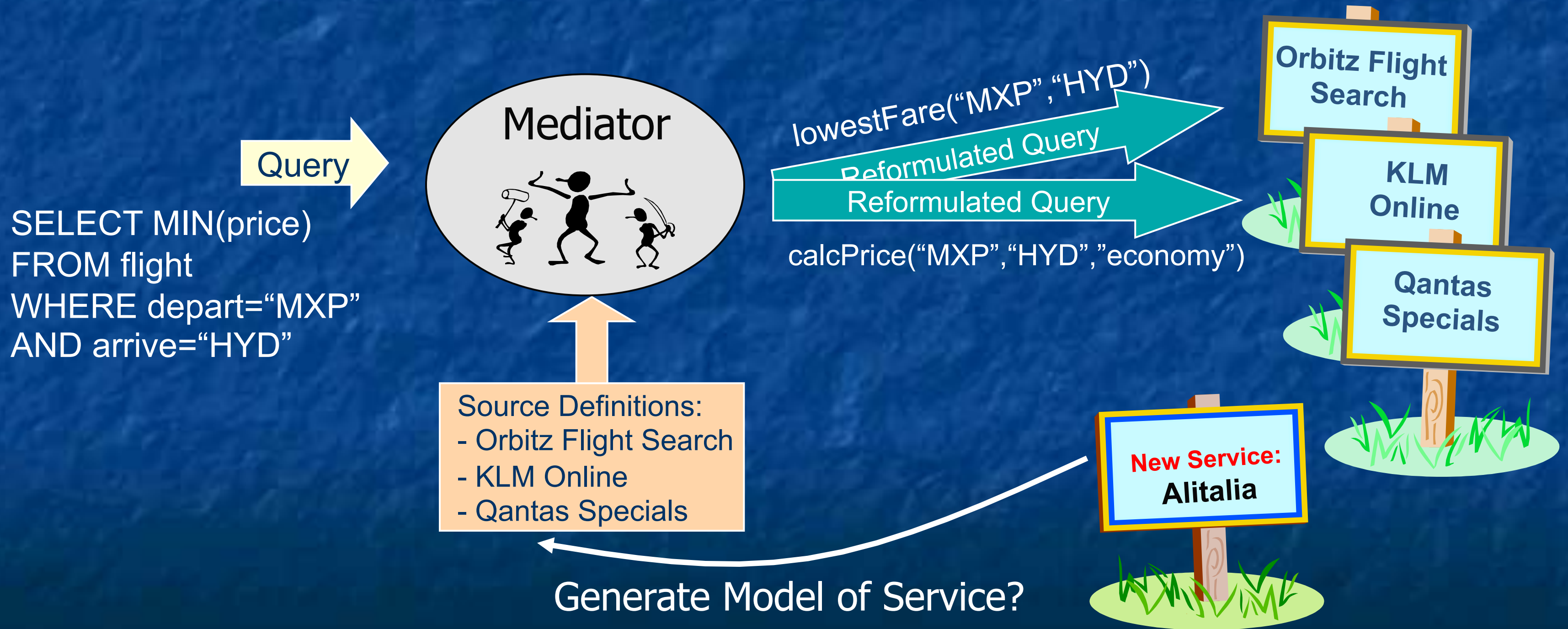
Mark James Carman*
Information Sciences Institute / USC
&
ITC-irst / University of Trento

Craig A. Knoblock
Information Sciences Institute / USC

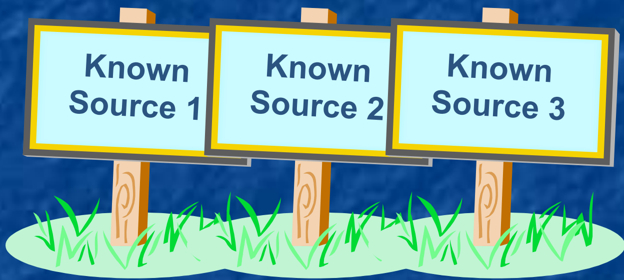
* I am currently seeking a Postdoc position in Europe somewhere near northern Italy …

Mediators & Source Definitions

- Explosion of online information sources
- Mediators run queries over multiple sources
- Require declarative source definitions
- New service → model it automatically?



