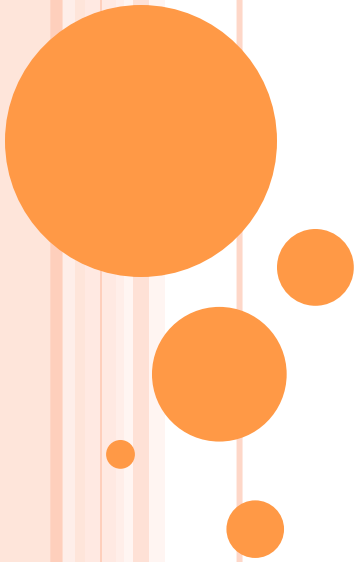

Supporting the Rapid Construction of Mashups:

Automatically Identifying Data Sources

Tim Cheeseman

A Presentation of The Following Research Paper for CS 690 Week 9

- Mark James Carman and Craig A. Knoblock. Learning semantic descriptions of web information sources, In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

Information All Around Us

- The Internet has a plethora of information available in multiple different forms:
  - Web Services
  - RSS Feeds
  - Raw Data (e.g. www.data.gov)
- Do we really want to manually write semantic descriptions for every service we want to use?
  - This is extremely tedious and error-prone work.
  - How could we do this automatically?

Learning From Experience

- We already have many data sources that have been semantically modeled by hand.
- Different sources will very often provide similar or overlapping data.
- Can we use our knowledge of previously modeled data sources to identify newly discovered ones?

An Example

- Assume we know about three sources (defined in Datalog):
  - 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).
- Given another source, can we determine what it does automatically?
  - source4($zip, $zip, distance)

An Example - Continued

- We can arrive at the output of distance from the input of two zip codes by combining calls to our known sources:
  - source4($zip1, $zip2, dist):-
    source1(zip1, lat1, long1),
    source1(zip2, lat2, long2),
    source2(lat1, long1, lat2, long2, dist2),
    source3(dist2 dist).

- We then compare the output of our combined calls to the output of source4, and compare the results.

An Example - Continued

- If the results are similar, then we have defined the source as finding the distance in miles from the centroids of two zip codes. If not, we can still try without using source3 to convert.

- Possible combinations are tested and scored until the outputs are “sufficiently similar” or all combinations have been exhausted.

Problems

- This is a top-down search algorithm and the search space can grow very large. We can limit this using know techniques:
  - Maximum clause length
  - Maximum predicate repetition
  - Maximum existential quantification level
  - Definitions must be executable
  - No repetition of variables allowed within a literal
- Partial definitions can be scored unfairly high, so we need to penalize these scores based on the size of the domain of each missing attribute.

Example Results

- GetDistanceBetweenZipCodes($zip0, $zip1, dis2) :-
  GetCentroid(zip0, lat1, lon2),
  GetCentroid(zip1, lat4, lon5),
  GetDistance(lat1, lon2, lat4, lon5, dis10),
  ConvertKm2Mi(dis10, dis2).
- USGSElevation($lat0, $lon1, dis2) :-
  ConvertFt2M(dis2, dis1), Altitude(lat0, lon1, dis1).
- GetQuote($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).