Record Linkage: Similarity Measures and Algorithms

Nick Koudas (University of Toronto)
Sunita Sarawagi (IIT Bombay)
Divesh Srivastava (AT&T Labs-Research)

Presenters

- U. Toronto

- IIT Bombay

- AT&T Research

Outline

- Part I: Motivation, similarity measures (90 min)
  - Data quality, applications
  - Linkage methodology, core measures
  - Learning core measures
  - Linkage based measures
- Part II: Efficient algorithms for approximate join (60 min)
- Part III: Clustering/partitioning algorithms (30 min)

Data Quality: Status

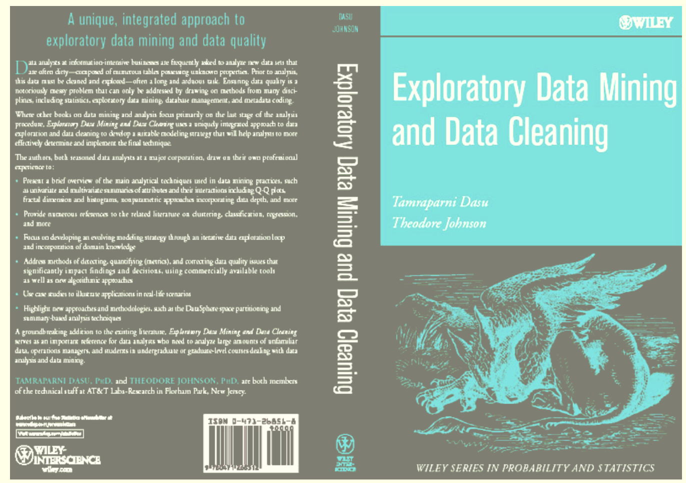
- Pervasive problem in large databases
  - Inconsistency with reality: 2% of records obsolete in customer files in 1 month (deaths, name changes, etc) [DWI02]
  - Pricing anomalies : UA tickets selling for $5, 1GB of memory selling for $19.99 at amazon.com

- Massive financial impact
  - $611B/year loss in US due to poor customer data [DWI02]
  - $2.5B/year loss due to incorrect prices in retail DBs [E00]

- Commercial tools: specialized, rule-based, programmatic

How are Such Problems Created?

- Human factors
  - Incorrect data entry
  - Ambiguity during data transformations
- Application factors
  - Erroneous applications populating databases
  - Faulty database design (constraints not enforced)
- Obsolence
  - Real-world is dynamic

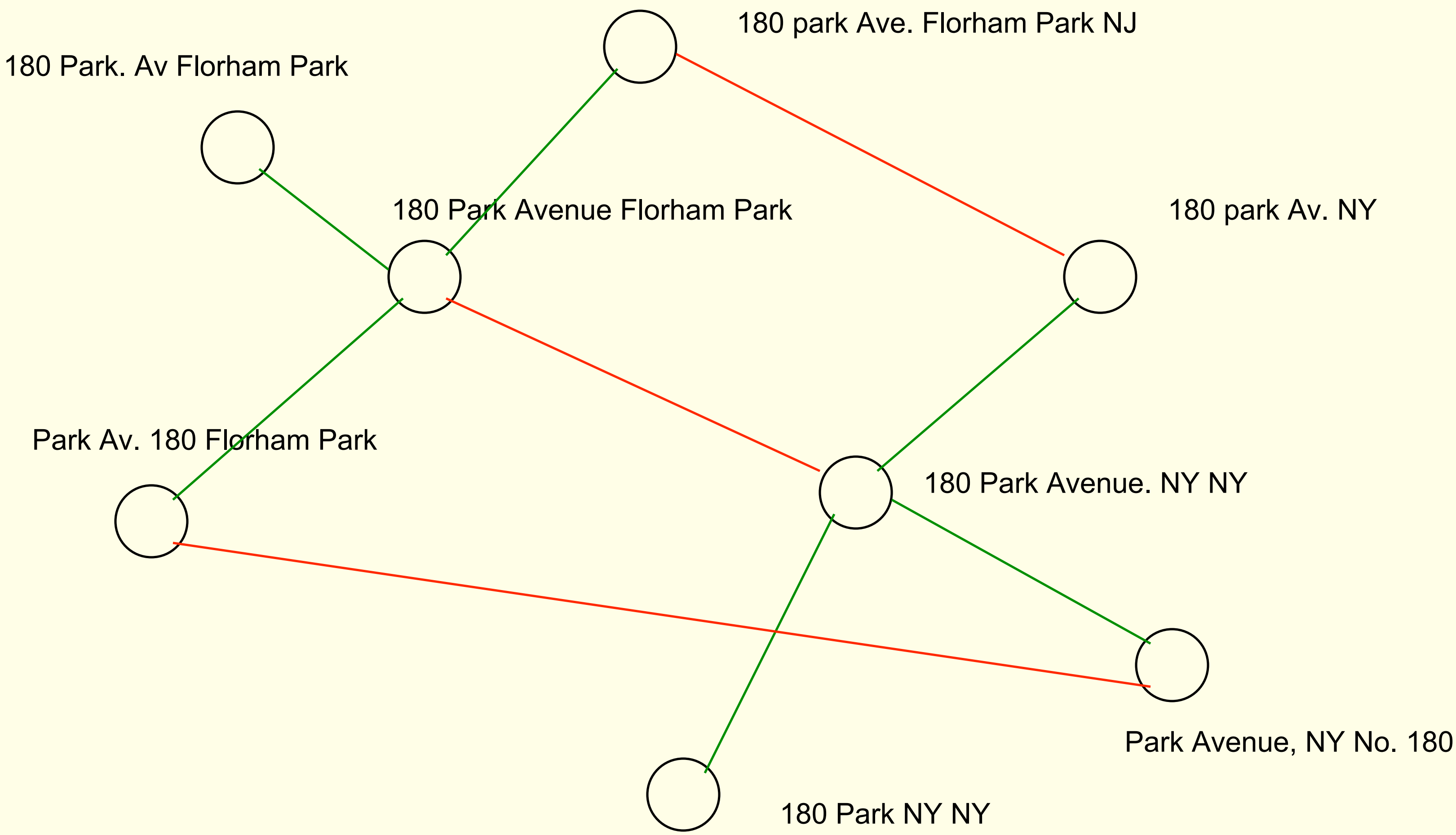


Application: Merging Lists

- Application: merge address lists (customer lists, company lists) to avoid redundancy
- Current status: “standardize”, different values treated as distinct for analysis
  - Lot of heterogeneity
  - Need approximate joins
- Relevant technologies
  - Approximate joins
  - Clustering/partitioning

ADDR
11810 WILLS RD ALPHARETTA GA 30076
11810 WILLS ROAD ALPHARETTA GA30004
FLR 110 RM 155 11810 WILLS RD ALPHARETTA GA 300042055
FLR 110 RM MAIN 11810 WILLIS RD ALPHARETTA GA 300042055
FLR 110 RM MAIN 11810 WILLS RD ALPHARETTA GA 30076
FLR BLDG 110 RM 155 11810 WILLS RD ALPHARETTA GA 300042055
FLR BLDG 110 RM MAIN 111810 WILLS RD ALPHARETTA GA 30076
FLR BLDG 110 RM MAIN 11810 WILLS RD ALPHARETTA GA 300042055
FLR MAIN RM 110 11810 WILLS RD ALPHARETTA GA 300042055
FLR MAIN RM BLGD 110 11810 WILLS RD ALPHARETTA GA 300042055
FLR NA RM NA 11810 WILLS RD ALPHARETTA GA 300042055
BLDG 110 FLR 1 RM 110 11810 WILLIS RD ALPHARETTA GA 30004205
BLDG 110 FLR 1 RM 110 11810 WILLS RD ALPHARETTA GA 300042055
BLDG 110 FLR 1 RM 110 11810 WILLS RD ALPHARETTA GA 300042081
BLDG 110 FLR 1 RM 110 11810 WILLS RD ALPHARETTA GA 30076
BLDG 110 FLR 1 RM RING 11810 WILLS RD ALPHARETTA GA 30004208
FLR 1 RM 1 11810 WILLS RD ALPHARETTA GA 300042055
FLR 1 RM 110 11801 WILLIS RD ALPHARETTA GA 30076
FLR 1 RM 110 11810 WILLIS RD ALPHARETTA GA 300042055
FLR 1 RM 110 11810 WILLIS RD ALPHARETTA GA 30076
FLR 1 RM 110 11810 WILLS RD ALPHARETTA GA 300042055
FLR 1 RM 110 11810 WILLS RD ALPHARETTA GA 300042081
FLR 1 RM 110 11810 WILLS RD ALPHARETTA GA 30076
FLR 1 RM 110 11810 WILLS RD ATLANTA GA 30076
FLR 1 RM 110 11810 WILLS ROAD ALPHARETTA GA 30076
FLR 1 RM BLDG 110 11810 WILLS RD ALPHARETTA GA 300042055
FLR 1 RM BLDG 11810 WILLS RD ALPHARETTA GA 300042055
FLR 1 RM COMPUTER 11810 WILLS RD ALPHARETTA GA 300042055

Application: Merging Lists



Application: Homeland Security

- Application: correlate airline passenger data with homeland security data for no-fly lists
- Current status: “match” on name, deny boarding
  - Use more match attributes
  - Obtain more information
- Relevant technologies
  - Schema mappings
  - Approximate joins

TRAVEL

'No-fly list' keeps infants off planes

Tuesday, August 16, 2005; Posted: 4:49 a.m. EDT (08:49 GMT)

WASHINGTON (AP) -- Infants have been stopped from boarding planes at airports throughout the United States because their names are the same as or similar to those of possible terrorists on the government's "no-fly list."

It sounds like a joke, but it's not funny to parents who miss flights while scrambling to have babies' passports and other documents faxed.

Ingrid Sanden's 1-year-old daughter was stopped in Phoenix, Arizona, before boarding a flight home to Washington at Thanksgiving.

"I completely understand the war on

(AP PHOTO)

Ingrid Sanden holds her 1-year-old daughter, who was stopped before boarding a flight last Thanksgiving.

Record Linkage: Tip of the Iceberg

- An approximate join of R_1 and R_2 is
  - A subset of the cartesian product of R_1 and R_2
  - “Matching” specified attributes of R_1 and R_2
  - Labeled with a similarity score > t > 0
- Clustering/partitioning of R: operates on the approximate join of R with itself.

The Fellegi-Sunter Model [FS69]

- Formalized the approach of Newcombe et al. [NKAJ59]
- Given two sets of records (relations) A and B perform an approximate join
  - $A \times B = \{(a,b) \mid a \in A,\ b \in B\} = M \cup U$
  - $M = \{(a,b) \mid a=b,\ a \in A,\ b \in B\}$; matched
  - $U = \{(a,b) \mid a <> b,\ a \in A,\ b \in B\}$; unmatched
- $\gamma(a,b) = (\gamma^i(a,b))$ i=1..K comparison vector
  - Contains comparison features e.g., same last names, same SSN, etc.
- Γ: range of $\gamma(a,b)$ the comparison space.