Modeling Sources: an Example



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

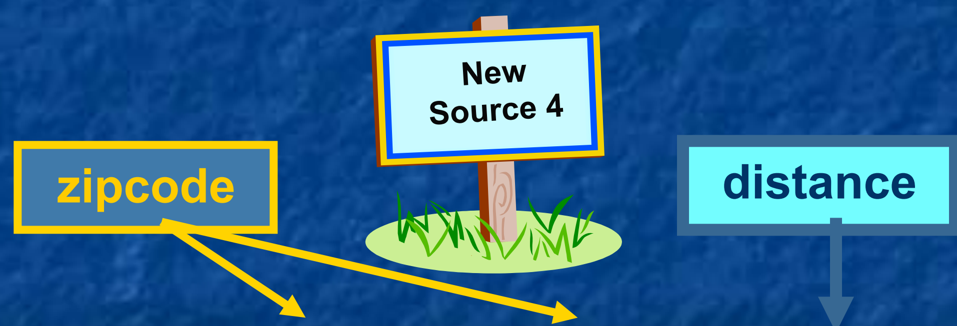
source2($lat1, $long1, $lat2, $long2, dist) :-
    greatCircleDist(lat1, long1, lat2, long2, dist).

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

Step 1:

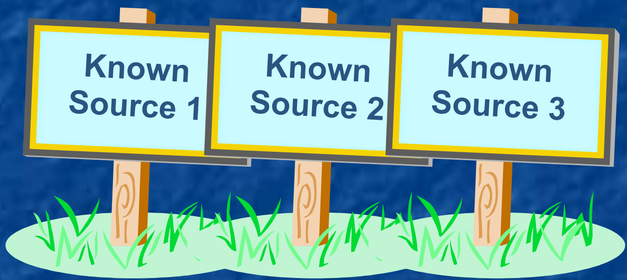
classify input & output semantic types, using:

- Metadata (labels)
- Data (content)



source4($startZip, $endZip, separation)

Modeling Sources: Step 2



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).

Step 2:

model functionality by:

- generating plausible definitions

source4($zip1, $zip2, dist) :-

centroid(zip1, lat1, long1),
centroid(zip2, lat2, long2),
greatCircleDist(lat1, long1, lat2, long2, dist2),
convertKm2Mi(dist1, dist2).

source1(zip1, lat1, long1),
source1(zip2, lat2, long2),
source2(lat1, long1, lat2, long2, dist2),
source3(dist2, dist).

Modeling Sources: Step 2

Step 2:

model functionality by:

- generating plausible definitions
- comparing the output they produce

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

match

$zip1	$zip2	dist *(actual)*	dist *(predicted)*
80210	90266	842.37	843.65
60601	15201	410.31	410.83
10005	35555	899.50	899.21

Summary - Modeling Sources

Step 1: Semantic Labeling

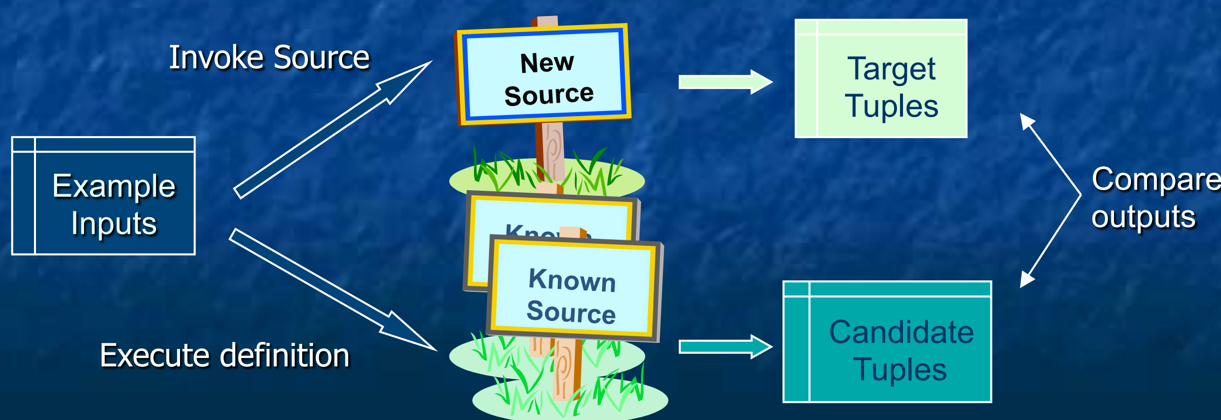
Classify input & output *semantic types*, using:

