Similarity Flooding: A versatile Graph Matching Algorithm and its Application to Schema Matching

Sergey Melnik, Hector Garcia-Molina (Stanford University) ,
and Erhard Rahm (University of Leipzig), ICDE 2002

Presented by:
George Beskales

Introduction to Schema Matching Problem

- The Goal is to map certain schema elements to other schema
- Applications:
  - Schema Integration / mediated schemas
  - Translate data between multiple databases
  - Databases Consolidation

Why schema matching is difficult?

- Imprecise wording, e.g. contact-info
- Different Ontology, e.g. ‘Load’ in electrical and mechanical contexts
- Schema and data maybe insufficient.
- Documentations and original schema designers are usually no available
- Matching decisions are highly subjective

Approaches

- Learning-based Approach
- Rules-based Approach
- Information Retrieval Approach

Most solutions requires user intervention either at the beginning (in case of learning) or after creating a mapping (correcting)

Similarity Flooding

- Based on Schema elements matching. (vs. schema + instances matching)
- Based on schema structure (vs. elements level matching)
- Each Schema is represented as a directed graph
- Based on the assumption :
  - Whenever any two elements in the graphs G_1 and G_2 are similar, their neighbors tend to be similar.

Similarity Flooding

CREATE TABLE *Personnel*(
- *Pno* int,
- *Pname* string,
- *Dept* string,
- *Born* date,
- UNIQUE *pkey*(*Pno*)

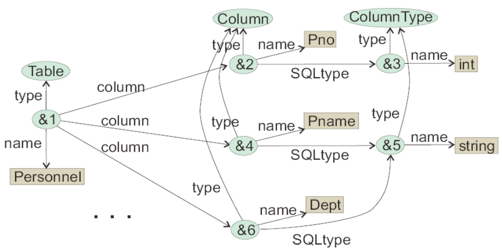
Schema 1

CREATE TABLE *Employee* (
- *EmpNo* int PRIMARY KEY,
- *EmpName* varchar(50),
- *DeptNo* int REFERENCES
- *Department*,
- *Salary* dec(15,2),
- *Birthdate* date
-)

CREATE TABLE *Department* (
- *DeptNo* int PRIMARY KEY,
- *DeptName*

Schema 2

Equivalent Graph Representation



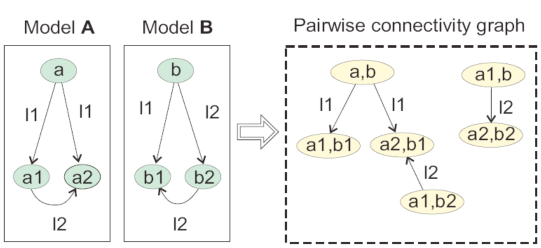
SF algorithm

1- Initial Mapping is made by simple string matching between graphs nodes.

Line#	Similarity	Node in G_1	Node in G_2
1.	1.0	Column	Column
2.	0.66	ColumnType	Column
3.	0.66	'Dept'	'DeptNo'
4.	0.66	'Dept'	'DeptName'
5.	0.5	UniqueKey	PrimaryKey
6.	0.26	'Pname'	'DeptName'
7.	0.26	'Pname'	'EmpName'
8.	0.22	'date'	'Birthdate'
9.	0.11	'Dept'	'Department'
10.	0.06	'int'	'Department'

SF algorithm (cont'd)

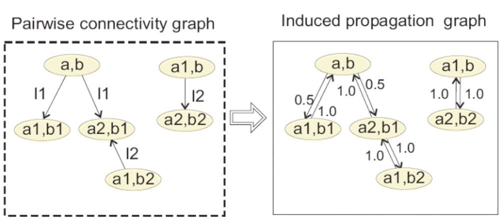
2- Similarity Flooding



Assumptions

- Each edge type has the same contribution = 1.0
- Similarity contribution for edges with the same type outgoing from one node is evenly distributed

SF algorithm (cont'd)



SF algorithm (cont'd)

Propagation Rule :

$$\sigma^{i+1}(x,y) = \sigma^i(x,y) + \sum_{(a_u,p,x)\in A,\ (b_u,p,y)\in B} \sigma^i(a_u,b_u)\cdot w((a_u,b_u),(x,y)) + \sum_{(x,p,a_v)\in A,\ (y,p,b_v)\in B} \sigma^i(a_v,b_v)\cdot w((a_v,b_v),(x,y))$$

- Normalize similarity values after each iteration
- Stopping condition : $\Delta(\sigma^n, \sigma^{n-1}) < \varepsilon$
- Convergence can be guaranteed when all initial similarities for all pairs > 0

SF algorithm (cont'd)