The Fellegi-Sunter Model

- Seeking to characterize (a,b) as
  - A_1 : match ; A_2 : uncertain ; A_3 : non-match
- Function (linkage rule) from Γ to $\{A_1\ A_2\ A_3\}$
- Distribution D over A x B
  - $m\ (\gamma) = P(\gamma(a,b)\ |\ (a,b) \in M\}$
  - $u\ (\gamma) = P(\gamma(a,b)\ |\ (a,b) \in U\}$

Fellegi-Sunter Result

- Sort vectors γ by m (γ)/u (γ) non increasing order; choose n < n'
  - $\mu = \sum_{i=1}^{n} u(\gamma_\iota)$ $\lambda = \sum_{i=n'}^{N} m(\gamma_\iota)$
  - Linkage rule with respect to minimizing $P(A_2)$, with $P(A_1|U) = \mu$ and $P(A_3|M) = \lambda$ is
    - $\gamma_1, \ldots\ldots, \gamma_n, \gamma_{n+1}, \ldots\ldots, \gamma_{n'-1}, \gamma_{n'}, \ldots\ldots, \gamma_N$
    - A_1 A_2 A_3
    - Intuition
      - Swap i-th vector declared as A_1 with j-th vector in A_2
      - If $u(\gamma_i) = u(\gamma_j)$ then $m(\gamma_j) < m(\gamma_I)$
      - After the swap, $P(A_2)$ is increased

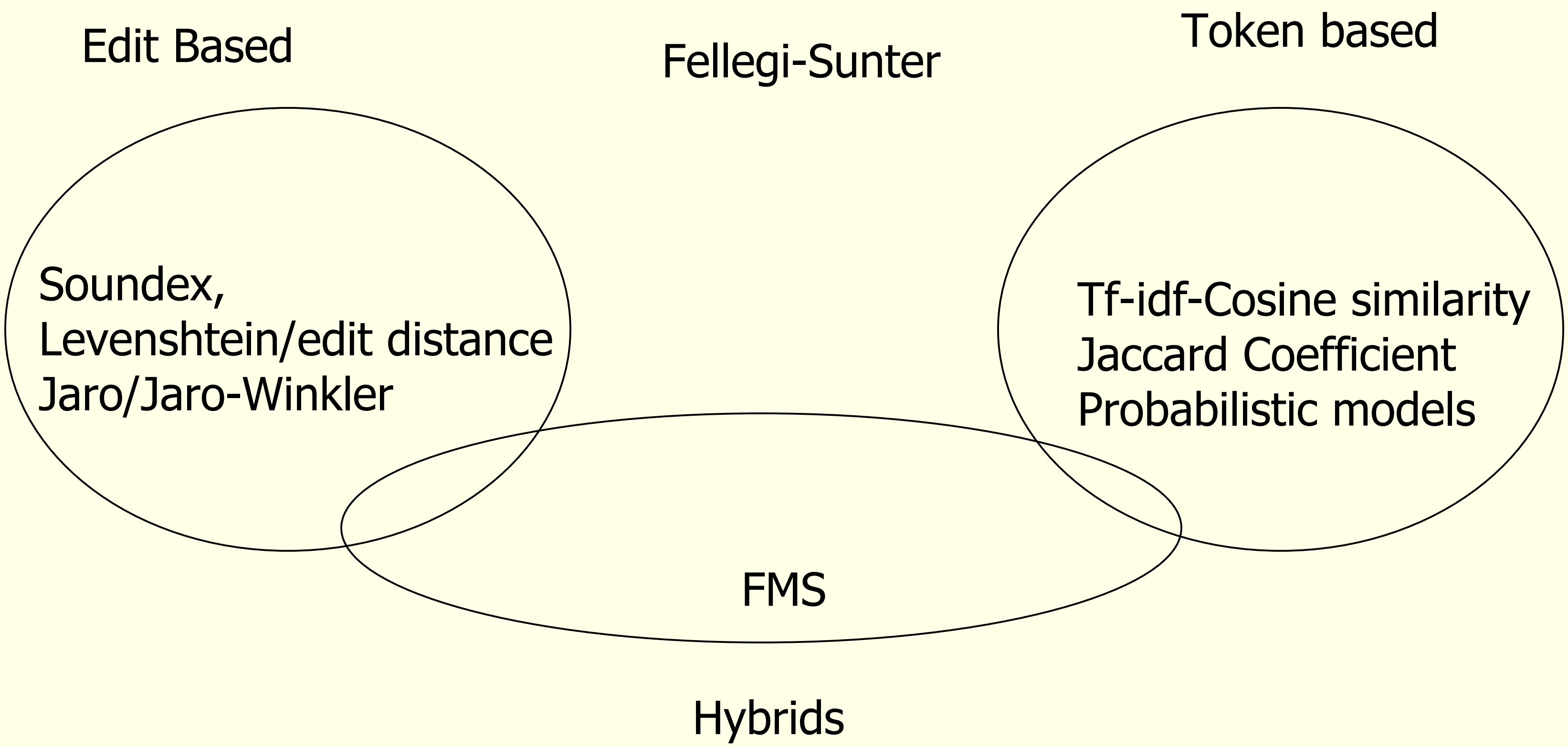
Fellegi-Sunter Issues:

- Tuning:
  - Estimates for m (γ), u (γ) ?
    - Training data: active learning for M, U labels
    - Semi or un-supervised clustering: identify M U clusters
  - Setting μ , λ?
  - Defining the comparison space Γ?
    - Distance metrics between records/fields
- Efficiency/Scalability
  - Is there a way to avoid quadratic behavior (computing all |A|x|B| pairs)?

Outline

- Part I: Motivation, similarity measures (90 min)
  - Data quality, applications
  - Linkage methodology, core measures
  - Learning core measures
  - Linkage based measures
- Part II: Efficient algorithms for approximate join (60 min)
- Part III: Clustering/partitioning algorithms (30 min)

Classification of the measures



Attribute Standardization

- Several attribute fields in relations have loose or anticipated structure:
  - Addresses, names
  - Bibliographic entries (mainly for web data)
- Preprocessing to standardize such fields
  - Enforce common abbreviations, titles
  - Extract structure from addresses
- Part of ETL tools, commonly using field segmentation and dictionaries
- Recently machine learning approaches
  - HMM encode universe of states [CCZ02]

Field Similarity

- Application notion of ‘field’
  - Relational attribute, set of attributes, entire tuples.
- Basic problem: given two field values quantify their ‘similarity’ (wlog) in [0..1].
- If numeric fields, use numeric methods.
- Problem challenging for strings.

Soundex Encoding

- A phonetic algorithm that indexes names by their sounds when pronounced in english.
- Consists of the first letter of the name followed by three numbers. Numbers encode similar sounding consonants.
    - Remove all W, H
    - B, F, P, V encoded as 1, C,G,J,K,Q,S,X,Z as 2
    - D,T as 3, L as 4, M,N as 5, R as 6, Remove vowels
    - Concatenate first letter of string with first 3 numerals
- Ex: great and grate become 6EA3 and 6A3E and then G63
- More recent, metaphone, double metaphone etc.

Edit Distance [G98]

- Character Operations: I (insert), D (delete), R (Replace).
- Unit costs.
- Given two strings, s,t, edit(s,t):
  - Minimum cost sequence of operations to transform s to t.
  - Example: edit(Error,Eror) = 1, edit(great,grate) = 2
- Folklore dynamic programming algorithm to compute edit();
- Computation and decision problem: quadratic (on string length) in the worst case.

Edit Distance

- Several variants (weighted, block etc) -- problem can become NP-complete easily.
- Operation costs can be learned from the source (more later)
  - String alignment = sequence of edit operations emitted by a memory-less process [RY97].
- Observations
  - May be costly operation for large strings
  - Suitable for common typing mistakes
    - Comprehensive vs Comprenhensive
  - Problematic for specific domains
    - AT&T Corporation vs AT&T Corp
    - IBM Corporation vs AT&T Corporation

Edit Distance with affine gaps

- Differences between ‘duplicates’ often due to abbreviations or whole word insertions.
  - John Smith vs John Edward Smith vs John E. Smith
  - IBM Corp. vs IBM Corporation
- Allow sequences of mis-matched characters (gaps) in the alignment of two strings.
- Penalty: using the affine cost model
  - $Cost(g) = s+e \cdot l$
  - s: cost of opening a gap
  - e: cost of extending the gap
  - l: length of a gap
  - Commonly e lower than s
- Similar dynamic programming algorithm

Jaro Rule [J89]

- Given strings $s = a_1,\ldots,a_k$ and $t = b_1,\ldots,b_L$ a_i in s is common to a character in t if there is a b_j in t such that $a_i = b_j$ $i\text{-}H \leq j \leq i\text{+}H$ where
  - $H = \min(|s|,|t|)/2$
- Let $s' = a_1',\ldots,a_{k'}'$ and $t' = b_1',\ldots,b_{L'}'$ characters in s (t) common with t (s)
- A transposition for s',t' is a position i such that $a_i' <> b_i'$.
- Let $T_{s',t'}$ be half the number of transpositions in s' and t'.

Jaro Rule

- Jaro(s,t) = $\frac{1}{3}\left(\frac{|s'|}{|s|}+\frac{|t'|}{|t|}+\frac{|s'|-T_{s',t'}}{|s'|}\right)$
- Example:
  - Martha vs Marhta
    - H = 3, s' = Martha, t' = Marhta, $T_{s',t'}$ = 1
    - Jaro(Martha,Marhta) = 0.9722
  - Jonathan vs Janathon
    - H = 4, s' = jnathn, t' = jnathn, $T_{s',t'}$ = 0
    - Jaro(Jonathan,Janathon) = 0.5

Jaro-Winkler Rule [W99]

- Uses the length P of the longest common prefix of s and t; P’ = max(P,4)
- Jaro-Winkler(s,t) = $Jaro(s,t) + \frac{P'}{10}(1 - Jaro(s,t))$
- Example:
  - JW(Martha,Marhta) = 0.9833
  - JW(Jonathan,Janathon) = 0.7
- Observations:
  - Both intended for small length strings (first,last names)

Term (token) based

- Varying semantics of ‘term’
  - Words in a field
    - ‘AT&T Corporation’ -> ‘AT&T’ , ‘Corporation’
  - Q-grams (sequence of q-characters in a field)
    - {‘AT&’,’T&T’,’&T ‘, ‘T C’,’ Co’,’orp’,’rpo’,’por’,’ora’,’rat’,’ati’,’tio’,’ion’} 3-grams
- Assess similarity by manipulating sets of terms.

Overlap metrics

- Given two sets of terms S, T
  - Jaccard coef.: Jaccard(S,T) = $|S \cap T| / |S \cup T|$
  - Variants
    - If scores (weights) available for each term (element in the set) compute Jaccard() only for terms with weight above a specific threshold.
- What constitutes a good choice of a term score?

TF/IDF [S83]

- Term frequency (tf) inverse document frequency (idf).
- Widely used in traditional IR approaches.
- The tf/idf value of a ‘term’ in a document:
  - log (tf+1) * log idf where
    - tf : # of times ‘term’ appears in a document d
    - idf : number of documents / number of documents containing ‘term’
  - Intuitively: rare ‘terms’ are more important

TF/IDF

- Varying semantics of ‘term’
  - Words in a field
    - ‘AT&T Corporation’ -> ‘AT&T’ , ‘Corporation’
  - Qgrams (sequence of q-characters in a field)
    - {‘AT&’,’T&T’,’&T ‘, ‘T C’,’ Co’,’orp’,’rpo’,’por’,’ora’,’rat’,’ati’,’tio’,’ion’} 3-grams
- For each ‘term’ in a field compute its corresponding tfidf score using the field as a document and the set of field values as the document collection.