- Labels: metadata
- Content: output data

Step 2: Functional Modeling

Model the *functionality* of service by:

- Search: generating plausible definitions
- Scoring: compare the output they produce



Summary - Modeling Sources

Step 1: Semantic

Cl… , using:

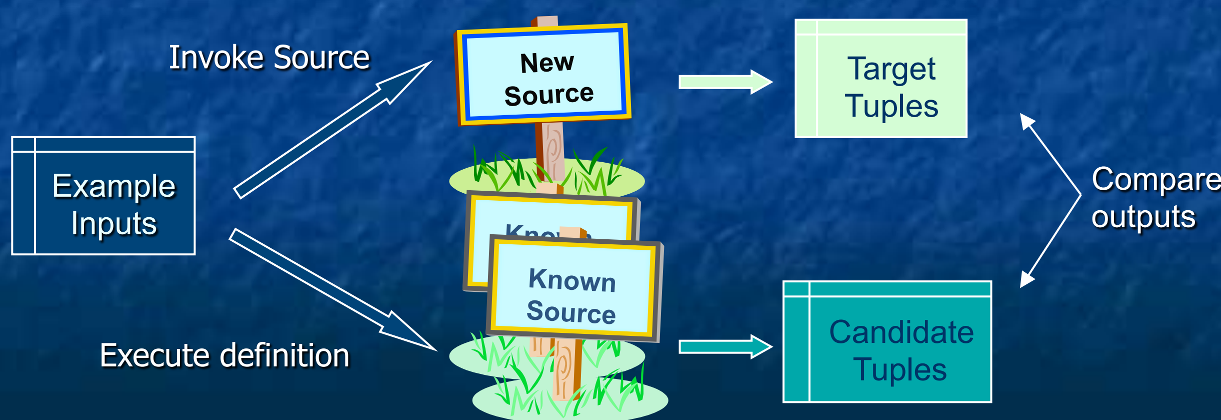
-
- … put data

Previous Work!
Lerman, Plangprasopchok and Knoblock.
Automatically labeling data used by web services.
AAAI'06.

Step 2: Functional Modeling

Model the *functionality* of service by:

- Search: generating plausible definitions
- Scoring: compare the output they produce



Searching for Definitions

- Search space of *conjunctive queries:*
  target($\underline{X}$) :- source1($\underline{X}_1$), source2($\underline{X}_2$), …

Expressive Language
Sufficient for modeling most online sources

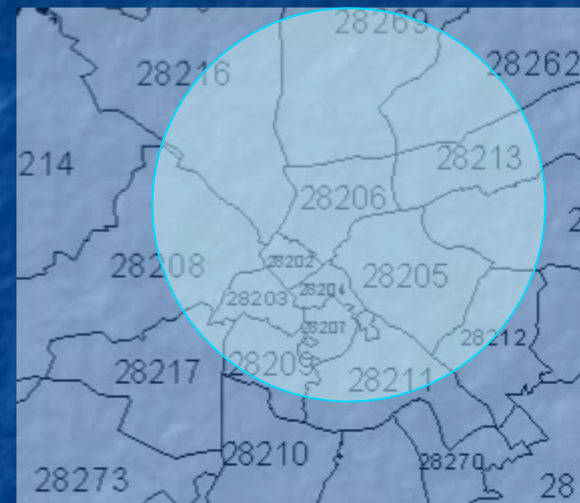
1. Sample the new source

Invoke ***target*** with set of random inputs;
Add empty clause to ***queue***;

while (***queue*** not empty)
    v := best definition from ***queue***;
    forall (***v'*** in **Expand**(***v***))
        if (**Eval**(***v'***) > **Eval**(***v***))
            insert ***v'*** into ***queue***;

2. Best-first search through space of candidate definitions

Invoking the Target



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

Invoke source with randomly generated tuples

- Use distribution if available
- If no output is produced try invoking other sources

randomly generated input tuples

Input <zip1, dist1>	Output <zip2, dist2>
<07307, 50.94>	{<07097, 0.26>, <07030, 0.83>, <07310, 1.09>, ...}
<60632, 10874.2>	{}

Non-empty Result

Empty Result

Top-down Generation of Candidates

Start with empty clause & specialize it by:

- Adding a predicate from set of sources
- Check that definition is not redundant

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

source5(_,_,_,_).

