An Interactive Clustering-based Approach to Integrating Source Query Interfaces on the Deep Web

Presented by Yingying Tao

Motivation

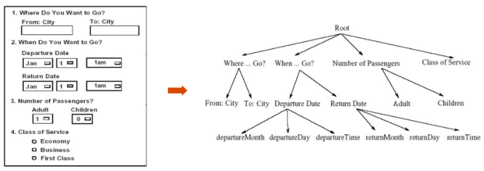
- Large number of data sources on web are hidden behind query interfaces
  - User has to access each source individually
  - A unified query interface is required for integration
- Limitations of current solutions
  - Flat schema ➡ Hierarchical model
  - 1:1 mapping ➡ 1:m mapping
  - Black-box fashion ➡ User interaction
  - Laborious parameter tuning ➡ Parameter learning

Hierarchical Modeling

- Query interface in HTML forms is consisted by *fields*
  Text input box, selection lists, check box, etc.
- Each field contains three properties:
  - *Name :* id of the field
    - concatenated/abbreviated words
  - *Label :* description of the field
    - ordinary words, can be absent
  - *Domain :* set of valid values the filed may take

Hierarchical Modeling

- Hierarchical schema – ordered tree
  - Leaf element : field in the interface
  - Internal element : group/super-group of fields
  - Sibling elements : elements with same parent



Interface Matching

- Interface matching – identify semantically similar fields over different query interfaces

  1:1 mapping vs. 1:m mapping
- Challenges and solutions
  - 1:1 mapping

    Label mismatch problem ➡ Bridging approach

    $a \leftrightarrow b$ & $b \leftrightarrow c$ $\Rightarrow$ $a \leftrightarrow c$
  - 1:m mapping –

    more complex ➡ field matching via clustering

Interface Matching via Clustering

- Field similarity function

  For two field ***e*** and ***f*** in different interface, their similarity

  AS(e,f) = λ_{ls}* linguistic_sim(e,f) + λ_{ds} domain_sim(e,f)

$$\lambda_n * \underbrace{\mathrm{nSim}(e,f)}_{\text{name similarity}} + \lambda_l * \underbrace{\mathrm{lSim}(e,f)}_{\text{label similarity}} + \lambda_{nl} * \underbrace{\mathrm{nlSim}(e,f)}_{\text{name vs. label similarity}}$$

$$\lambda_t * \mathrm{typeSim}(d,d') + \lambda_v * \mathrm{valueSim}(d,d')$$

d and d' – domain of field e and f

Interface Matching via Clustering

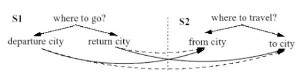
- Finding 1:1 mappings – Greedy matching

```
Cluster(S, M, τc) → P:
(1) place each field in S in a cluster by itself.
(2) while there are two clusters with similarity > τc,
    (a) choose two clusters, ci and cj, whose similarity
        is the largest over all pairs of clusters.
    (b) resolve the ties if necessary.
    (c) merge ci and cj into a new cluster ck, and
        remove clusters ci and cj.
    (d) remove all rows and columns associated with ci and
        cj in M, and add a new row and column for ck.
    (e) compute similarities of ck with other clusters
        using Formula 4.
(3) return the clusters of fields.
```

Interface Matching via Clustering

- Breaking tie – more than one pair with same max similarity

  ➡ Select the first best choice



Question: how to determine the order of fields?