Probabilistic analog (from FS model)

- $P_s(j)$: probability for j in set S
- γ^j : event that values of corresponding fields are j in a random draw from sets A and B
- $m(\gamma^j) = P(\gamma^j|M) = P_{A\cap B}(j)$
- $u(\gamma^j) = P(\gamma^j|U) = P_A(j)P_B(j)$

- Assume $P_A(j) = P_B(j) = P_{A\cap B}(j)$
  - Provide more weight to agreement on rare terms and less weight to common terms
- IDF measure related to Fellegi-Sunter probabilistic notion:
  - $\mathrm{Log}(m(\gamma^{str})/u(\gamma^{str})) = \log(P_{A\cap B}(str)/P_A(str)P_B(str)) = \log(1/P_A(str)) = \mathrm{IDF}(str)$

Cosine similarity

- Each field value transformed via tfidf weighting to a (sparse) vector of high dimensionality d.
- Let a,b two field values and S_a, S_b the set of terms for each. For w in S_a (S_b), denote $W(w,S_a)$ ($W(w,S_b)$) its tfidf score.
- For two such values:
  - Cosine(a,b) = $\sum_{z \in Sa \cap Sb} W(z,Sa)W(z,Sb)$

Cosine similarity

- Suitable to assess closeness of
  - ‘AT&T Corporation’, ‘AT&T Corp’ or ‘AT&T Inc’
    - Low weights for ‘Corporation’,’Corp’,’Inc’
    - Higher weight for ‘AT&T’
    - Overall Cosine(‘AT&T Corp’,’AT&T Inc’) should be high
  - Via q-grams may capture small typing mistakes
    - ‘Jaccard’ vs ‘Jacard’ -> {‘Jac’,’acc’,’cca’,’car’,’ard’} vs {‘Jac’,’aca’,’car’,’ard’}
    - Common terms ‘Jac’, ‘car’, ‘ard’ would be enough to result in high value of Cosine(‘Jaccard’,’Jacard’).

Hybrids [CRF03]

- Let S = $\{a_1,\ldots,a_K\}$, T = $\{b_1,\ldots b_L\}$ sets of terms:
- Sim(S,T) = $$\frac{1}{K}\sum_{i=1}^{K}\max_{j=1}^{L} sim'(a_i, b_j)$$
- Sim'() some other similarity function
- $C(t,S,T) = \{w \in S \text{ s.t } \exists\, v \in T, sim'(w,v) > t\}$
- $D(w,T) = \max_{v \in T} sim'(w,v),\ w \in C(t,S,T)$
  - sTFIDF = $$\sum_{w \in C(t,S,T)} W(w,S) * W(w,T) * D(w,T)$$

Other choices for term score?

- Several schemes proposed in IR
  - Okapi weighting
    - Model within document term frequencies as a mixture of two poisson distributions: one for relevant and one for irrelevant documents
  - Language models
    - Given $Q=t_1,...t_n$ estimate $p(Q|M_d)$
    - MLE estimate for term t : $p(t|M_d) = tf(t,d)/dl_d$
      - dl_d:total number of tokens in d
    - Estimate $p_{avg}(t)$
    - Weight it by a risk factor (modeled by a geometric distribution)
  - HMM

Fuzzy Match Similarity [CGGM03]

- Sets of terms S, T
- Main idea: cost of transforming S to T, tc(S,T).
- Transformation operations like edit distance.
  - Replacement cost: edit(s,t)*W(s,S)
  - Insertion cost: c_{ins} W(s,S) (c_{ins} between 0,1)
  - Deletion cost: W(s,S)
- Computed by DP like edit()
- Generalized for multiple sets of terms

Fuzzy Match Similarity

- Example
  - ‘Beoing Corporation’,’Boeing Company’
  - S = {‘Beoing’,’Corporation}, T = {‘Boeing’,Company’}
  - tc(S,T) = 0.97 (unit weights for terms)
    - sum of
      - edit(‘Beoing’,’Boeing’) = 2/6 (normalized)
      - edit(‘Corporation’,Company’) = 7/11

Fuzzy Match Similarity

- W(S) = sum of W(s,S) for all s $\in$S
- fms = 1-min((tc(S,T)/W(S),1)
- Approximating fms:
  - For s $\in$ S let QG(s) set of qgrams of s
  - d= (1-1/q)
  - fms^{apx} = $\frac{1}{W(S)}\sum_{s\in S} W(s,S) * \max_{t\in T}(\frac{2}{q} sim_{mh}(QG(s),QG(t)) + d)$
  - For suitable δ, ε and size of min hash signature
    - $E(fms^{apx}(S,T)) \geq fms(S,T)$
    - $P(fms^{apx}(S,T) \leq (1-\delta)fms(S,T)) \leq \varepsilon$

Multi-attribute similarity measures

- Weighted sum of per attribute similarity
- Application of voting theory
- Rules (more of this later)

Voting theory application [GKMS04]

- Relations R with n attributes.
- In principle can apply a different similarity function for each pair of attributes into consideration.
- N orders of the relation tuples, ranked by a similarity score to a query.

Voting Theory

Tuple id	custname	address	location
T1	John smith	800 Mountain Av springfield	5,5
T2	Josh Smith	100 Mount Av Springfield	8,8
T3	Nicolas Smith	800 spring Av Union	11,11
T4	Joseph Smith	555 Mt. Road Springfield	9,9
T5	Jack Smith	100 Springhill lake Park	6,6

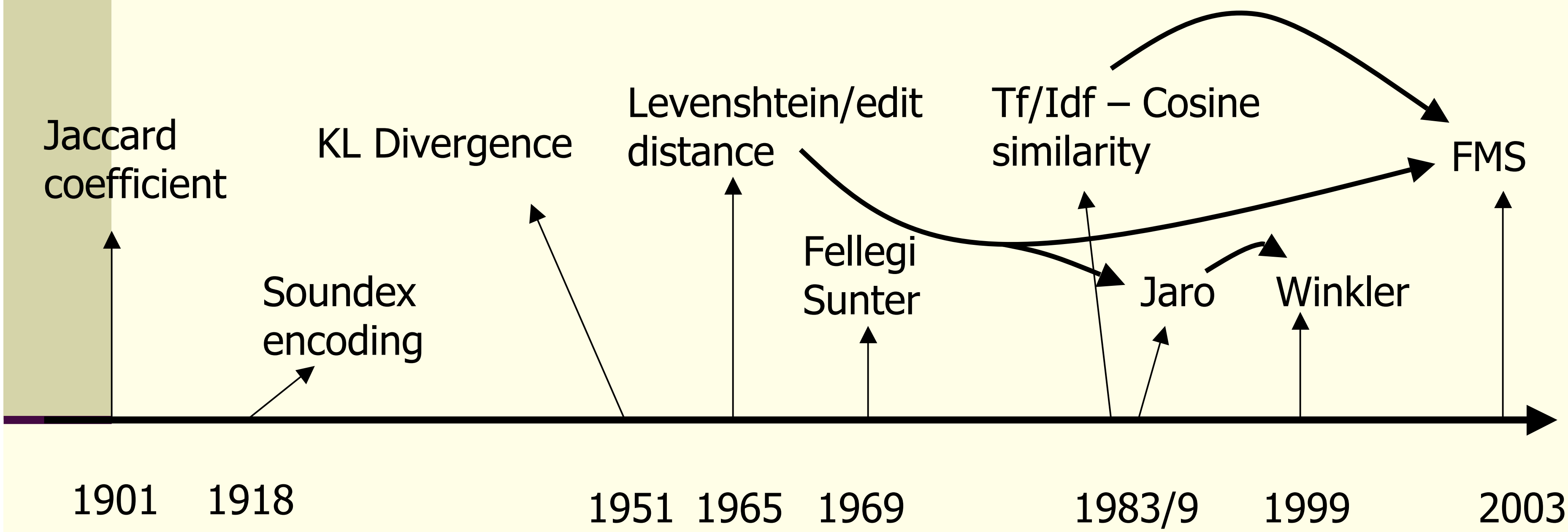
Query: John smith 100 Mount Rd. Springfield 5.1,5.1

custname	address	location
T1 (1.0)	T2 (0.95)	T1 (0.95)
T2 (0.8)	T1 (0.8)	T5 (0.9)
T5 (0.7)	T4 (0.75)	T2 (0.7)
T4 (0.6)	T3 (0.3)	T4 (0.6)
T3 (0.4)	T5 (0.1)	T3 (0.3)

Voting theory application

- Merge rankings to obtain a consensus
- Foot-rule distance
  - Let S,T orderings of the same domain D
  - S(i) (T(i)) the order position of the i-th element of D in S (T)
  - F(S,T) = $\sum_{i \in D} |S(i) - T(i)|$
  - Generalized to distance between S and T1,..Tn
    - F(S,T1,..Tn) = $\sum_{j=1}^{n} F(S,Tj)$

Historical timeline



Outline

- Part I: Motivation, similarity measures (90 min)
  - Data quality, applications
  - Linkage methodology, core measures
  - Learning core measures
  - Linkage based measures
- Part II: Efficient algorithms for approximate join (60 min)
- Part III: Clustering algorithms (30 min)

Learning similarity functions

- Per attribute
  - Term based (vector space)
  - Edit based
    - Learning constants in character-level distance measures like levenshtein distances
    - Useful for short strings with systematic errors (e.g., OCRs) or domain specific error (e.g.,st., street)
- Multi-attribute records
  - Useful when relative importance of match along different attributes highly domain dependent
  - Example: comparison shopping website
    - Match on title more indicative in books than on electronics
    - Difference in price less indicative in books than electronics

Learning Distance Metrics [ST03]

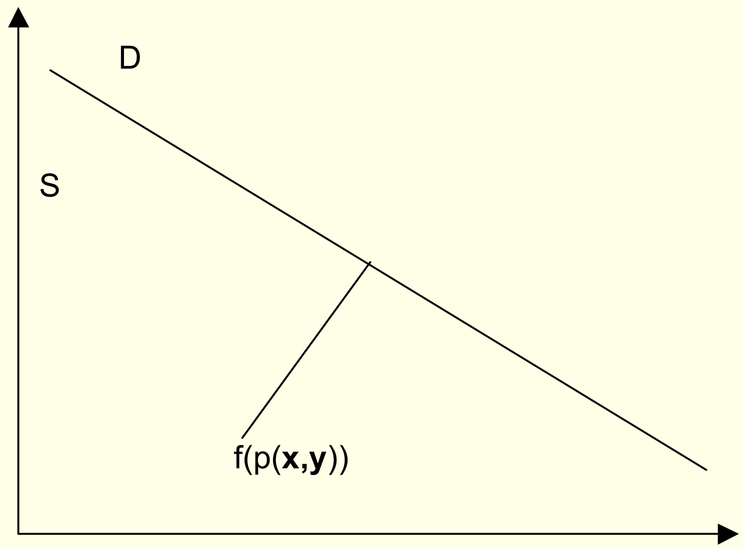
- Learning a distance metric from relative comparisons:
  - A is closer to B than A is to C, etc
- $d_{(A,W)}$ (**x-y**) = $\sqrt{(x-y)^T AWA^T (x-y)}$
  - A can be a real matrix: corresponds to a linear transform of the input
  - W a diagonal matrix with non-negative entries (guarantees d is a distance metric)
  - Learn entries of W such that to minimize training error
- Zero training error:
  - $\forall$ (i,j,k) ε Training set: $d_{(A,W)}(x_i,x_k)-d_{(A,W)}(x_i,x_k) > 0$
- Select A,W such that d remains as close to an un-weighted euclidean metric as possible.