Expand

source5(zip1,_,_,_) :- **source4**(zip1,zip1,_).
source5(zip1,_,zip2,dist2) :- **source4**(zip2,zip1,dist2).
source5(_,dist1,_,dist2) :- **<**(dist2,dist1).
…

Best-first Enumeration of Candidates

Evaluate clauses & expand the best one

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

source5(_,_,_,_).

Expand

source5(zip1,_,_,_) :- **source4**(zip1,zip1,_).
source5(zip1,_,zip2,dist2) :- **source4**(zip2,zip1,dist2).
source5(_,dist1,_,dist2) :- **<**(dist2,dist1).
…

Expand

source5(zip1,dist1,zip2,dist2) :- **source4**(zip2,zip1,dist2), **source4**(zip1,zip2,dist1).
source5(zip1,dist1,zip2,dist2) :- **source4**(zip2,zip1,dist2), **<**(dist2,dist1).
…

Limiting the Search

Extremely Large Search space!

- Constrained by use of Semantic Types
- Limit search by:
  - Maximum Clause length
  - Maximum Predicate Repetition
  - Maximum Number of Existential Variables
  - Definition must be Executable
  - Maximum Variable Repetition within Literal

Standard techniques

Non-standard technique

Scoring Candidates

Need to score candidates to direct best-first search

- Score definitions based on overlap

Input <$zip1, $dist1>	**Target Output** <zip2, dist2>	**Clause Output** <zip2, dist2>
<60632, 874.2>	{}	{<60629, 2.15>, <60682, 2.27>, <60623, 2.64>, ..}
<07307, 50.94>	{<07097, 0.26>, <07030, 0.83>, <07310, 1.09>, ...}	{}
<28041, 240.46>	{<28072, 1.74>, <28146, 3.41>, <28138, 3.97>,...}	{<28072, 1.74>, <28146, 3.41>}

No Overlap

No Overlap

Overlap!

Scoring Candidates II

Sources may return multiple tuples and not be complete:

- Use Jaccard similarity as fitness function
- Average results across different inputs

At least half of input tuples are non-empty invocations of target

forall (tuple in **InputTuples**)

T_target = **invoke**(target, tuple)
T_clause = **execute**(clause, tuple)

if not (|*T_target*|=0 and |*T_clause*|=0)

$$fitness = \frac{|T\_target \cap T\_clause|}{|T\_target \cup T\_clause|}$$

Jaccard similarity

Average results only when output is returned

return average(*fitness*)

Approximating Equality

Allow flexibility in values from different sources

- Numeric Types like *distance*

  10.6 km ≈ 10.54 km

  Error Bounds (eg. +/- 1%)

- Nominal Types like *company*

  Google Inc. ≈ Google Incorporated

  String Distance Metrics (e.g. JaroWinkler Score > 0.9)

- Complex Types like *date*

  Mon, 31. July 2006 ≈ 7/31/06

  Hand-written equality checking procedures.

Experimental Setup

- 25 problems
- 35 known sources
- All real services
- Time limit of 20 minutes

Inductive search bias:
- Max clause length: 7
- Predicate repetition: 2
- Max variable level: 5
- Executable candidates
- No variable repetition

Equality Approximations:
- 1% for *distance*, *speed*, *temperature* & *price*
- 0.002 degrees for *latitude & longitude*
- JaroWinkler > 0.85 for *company*, *hotel & airport*
- hand-written procedure for *date*.

Actual Learned Examples

1 GetDistanceBetweenZipCodes($zip0, $zip1, dis2):-
GetCentroid(zip0, lat1, lon2), **GetCentroid**(zip1, lat4, lon5),
GetDistance(lat1, lon2, lat4, lon5, dis10), **ConvertKm2Mi**(dis10, dis2).