- 3- Filters

The goal of filtering is to choose the best match candidates from the output list

- Application-Specific Constraints filter : e.g. cardinality constraint
- Selection Metrics : e.g. stable marriage ,maximal matching, assignment problem
- Selection Threshold ($0 < t_{rel} \leq 1$)

Matching Quality

- Accuracy : how much effort is needed to convert the output matching pairs to the intended one, i.e. the number of removing false positives and adding false negatives
- $$Accuracy = 1 - \frac{(n-c)+(m-c)}{m}$$

n : number of returned matches

m : number of intended matches

c : number of returned intended matches

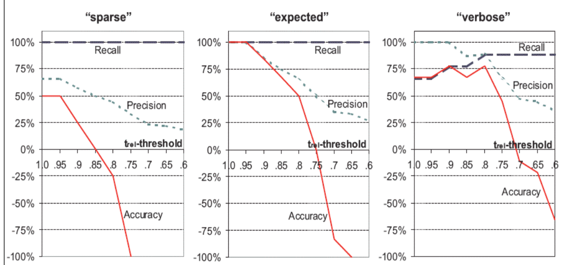
Intended Results Specification

Intended match result is categorized into 3 types:

- Sparse
- Expected
- Verbose

Sparse	Expected	Verbose	Node in G_1	Node in G_2
	+	+	[Table: Personnel]	[Table: Employee]
		+	[Table: Personnel]	[Table: Department]
	+	+	[UniqueKey: perskey]	[PrimaryKey: on EmpNo]
+	+	+	[Col: Personnel/Dept]	[Col: Department/DeptName]
		+	[Col: Personnel/Dept]	[Col: Department/DeptNo]
		+	[Col: Personnel/Dept]	[Col: Employee/DeptNo]
+	+	+	[Col: Personnel/Pno]	[Col: Employee/EmpNo]
+	+	+	[Col: Personnel/Pname]	[Col: Employee/EmpName]
+	+	+	[Col: Personnel/Born]	[Col: Employee/Birthdate]

Results (cont'd)



Convergence Speed

Identifier	Fixpoint formula
Basic	$\sigma^{i+1} = normalize(\sigma^i + \varphi(\sigma^i))$
A	$\sigma^{i+1} = normalize(\sigma^0 + \varphi(\sigma^i))$
B	$\sigma^{i+1} = normalize(\varphi(\sigma^0 + \sigma^i))$
C	$\sigma^{i+1} = normalize(\sigma^0 + \sigma^i + \varphi(\sigma^0 + \sigma^i))$

Formula	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	Total
A (as is)	18	48	122	78	∞	12	37	25	25	∞
A (strongly connected)	15	56	89	81	1488	18	48	25	31	1851
B (as is)	8	428	17	39	8	13	10	24	21	568
B (strongly connected)	7	268	21	32	13	15	14	21	53	444
C (as is)	7	9	9	11	7	7	9	10	9	78
C (strongly connected)	7	9	8	11	7	5	9	7	9	72

Strongly connected : $\sigma^0(x,y)>0$, for all $x \in G_1$, $y \in G_2$

Pros

- Innovative method for quality matching.
- General Model, provided that there is a straightforward method to map the used schema to a graph.
- No learning phase is required before use
- Flexibility in filters to suits specific application constrains

Cons

- Weak basis for similarity propagation
- Estimation errors can also be propagated to neighbors
- Flooding techniques are usually slow. Not practical for large number of elements
- Initial similarities have huge impact on the output quality and the convergence speed. Which returns us to the first square : how to get good matching?
- Heterogeneous sources can be problematic when mapped to graphs
- Does not utilize data instances in the graph
- Unable to detect complex relations between elements

Recommendations

- Extension to N graph
- Adding more powerful more enhanced matcher to initialize similarities
- Adding sample of data to the graph and use instance matchers (e.g. format matcher) as initial similarity
- More useful in case of hierarchical schemas (e.g. XML)

Recommendations (cont'd)

- Restrict the graph to a hierarchical structure
  - The similarity will propagate in bottom-up and top-down fashions
  - Similarity propagation is much reasonable in case of parent/child relations
  - The performance and the convergence speed is improved due to limited propagation paths
  - Can fit most practical schemas, e.g., SQL /XML.

Open Questions

- How to improve convergence speed?
- Can a directed (ordered) flooding affect the convergence or matching quality?
- How to extend the model to N graph?
- Will using sample instances increase the matching quality? to what extent? at what cost?

Conclusion

- Similarity Flooding is a structural based approach that fits various data sources
- Generality is chosen over performance
- Useful for specific contexts (e.g. no complex relations)
- Further investigation is required to improve matching quality and convergence speed