Learnable Vector Space Similarity

- Generic vector space similarity via tfidf
  - Tokens ‘11th’ and ‘square’ in a list of addresses might have same IDF values
  - Addresses on same street more relevant than addresses on a square..
  - Can we make the distinction?
- Vectors **x**,**y,** Sim(**x,y**) = $\sum_{i=1}^{d} \frac{x_i y_i}{\| x \| \| y \|}$
- Training data:
  - S = {(**x,y**): x similar y}, D = {(**x,y**) x different y}

Learnable Vector Space Similarity

$$\begin{array}{l} \text{7} \\ \text{walmer} \\ \text{road} \\ \text{toronto} \\ \text{ontario} \\ \text{on} \end{array} \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \end{bmatrix} \begin{bmatrix} y1 \\ y2 \\ y3 \\ y4 \\ y5 \\ y6 \end{bmatrix} \longrightarrow \underbrace{\begin{bmatrix} x1y1 \\ x2y2 \\ x3y3 \\ x4y4 \\ x5y5 \\ x6y6 \end{bmatrix}}_{P(\mathbf{x},\mathbf{y})}$$



7 walmer road toronto ontario
7 walmer road toronto on

$$\mathrm{sim}(x,y) = \frac{f(p(x,y) - f_{min}}{f_{max} - f_{min}}$$

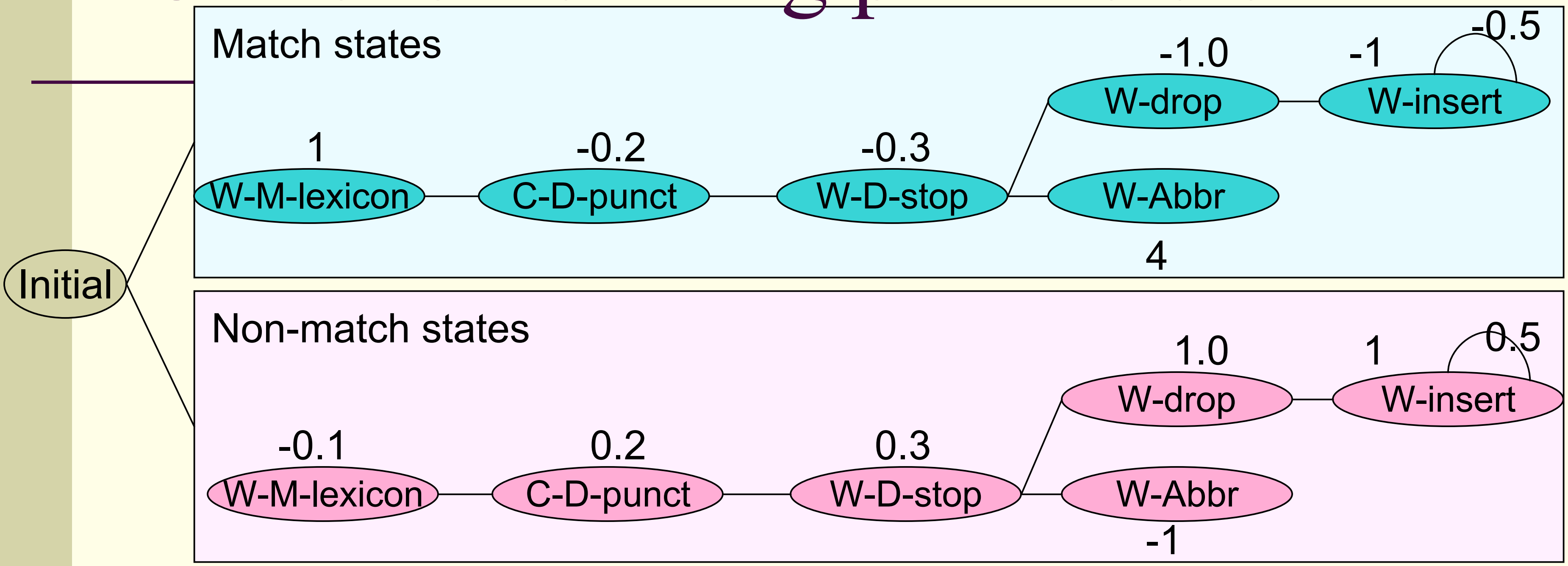
Learning edit distance parameters

- Free to set relative weights of operations
- May learn weights from input [RY97] using an EM approach.
  - Input: similar pairs
  - Parameters: probability of edit operations
  - E: highest probability edit sequence
  - M: re-estimate probabilities using expectations of the E step
    - Pros: FSM representation (generative model)
    - Cons: fails to incorporate negative examples
- [BM03] extend to learn weights of edit operations with affine gaps
- [MBP05] use CRF approach (incorporates positive and negative input)

Learning edit parameters using CRFs

- Sequence of edit operations
  - Standard character-level: Insert, Delete, Substitute
    - Costs depends on type: alphabet, number, punctuation
  - Word-level: Insert, Delete, Match, Abbreviation
    - Varying costs: stop words (Eg: The), lexicons (Eg: Corporation, Road)
- Given: examples of duplicate and non-duplicate strings
- Learner: Conditional Random Field
  - Allows for flexible overlapping feature sets
    - Ends with a dot and appears in a dictionary
  - Discriminative training ~ higher accuracy than earlier generative models

CRFs for learning parameters



Proc. of SIGMOD
Proc Sp. Int. Gr Management of Data

- State and transition parameters for match and non-match states
- Multiple paths through states summed over for each pair
- EM-like algorithm for training.

Results

Citations

Distance Metric	Restaurant name	Restaurant address	Reasoning	Face
Edit Distance	0.290	0.686	0.927	0.952
Learned Edit Distance	0.354	0.712	0.938	**0.966**
Vector-space	0.365	0.380	0.897	0.922
Learned Vector-space	0.433	0.532	0.924	0.875
CRF Edit Distance	**0.448**	**0.783**	**0.964**	0.918

Earlier generative approach (BM03)

Word-level only, no order

Initialized with manual weights

(McCallum, Bellare, Pereira EMNLP 2005)

- Edit-distance is better than word-level measures
- CRFs trained with both duplicates and non-duplicates better than generative approaches using only duplicates
- Learning domain-specific edit distances could lead to higher accuracy than manually tuned weights

Learning similarity functions

- Per attribute
  - Term based (vector space)
  - Edit based
    - Learning constants in character-level distance measures like levenshtein distances
    - Useful for short strings with systematic errors (e.g., OCRs) or domain specific error (e.g.,st., street)
- Multi-attribute records
  - Useful when relative importance of match along different attributes highly domain dependent
  - Example: comparison shopping website
    - Match on title more indicative in books than on electronics
    - Difference in price less indicative in books than electronics

Multi Attribute Similarity

Record 1, Record 2	D
Record 1, Record 3	N
Record 4, Record 5	D

f_1	f_2 ... f_n			
1.0	0.4	...	0.2	1
0.0	0.1	...	0.3	0
0.3	0.4	...	0.4	1

Mapped examples

Unlabeled list

Record 6
Record 7
Record 8
Record 9
Record 10
Record 11

0.0	0.1	...	0.3	?
1.0	0.4	...	0.2	?
0.6	0.2	...	0.5	?
0.7	0.1	...	0.6	?
0.3	0.4	...	0.4	?
0.0	0.1	...	0.1	?
0.3	0.8	...	0.1	?
0.6	0.1	...	0.5	?

Nor

N

All-Ngrams*0.4 + AuthorTitleNgram*0.2 – 0.3YearDifference + 1.0*AuthorEditDis + 0.2*PageMatch – 3 > 0

Learners:

- Support Vector Machines (SVM)
- Logistic regression,
- Linear regression,
- Perceptron

0.6	0.2	...	0.5	0
0.7	0.1	...	0.6	0
0.3	0.4	...	0.4	1
0.0	0.1	...	0.1	0
0.3	0.8	...	0.1	1
0.6	0.1	...	0.5	1

Learning approach

- Learners used:
  - SVMs: high accuracy with limited data,
  - Decision trees:interpretable, efficient to apply
  - Perceptrons: efficient incremental training (Bilenko et al 2005, Comparison shopping)
- Results:
  - Learnt combination methods better than both
    - Averaging of attribute-level similarities
    - String based methods like edit distance (Bilenko et al 2003)
- Downside
  - Creating meaningful training data a huge effort

Training data for learning approach

- Heavy manual search in preparing training data
  - Hard to spot challenging/covering duplicates in large lists
  - Even harder to find close non-duplicates that will capture the nuances
  - Need to seek out rare forms of errors in data
- A solution from machine learning → Active learning
  - Given
    - Lots of unlabeled data → pairs of records
    - Limited labeled data
  - Find examples most informative for classification
    - Highest uncertainty of classification from current data

The active learning approach

Similarity functions

Record 1 D
Record 2

Record 3 N
Record 4

f_1	f_2	...	f_n	
1.0	0.4	...	0.2	1
0.0	0.1	...	0.3	0

Committee of classifiers

0.7	0.1	...	0.6	1
0.3	0.4	...	0.4	0

Unlabeled list

Record 6
Record 7
Record 8
Record 9
Record 10
Record 11

0.0	0.1	...	0.3	?
1.0	0.4	...	0.2	?
0.6	0.2	...	0.5	?
0.7	0.1	...	0.6	?
0.3	0.4	...	0.4	?
0.0	0.1	...	0.1	?
0.3	0.8	...	0.1	?
0.6	0.1	...	0.5	?

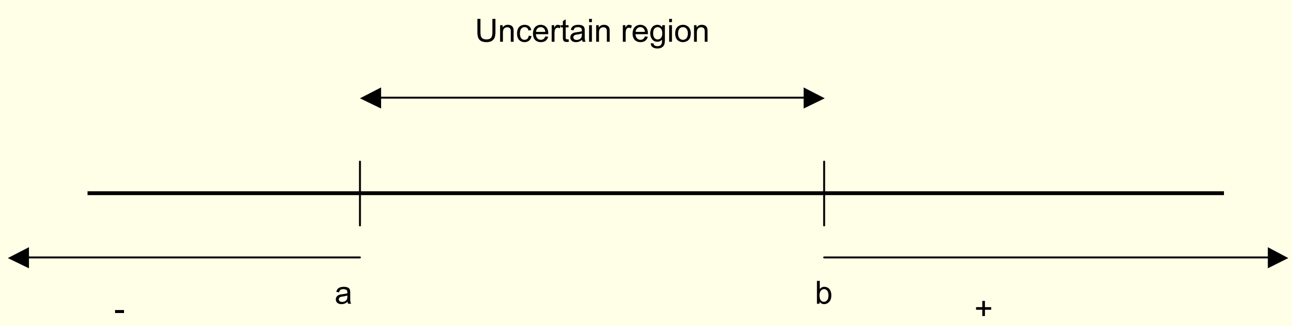
Active Learner

0.7	0.1	...	0.6	?
0.3	0.4	...	0.4	?