2 USGSElevation($lat0, $lon1, dis2):-
ConvertFt2M(dis2, dis1), **Altitude**(lat0, lon1, dis1).

Distinguished forecast from current conditions

3 YahooWeather($zip0, cit1, sta2, , lat4, lon5, day6, dat7,tem8, tem9, sky10) :-
WeatherForecast(cit1,sta2,,lat4,lon5,,day6,dat7,tem9,tem8,,,sky10,,,),
GetCityState(zip0, cit1, sta2).

current price = yesterday's close + change

4 GetQuote($tic0,pri1,dat2,tim3,pri4,pri5,pri6,pri7,cou8,,pri10,,,pri13,,com15) :-
YahooFinance(tic0, pri1, dat2, tim3, pri4, pri5, pri6,pri7, cou8),
GetCompanyName(tic0,com15,,),**Add**(pri5,pri13,pri10),**Add**(pri4,pri10,pri1).

5 YahooAutos($zip0, $mak1, dat2, yea3, mod4, , , pri7,) :-
GoogleBaseCars(zip0, mak1, , mod4, pri7, , , yea3),
ConvertTime(dat2, , dat10, ,), **GetCurrentTime**(, , dat10,).

Experimental Results

Overall Results:

- Average Precision: 88%
- Average Recall: 69%

Results for different domains:

Problem Domain	# of Problems	Avg. # of Candidates	Avg. Time (s)	Avg. Precision	Avg. Recall
Geospatial	9	136	303	100%	84%
Financial	2	1606	335	56%	63%
Weather	8	368	693	91%	62%
Hotels	4	43	374	90%	60%
Cars	2	68	940	50%	50%

Related Work

Semantic Labeling:

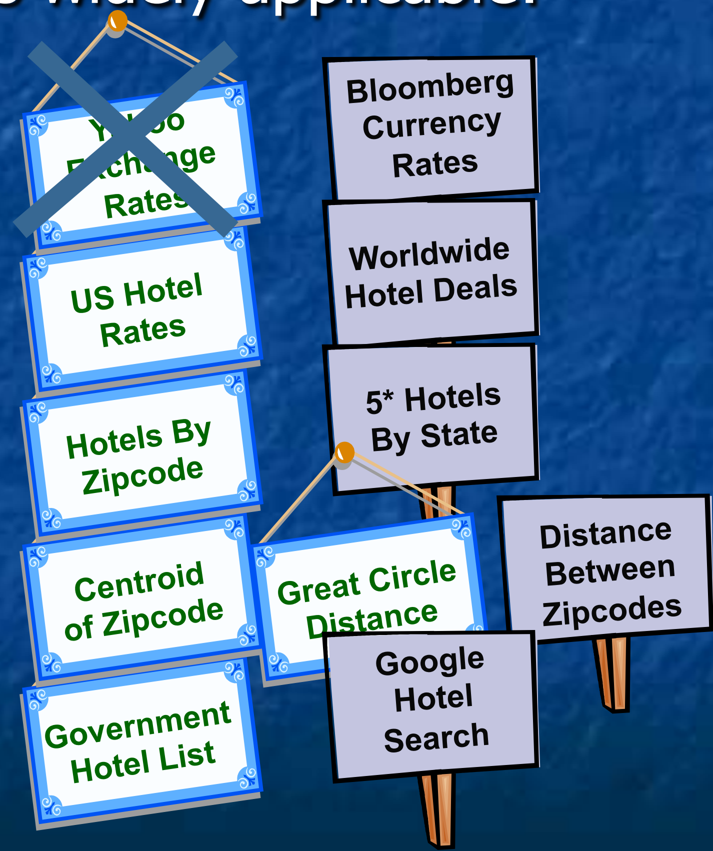
- Metadata-based service classification (Hess & Kushmerick, '03)
- Woogle: Web Service clustering (Dong et al, 2004)
  - Neither system produces sufficient information for integration

Functional Modeling:

- Category Translation (Perkowitz & Etzioni 1995)
  - Less complicated (single input, single output) definitions.
- iMAP: Complex schema matcher (Dhamanka et. al. 2004)
  - Many-to-1 not many-to-many mappings
  - Type-specific search algorithms
  - Not designed for live information sources

Conclusions

- Assumption: overlap between new & known sources
- Technique is nonetheless widely applicable:
  - Redundancy
  - Scope or Completeness
  - Binding Constraints
  - Composed Functionality
  - Access Time



Conclusions

- Integrated approach for learning:
  - *How to invoke a web service (inputs & outputs)*
  - *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 generate metadata automatically
- Provides approach to discover new sources of data automatically