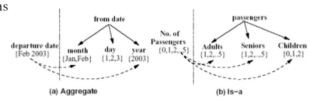
Interface Matching via Clustering

- Finding 1:m mappings – three phases
  - Preliminary 1-m matching phase
  - Clustering phase
  - Final 1-m matching phase

```
FieldMatch(S) → P and Q'
(1) /* Preliminary-1-m-matching phase: */
    Q ← IdentifyInitialOneToManyMappings(S)
(2) /* Clustering phase: */
    (a) /* compute pairwise aggregate similarities of fields */
        M ← ComputeAggregateSimilarities(S)
    (b) /* identify 1:1 mappings via clustering */
        P ← Cluster(S,M,τc)
(3) /* Final-1-m-matching phase:
    combine P and Q to obtain final 1:m mappings */
    Q' ← ObtainFinalOneToManyMapping(P, Q)
```

Interface Matching via Clustering

- Identify preliminary 1:m mappings
  - Aggregate type
  - Is-a type
  - Infinite domains



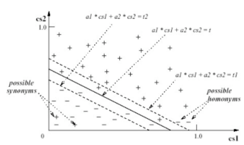
- Obtain final 1:m mappings
  - Bridging approach:
  - $a \leftrightarrow \{b_1,b_2\}$ & $b_1 \leftrightarrow c_1$, $b_2 \leftrightarrow c_2$ $\Rightarrow$ $a \leftrightarrow \{c_1,c_2\}$

User Interactions and Parameter Learning

- Parameter learning – learning the threshold

  Observation:
  - Matching fields typically have at least one large component similarities
  - Non-matching fields normally have small similarities in both components

Approach:
Finding the gap



User Interactions and Parameter Learning

- User Interaction – resolving uncertainties
  - Determine possible homonyms
    - High linguistic similarity but low domain similarity
  - Determine possible synonyms
    - Check-Ask-Merge procedure
  - Determine Possible 1:m mappings

Experiments

- Data set
  - 5 domains, 20 query interfaces for each
  - Manually transformed into schema trees
  - All weight coefficients based on observation
- Performance Measurement
  - Precision (P)
  - Recall (R)
  - F-measure (F)

Experiments

- Experimental results
  - Automatic field matching accuracy

    Threshold set to zero

    Average P – 88.2%, R – 91.1%, F – 89.5%
  - Threshold learning results

    Average P – 95.2%, R – 88.0%, F – 91.3%

    Larger threshold will lead to higher precision but lower recall
  - User interaction results

    Average P – 96.0%, R – 94.0%, F – 94.8%

Experiments

- Component contribution
  - 1:m mappings
  - Instance information
  - Tie resolution

Domain	None			No 1:m Handling			No Instances			No Tie Res.			All		
	Prec	Rec	F	Prec	Rec	F	Prec	Rec	F	Prec	Rec	F	Prec	Rec	F
Airfare	81.0	66.9	73.3	93.0	81.8	87.0	82.2	83.4	82.8	84.9	87.4	86.1	92.0	90.7	91.4
Automobile	90.1	88.8	89.5	92.7	91.2	92.0	88.9	88.5	88.7	92.8	92.3	92.6	92.8	92.3	92.6
Book	97.7	86.8	91.9	93.5	92.0	92.8	97.7	87.2	92.1	93.5	92.5	93.0	93.5	92.5	93.0
Job	79.1	74.7	76.8	81.6	81.0	81.3	79.7	77.2	78.4	81.8	83.5	82.6	81.8	83.5	82.6
Real Estate	77.8	75.4	76.6	79.8	81.3	80.6	79.6	92.2	85.5	80.1	96.0	87.3	81.0	96.7	88.1
Average	85.1	78.5	81.6	88.1	85.5	86.7	85.6	85.7	85.5	86.6	90.3	88.3	88.2	91.1	89.5

Question: Under what circumstances 1:m mapping may have a worse performance?

Conclusions and Future Work

- Conclusions
  - Flat schema vs. schema tree
  - 1:1 mapping vs. 1:m mapping
  - Blackbox vs. user interaction
  - Threshold tuning vs. threshold learning
- Future work
  - Automatically generating schema trees
  - Better solutions for breaking the tie
  - Self-learning on weight coefficients

Discussions

- Effectiveness vs. efficiency?
- Depth of the schema tree: what's the purpose?
- Transitivity of the bridging approach?
- How to handle dynamic query interfaces?
- How to determine the weight coefficients?
- How to define the order of fields for breaking the tie?
- When will the 1:m mapping approach has a worse performance?