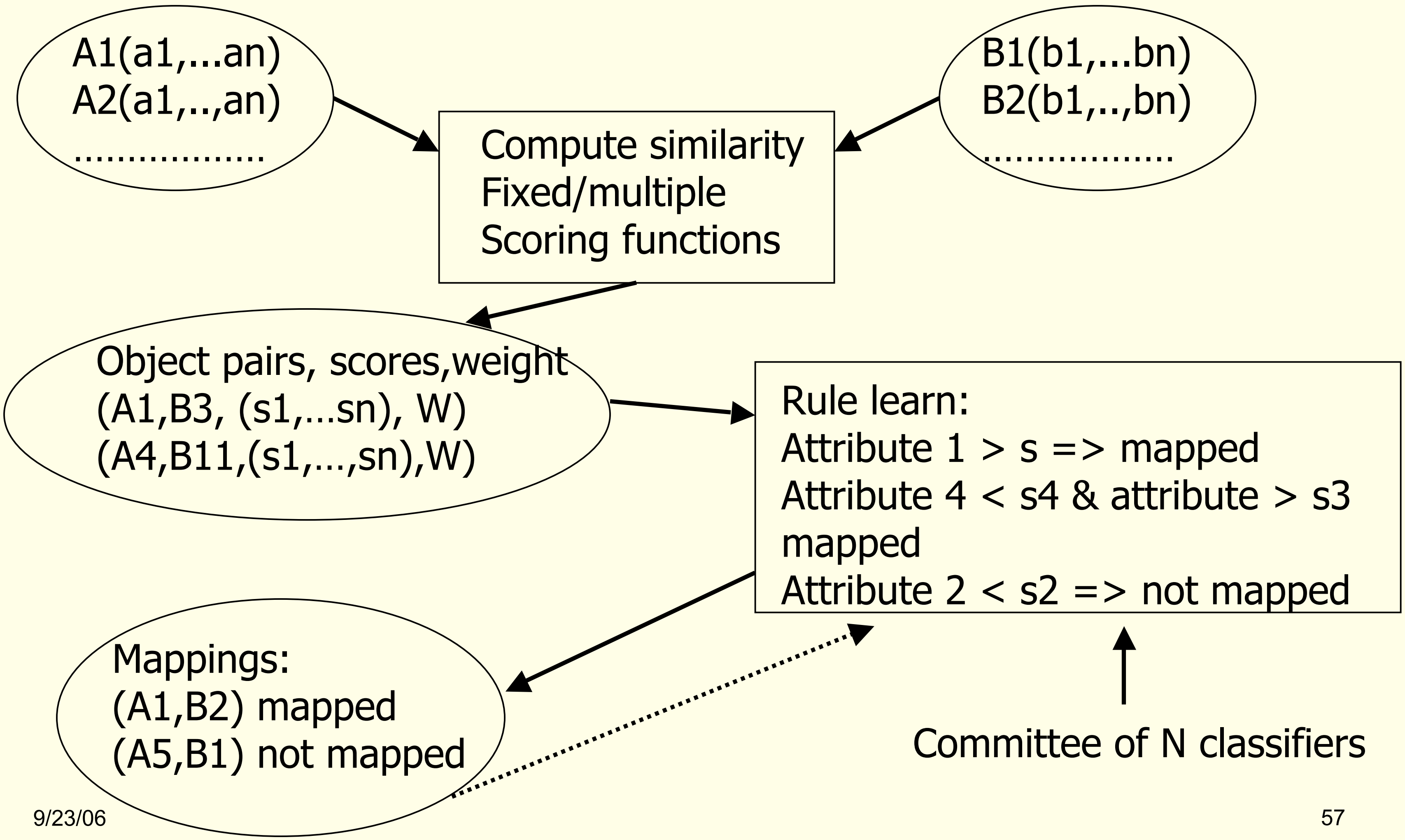
Picks highest disagreement records

Active Learning [SB02]

- Learn a 'similarity' function (classifier) from labeled data
- Small set of labeled data (pos,neg) and unlabeled data
- Seek instances that when labeled will strengthen the classification process
- Initial classifier sure about prediction on some unlabeled instances and unsure about others (confusion region)
  - Seek predictors on uncertain instances



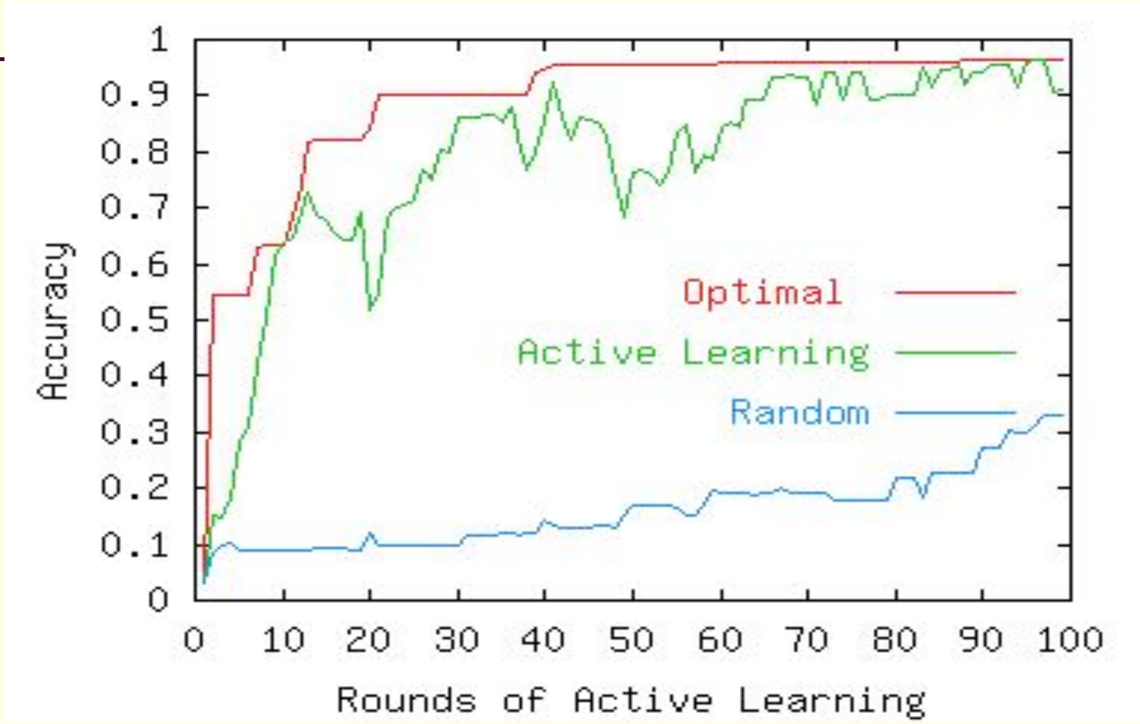
Active Learning Approaches [TKM01]



Active learning algorithm

- Train k classifiers C1, C2,.. Ck on training data through
  - Data resampling,
  - Classifier perturbation
- For each unlabeled instance x
  - Find prediction y1,.., yk from the k classifiers
  - Compute uncertainty U(x) as entropy of above y-s
- Pick instance with highest uncertainty

Benefits of active learning



- Active learning much better than random
  - With only 100 active instances
    - 97% accuracy, Random only 30%
- Committee-based selection close to optimal

Learning: beyond paired 0/1 classification

- Exploiting monotonicity between attribute similarity and class label to learn better
  - A Hierarchical Graphical Model for Record Linkage (Ravikumar, Cohen, UAI 2004)
- Exploiting transitivity to learn on groups
  - T. Finley and T. Joachims, *Supervised Clustering with Support Vector Machines*, Proceedings of the International Conference on Machine Learning (ICML), 2005.

Outline

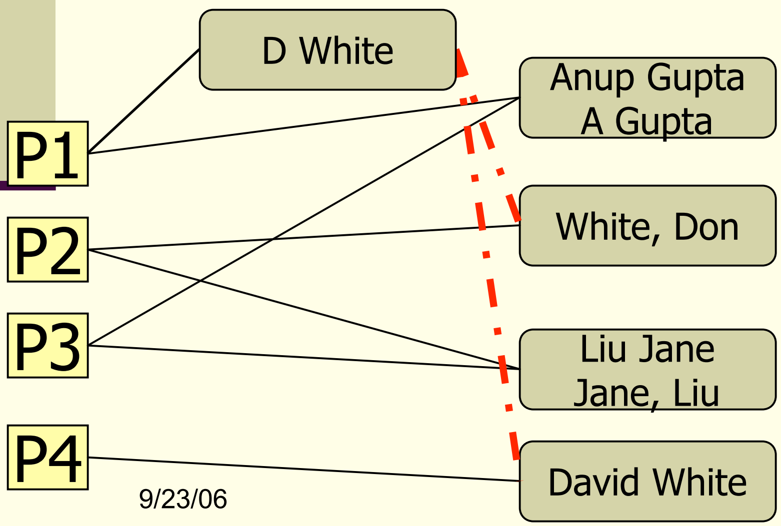
- Part I: Motivation, similarity measures (90 min)
  - Data quality, applications
  - Linkage methodology, core measures
  - Learning core measures
  - Linkage based measures
- Part II: Efficient algorithms for approximate join (60 min)
- Part III: Clustering algorithms (30 min)

Similarity based on linkage pattern

P1	D White, A Gupta
P2	Liu, Jane & White, Don
P3	Anup Gupta and Liu Jane
P4	David White

Relate D White and Don White through the third paper

Path in graph makes D White more similar to Don White than David White



Lots of work on node similarities in graph

- sim-rank, conductance models, etc

  RelDC (Kalashnikov et al 2006)

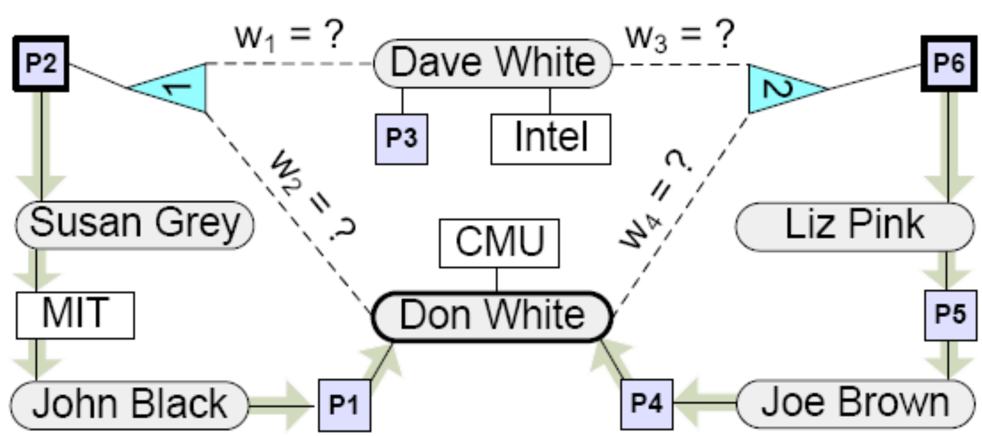
RelDC: Example with multiple entity types

$\langle A_1$, 'Dave White', 'Intel'$\rangle$
$\langle A_2$, 'Don White', 'CMU'$\rangle$
$\langle A_3$, 'Susan Grey', 'MIT'$\rangle$
$\langle A_4$, 'John Black', 'MIT'$\rangle$
$\langle A_5$, 'Joe Brown', unknown$\rangle$
$\langle A_6$, 'Liz Pink', unknown$\rangle$

$\langle P_1$, 'Databases . . . ', 'John Black', 'Don White'$\rangle$
$\langle P_2$, 'Multimedia . . . ', 'Sue Grey', **'D. White'**$\rangle$
$\langle P_3$, 'Title3 . . . ', 'Dave White'$\rangle$
$\langle P_4$, 'Title5 . . . ', 'Don White', 'Joe Brown'$\rangle$
$\langle P_5$, 'Title6 . . . ', 'Joe Brown', 'Liz Pink'$\rangle$
$\langle P_6$, 'Title7 . . . ', 'Liz Pink', **'D. White'**$\rangle$

Task: resolve author references in papers to author table

Path through co-affiliation



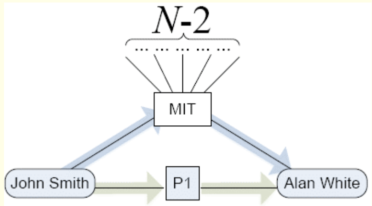
Path through co-authorship

(From: Kalashninov et al 2006)

Quantifying strength of connection

- Given a graph G with edges denoting node similarity or some form of relationship, find connection strength between any two nodes u, v
- Methods
  - Simple methods: shortest path length or flow
    - Fails for high-degree nodes
  - Diffusion kernels

$$\sum_k \lambda^k \sum_{\text{path}\,P\,k\text{-long}} \prod_{edge(i,j)\in P} s(i,j)$$



  - Electric circuit conductance model (Faloutsos et. al. 2004)
  - Walk-based model (WM)
    - Probabilistic
      - Treat edge weights as probability of transitioning out of node
      - Probability of reaching u from v via random walks
    - SimRank (Jeh&Widom 2002)
      - Expected distance to first meet of random walks from u and v
- RelDC extends (WM) to work for graphs with mutually exclusive choice nodes

RelDC

- Resolve whatever is possible via textual similarity alone
- Create relationship graph with unresolved references connected via choice nodes to options
  - Weights of options related to similarity
- Find connection strength between each unresolved reference to options, resolve to strongest of these
- Results
  - Authors: Author names, affiliation (HP Search)
  - Papers: Titles and Author names (Citeseer)
  - 13% ambiguous references (cannot be resolved via text alone)
  - 100% accuracy on 50 random tests

Outline

- Part I: Motivation, similarity measures (90 min)
- Part II: Efficient algorithms for approximate join (60 min)
  - Use traditional join methods
  - Extend traditional join methods
  - Commercial systems
- Part III: Clustering algorithms (30 min)

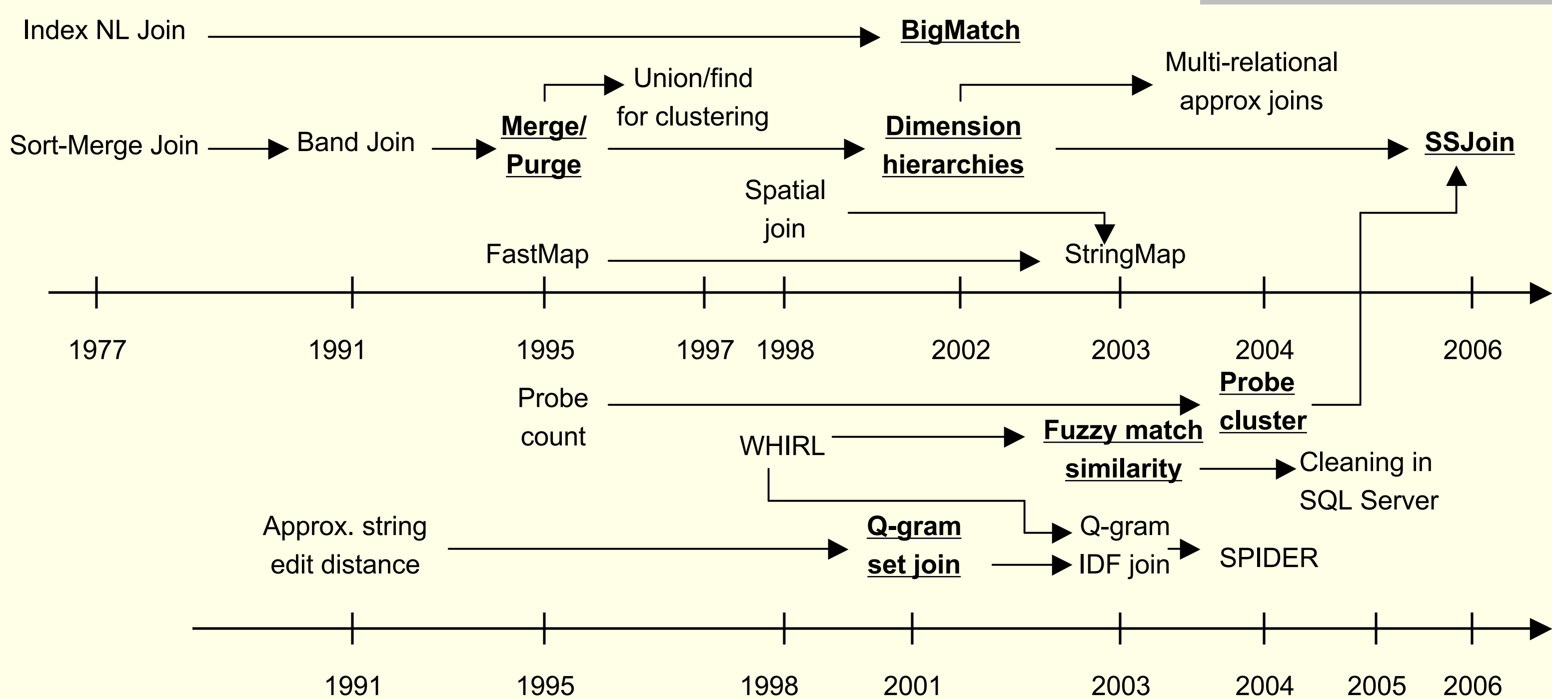
Approximate Joins: Baseline + Goal

- An **approximate join** of $R_1(A_1, \ldots, A_n)$ and $R_2(B_1, \ldots, B_m)$ is
  - A subset of the cartesian product of R_1 and R_2
  - "Matching" specified attributes $A_{i1}, \ldots, A_{ik}$ with $B_{i1}, \ldots, B_{ik}$
  - Labeled with a similarity score > t > 0

- Naïve method: for each record pair, compute similarity score
  - I/O and CPU intensive, not scalable to millions of records

- Goal: reduce $O(n^2)$ cost to $O(n*w)$, where w << n
  - Reduce number of pairs on which similarity is computed
  - Take advantage of efficient relational join methods

Historical Timelines



Sorted Neighborhood Method [HS95]

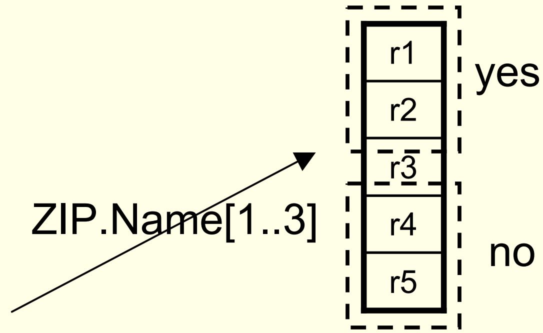
- Goal: bring matching records close to each other in linear list
- Background: duplicate elimination [BD83], band join [DNS91]
- Methodology: domain-specific, arbitrary similarity
  - Compute discriminating key per record, sort records
  - Slide fixed size window through sorted list, match in window
  - Use OPS5 rules (equational theory) to determine match
  - Multiple passes with small windows, based on distinct keys
- Lesson: multiple “cheap” passes faster than an “expensive” one

Sorted Neighborhood Method [HS95]

- Goal: bring matching records close to each other in linear list

- Example:

ID	Name	SS	DOB	ZIP
r1	Smith, John	123-45	1960/08/24	07932
r2	Smyth, Jon	123-45	1961/08/24	07932
r3	Smith, John	312-54	1995/07/25	98301
r4	Smith, J.	723-45	1960/08/24	98346
r5	Smith, J.	456-78	1975/12/11	98346

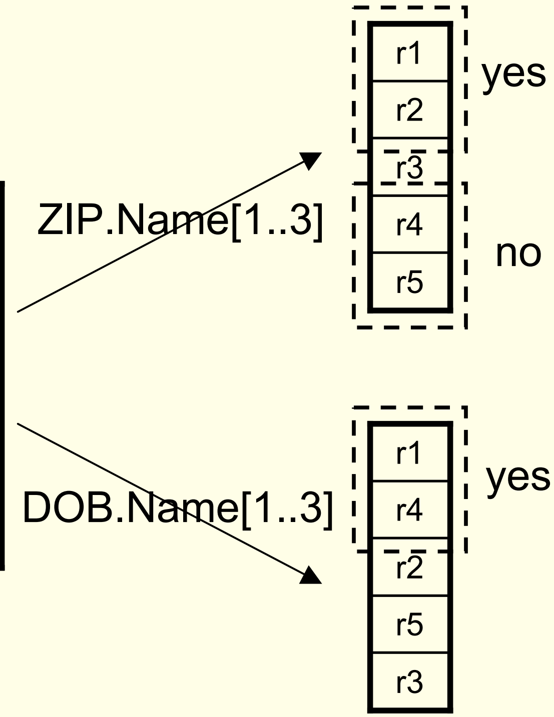


Sorted Neighborhood Method [HS95]

- Goal: bring matching records close to each other in linear list

- Example:

ID	Name	SS	DOB	ZIP
r1	Smith, John	123-45	1960/08/24	07932
r2	Smyth, Jon	123-45	1961/08/24	07932
r3	Smith, John	312-54	1995/07/25	98301
r4	Smith, J.	723-45	1960/08/24	98346
r5	Smith, J.	456-78	1975/12/11	98346



- Blocking is a special case

BigMatch [Y02]

- Goal: block/index matching records, based on multiple keys
- Background: indexed nested loop join [BE77]
- Methodology: domain-specific, Jaro-Winkler similarity
  - Store smaller table (100M) in main memory (4GB)
  - Create indexes for each set of grouping/blocking criteria
  - Scan larger table (4B), repeatedly probe smaller table
  - Avoids multiple matches of the same pair
- Lesson: traditional join technique can speed up approximate join

BigMatch [Y02]

- Goal: block/index matching records, based on multiple keys
- Example:

record from outer table

SS.Name[1..2]

yes

no

Smith, John	123-45	1960/08/24	98346

inner table

ID	Name	SS	DOB	ZIP
r1	Smith, John	123-45	1960/08/24	07932
r2	Smyth, Jon	123-45	1961/08/24	07932
r3	Smith, John	312-54	1995/07/25	98301
r4	Smith, J.	723-45	1960/08/24	98346
r5	Smith, J.	456-78	1975/12/11	98346

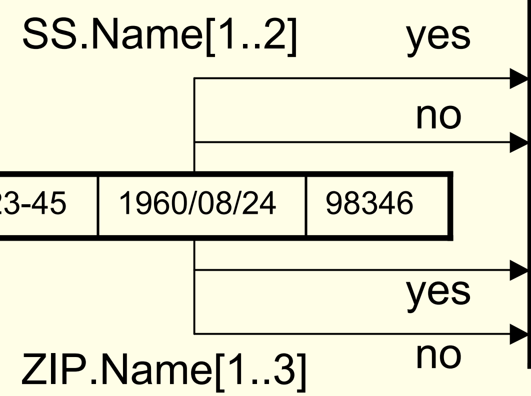
BigMatch [Y02]

- Goal: block/index matching records, based on multiple keys

- Example:

record from outer table

Smith, John	123-45	1960/08/24	98346



inner table

ID	Name	SS	DOB	ZIP
r1	Smith, John	123-45	1960/08/24	07932
r2	Smyth, Jon	123-45	1961/08/24	07932
r3	Smith, John	312-54	1995/07/25	98301
r4	Smith, J.	723-45	1960/08/24	98346
r5	Smith, J.	456-78	1975/12/11	98346

- Avoids multiple matches of the same pair

Use Dimension Hierarchies [ACG02]

- Goal: exploit dimension hierarchies for duplicate elimination
- Background: clustering categorical data [GKR98]
- Methodology: domain-independent, structure+text similarity
  - Use hierarchical grouping, instead of sorting, to focus search
  - “Structural” similarity based on overlap of children sets
  - Textual similarity based on weighted token set containment
  - Top-down processing of dimension hierarchy for efficiency
- Lesson: useful to consider group structure in addition to content

Use Dimension Hierarchies [ACG02]

- Goal: exploit dimension hierarchies for duplicate elimination

- Example:

AI	Address	CI	CI	City	SI	SI	State	YI	YI	Country
a1	10 Mountain Avenue	c1	c1	Summit	s1	s1	NJ	y1	y1	USA
a2	250 McCarter	c2	c2	Newark	s2	s2	New Jersey	y1	y2	United States
a3	250 McCarter Hwy	c3	c3	Newark	s3	s3	NJ	y2	y3	US
a4	10 Mountain	c4	c4	Summit	s4	s4	New Jersey	y2		
a5	10 Mountain Street	c5	c5	Summitt	s5	s5	NJ	y3		

Use Dimension Hierarchies [ACG02]

- Goal: exploit dimension hierarchies for duplicate elimination

- Example:

AI	Address	CI
a1	10 Mountain Avenue	c1
a2	250 McCarter	c2
a3	250 McCarter Hwy	c3
a4	10 Mountain	c4
a5	10 Mountain Street	c5

CI	City	SI
c1	Summit	s1
c2	Newark	s2
c3	Newark	s3
c4	Summit	s4
c5	Summitt	s5

SI	State	YI
s1	NJ	y1
s2	New Jersey	y1
s3	NJ	y2
s4	New Jersey	y2
s5	NJ	y1

YI	Country
y1	USA
y2	United States
~~y3~~	~~US~~

- Textual similarity

Use Dimension Hierarchies [ACG02]

- Goal: exploit dimension hierarchies for duplicate elimination

- Example:

AI	Address	CI
a1	10 Mountain Avenue	c1
a2	250 McCarter	c2
a3	250 McCarter Hwy	c3
a4	10 Mountain	c4
a5	10 Mountain Street	c5

CI	City	SI
c1	Summit	s1
c2	Newark	s2
c3	Newark	s3
c4	Summit	s4
c5	Summitt	s5

SI	State	YI
s1	NJ	y1
s2	New Jersey	y1
s3	NJ	y1
s4	New Jersey	y1
s5	NJ	y1

YI	Country
y1	USA
~~y2~~	~~United States~~
~~y3~~	~~US~~

- Structural similarity

Use Dimension Hierarchies [ACG02]

- Goal: exploit dimension hierarchies for duplicate elimination

- Example:

AI	Address	CI
a1	10 Mountain Avenue	c1
a2	250 McCarter	c2
a3	250 McCarter Hwy	c3
a4	10 Mountain	c4
a5	10 Mountain Street	c5

CI	City	SI
c1	Summit	s1
c2	Newark	s2
c3	Newark	s1
c4	Summit	s2
c5	Summitt	s1

SI	State	YI
s1	NJ	y1
s2	New Jersey	y1
~~s3~~	~~NJ~~	~~y1~~
~~s4~~	~~New Jersey~~	~~y1~~
~~s5~~	~~NJ~~	~~y1~~

YI	Country
y1	USA
~~y2~~	~~United States~~
~~y3~~	~~US~~

Use Dimension Hierarchies [ACG02]

- Goal: exploit dimension hierarchies for duplicate elimination

- Example:

AI	Address	CI
a1	10 Mountain Avenue	c1
a2	250 McCarter	c2
a3	250 McCarter Hwy	c3
a4	10 Mountain	c4
a5	10 Mountain Street	c5

CI	City	SI
c1	Summit	s1
c2	Newark	s1
c3	Newark	s1
c4	Summit	s1
c5	Summitt	s1

SI	State	YI
s1	NJ	y1
~~s2~~	~~New Jersey~~	~~y1~~
~~s3~~	~~NJ~~	~~y1~~
~~s4~~	~~New Jersey~~	~~y1~~
~~s5~~	~~NJ~~	~~y1~~

YI	Country
y1	USA
~~y2~~	~~United States~~
~~y3~~	~~US~~

Use Dimension Hierarchies [ACG02]

- Goal: exploit dimension hierarchies for duplicate elimination

- Example:

AI	Address	CI	CI	City	SI	SI	State	YI	YI	Country
a1	10 Mountain Avenue	c1	c1	Summit	s1	s1	NJ	y1	y1	USA
a2	250 McCarter	c2	c2	Newark	s1	~~s2~~	~~New Jersey~~	~~y1~~	~~y2~~	~~United States~~
a3	250 McCarter Hwy	c2	~~c3~~	~~Newark~~	~~s1~~	~~s3~~	~~NJ~~	~~y1~~	~~y3~~	~~US~~
a4	10 Mountain	c1	~~c4~~	~~Summit~~	~~s1~~	~~s4~~	~~New Jersey~~	~~y1~~		
a5	10 Mountain Street	c1	~~c5~~	~~Summitt~~	~~s1~~	~~s5~~	~~NJ~~	~~y1~~		

Use Dimension Hierarchies [ACG02]

- Goal: exploit dimension hierarchies for duplicate elimination
- Example:

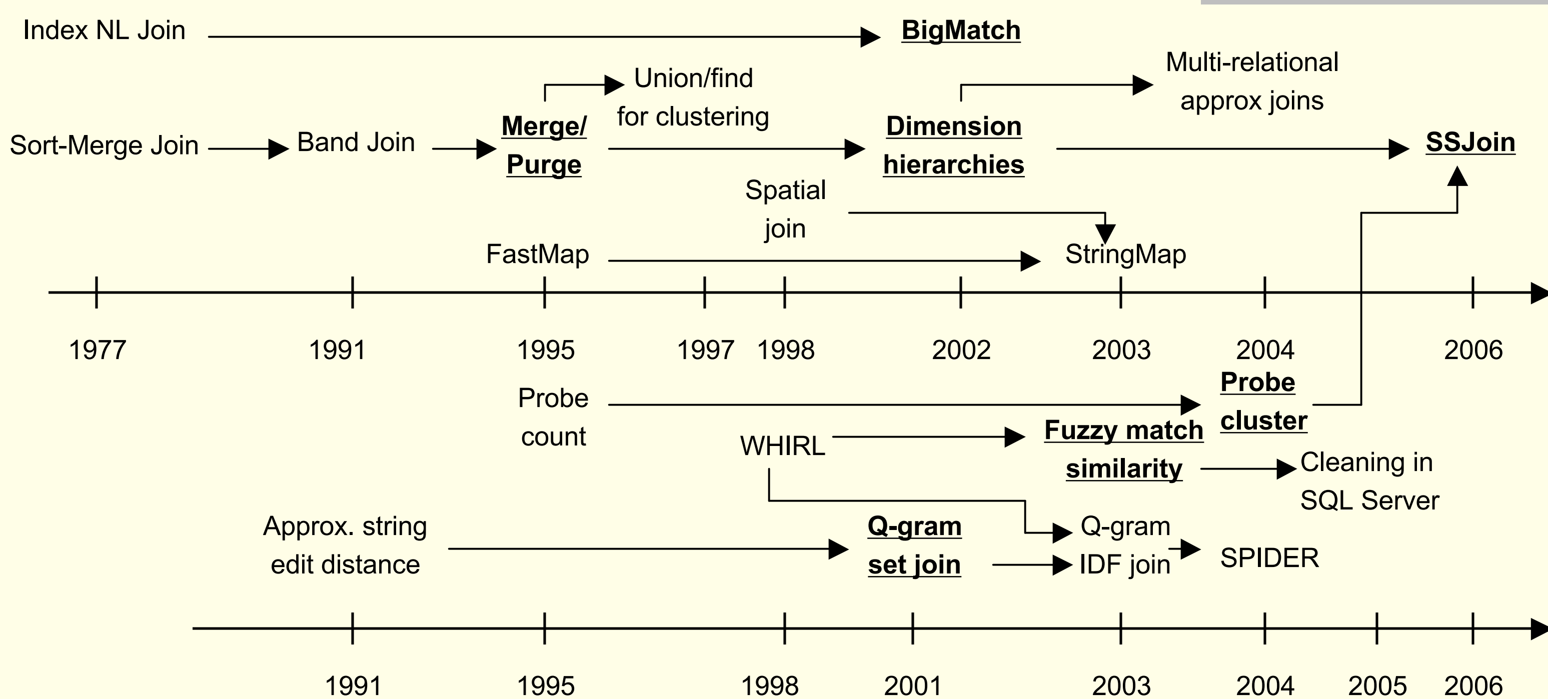
AI	Address	CI
a1	10 Mountain Avenue	c1
~~a2~~	~~250 McCarter~~	c2
a3	250 McCarter Hwy	c2
~~a4~~	~~10 Mountain~~	c1
~~a5~~	~~10 Mountain Street~~	c1

CI	City	SI
c1	Summit	s1
c2	Newark	s1
~~c3~~	~~Newark~~	~~s1~~
~~c4~~	~~Summit~~	~~s1~~
~~c5~~	~~Summitt~~	~~s1~~

SI	State	YI
s1	NJ	y1
~~s2~~	~~New Jersey~~	~~y1~~
~~s3~~	~~NJ~~	~~y1~~
~~s4~~	~~New Jersey~~	~~y1~~
~~s5~~	~~NJ~~	~~y1~~

YI	Country
y1	USA
~~y2~~	~~United States~~
~~y3~~	~~US~~

Historical Timelines



Q-gram Set Join [GIJ+01]

- Goal: compute thresholded edit distance join on string attributes
- Background: combinatorial pattern matching [JU91]
- Methodology: domain-independent, edit distance similarity
  - Extract set of all overlapping q-grams Q(s) from string s
  - $ED(s_1,s_2) \leq d \rightarrow |Q(s_1) \cap Q(s_2)| \geq \max(|s_1|,|s_2|) - (d-1)*q - 1$
  - Cheap filters (length, count, position) to prune non-matches
  - Pure SQL solution: cost-based join methods
- Lesson: reduce approximate join to aggregated set intersection

Q-gram Set Join [GIJ+01]

- Goal: compute thresholded edit distance join on string attributes
- Example:

ID	Name
r1	Srivastava
r2	Shrivastava
r3	Shrivastav

Q-gram Set Join [GIJ+01]

- Goal: compute thresholded edit distance join on string attributes
- Example:

ID	Name	3-grams
r1	Srivastava	##s, #sr, sri, riv, iva, vas, ast, sta, tav, ava, va$, a$$
r2	Shrivastava	##s, #sh, shr, hri, riv, iva, vas, ast, sta, tav, ava, va$, a$$
r3	Shrivastav	

  - $ED(s_1,s_2) \leq d \rightarrow |Q(s_1) \cap Q(s_2)| \geq \max(|s_1|,|s_2|) - (d-1)*q - 1$
  - $ED(r1, r2) = 1$, $|Q(r1) \cap Q(r2)| = 10$

Q-gram Set Join [GIJ+01]

- Goal: compute thresholded edit distance join on string attributes
- Example:

ID	Name	3-grams
r1	Srivastava	##s, #sr, sri, riv, iva, vas, ast, sta, tav, ava, va$, a$$
r2	Shrivastava	
r3	Shrivastav	##s, #sh, shr, hri, riv, iva, vas, ast, sta, tav, av$, v$$

  - $ED(s_1,s_2) \le d \rightarrow |Q(s_1) \cap Q(s_2)| \ge \max(|s_1|,|s_2|) - (d-1)*q - 1$
  - $ED(r1, r2) = 2$, $|Q(r1) \cap Q(r2)| = 7$

Q-gram Set Join [GIJ+01]

- Goal: compute thresholded edit distance join on string attributes

- Example:

ID	Name
r1	Srivastava
r2	Shrivastava
r3	Shrivastav

Q

ID	Qg
r1	##s
r1	#sr
r1	sri
r1	riv
r1	iva
r1	vas
r1	ast
r1	sta
r1	tav
r1	ava
r1	va$
r1	a$$

ID	Qg
r3	##s
r3	#sh
r3	shr
r3	hri
r3	riv
r3	iva
r3	vas
r3	ast
r3	sta
r3	tav
r3	av$
r3	v$$

Q-gram Set Join [GIJ+01]

- Goal: compute thresholded edit distance join on string attributes
- Example:

ID	Name
r1	Srivastava
r2	Shrivastava
r3	Shrivastav

```
SELECT Q1.ID, Q2.ID
FROM Q AS Q1, Q AS Q2
WHERE Q1.Qg = Q2.Qg
GROUP BY Q1.ID, Q2.ID
HAVING COUNT(*) > T
```

Q

ID	Qg
r1	##s
r1	#sr
r1	sri
r1	riv
r1	iva
r1	vas
r1	ast
r1	sta
r1	tav
r1	ava
r1	va$
r1	a$$

ID	Qg
r3	##s
r3	#sh
r3	shr
r3	hri
r3	riv
r3	iva
r3	vas
r3	ast
r3	sta
r3	tav
r3	av$
r3	v$$

Fuzzy Match Similarity [CGGM03]

- Goal: identify K "closest" reference records in on-line setting
- Background: IDF weighted cosine similarity, WHIRL [C98]
- Methodology: domain-independent, IDF+ED similarity
  - Similarity metric based on IDF weighted token edit distance
  - Approximate similarity metric using Jaccard on q-gram sets
  - Small error tolerant index table, sharing of minhash q-grams
  - Optimistic short circuiting exploits large token IDF weights
- Lesson: IDF weighting useful to capture erroneous tokens

Fuzzy Match Similarity [CGGM03]

- Goal: identify K “closest” reference records in on-line setting
- Example:

reference table

ID	OrgName	City	State	ZIP
r1	Boeing Company	Seattle	WA	98004
r2	Bon Corporation	Seattle	WA	98014
r3	Companions	Seattle	WA	98024

best ED match

input record

Beoing Corporation	Seattle	WA	98004

Fuzzy Match Similarity [CGGM03]

- Goal: identify K “closest” reference records in on-line setting
- Example:

reference table

ID	OrgName	City	State	ZIP
r1	Boeing Company	Seattle	WA	98004
r2	Bon Corporation	Seattle	WA	98014
r3	Companions	Seattle	WA	98024

best FMS match

input record

Beoing Corporation	Seattle	WA	98004

Fuzzy Match Similarity [CGGM03]

- Goal: identify K "closest" reference records in on-line setting

- Example:

reference table

ID	OrgName	City	State	ZIP
r1	Boeing Company	Seattle	WA	98004
r2	Bon Corporation	Seattle	WA	98014
r3	Companions	Seattle	WA	98024

input record

Beoing Corporation	Seattle	WA	98004

[eoi, ing] [orp, ati] [sea, ttl] [wa] [980, 004]

all minhash q-grams

ETI table

Qg	MHC	Col	Freq	TIDList
ing	2	1	1	{r1}
orp	1	1	1	{r2}
sea	1	2	3	{r1, r2, r3}
004	2	4	1	{r1}

Fuzzy Match Similarity [CGGM03]

- Goal: identify K “closest” reference records in on-line setting
- Example:

reference table

ID	OrgName	City	State	ZIP
r1	Boeing Company	Seattle	WA	98004
r2	Bon Corporation	Seattle	WA	98014
r3	Companions	Seattle	WA	98024

input record

Beoing Corporation	Seattle	WA	98004

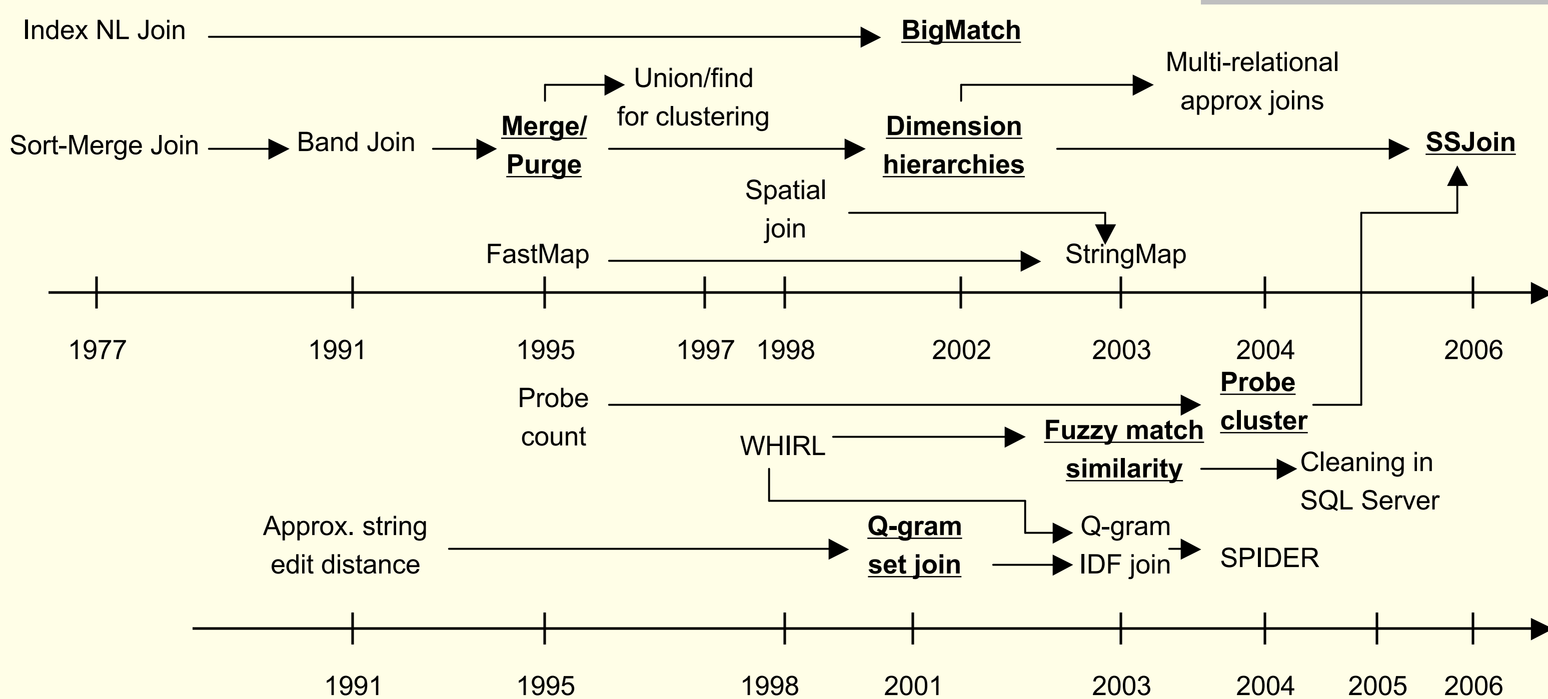
[eoi, ing] [orp, ati] [sea, ttl] [wa] [980, 004]

ETI table

Qg	MHC	Col	Freq	TIDList
ing	2	1	1	{r1}
orp	1	1	1	{r2}
sea	1	2	3	{r1, r2, r3}
004	2	4	1	{r1}

optimistic short circuiting

Historical Timelines



Probe-Cluster: Set Joins [SK04]

- Goal: generic algorithm for set join based on similarity predicate
- Background: IR and probe count using inverted index [TF95]
- Methodology: domain-independent, weighted set similarity
  - Map a string to a set of elements (words, q-grams, etc.)
  - Build inverted lists on individual set elements
  - Optimization: process skewed lists in increasing size order
  - Optimization: sort lists in decreasing order of record sizes
- Lesson: IR query optimizations useful for approximate joins

Probe-Cluster: Set Joins [SK04]

- Goal: generic algorithm for set join based on similarity predicate
- Example:

ID	SVA
r1	{##s, #sr, sri, riv, iva, vas, ast, sta, tav, ava, va$, a$$}
r2	{##s, #sh, shr, hri, riv, iva, vas, ast, sta, tav, ava, va$, a$$}
r3	{##s, #sh, shr, hri, riv, iva, vas, ast, sta, tav, av$, v$$}

Inverted index

SE	IDs
##s	r1, r2, r3
#sr	r1
#sh	r2, r3
sri	r1
shr	r2, r3
hri	r2, r3
riv	r1, r2, r3
…	…
tav	r1, r2, r3
ava	r1, r2
…	…
v$$	r3

Probe-Cluster: Set Joins [SK04]

- Goal: generic algorithm for set join based on similarity predicate
- Example:

ID	SVA
r1	{##s, #sr, sri, riv, iva, vas, ast, sta, tav, ava, va$, a$$}
r2	{##s, #sh, shr, hri, riv, iva, vas, ast, sta, tav, ava, va$, a$$}
r3	{##s, #sh, shr, hri, riv, iva, vas, ast, sta, tav, av$, v$$}

Inverted index

SE	IDs
##s	r2, r1, r3
#sr	r1
#sh	r2, r3
sri	r1
shr	r2, r3
hri	r2, r3
riv	r2, r1, r3
…	…
tav	r2, r1, r3
ava	r2, r1
…	…
v$$	r3

- Sort lists in decreasing order of record sizes

Probe-Cluster: Set Joins [SK04]

- Goal: generic algorithm for set join based on similarity predicate

- Example:

ID	SVA
r1	{##s, #sr, sri, riv, iva, vas, ast, sta, tav, ava, va$, a$$}
r2	{##s, #sh, shr, hri, riv, iva, vas, ast, sta, tav, ava, va$, a$$}
r3	{##s, #sh, shr, hri, riv, iva, vas, ast, sta, tav, av$, v$$}

- Process skewed lists in increasing size order

Inverted index

SE	IDs
##s	r2, r1, r3
#sr	r1
#sh	r2, r3
sri	r1
shr	r2, r3
hri	r2, r3
riv	r2, r1, r3
…	…
tav	r2, r1, r3
ava	r2, r1
…	…
v$$	r3

Probe-Cluster: Set Joins [SK04]

- Goal: generic algorithm for set join based on similarity predicate

- Example:

ID	SVA
r1	{##s, #sr, sri, riv, iva, vas, ast, sta, tav, ava, va$, a$$}
r2	{##s, #sh, shr, hri, riv, iva, vas, ast, sta, tav, ava, va$, a$$}
r3	{##s, #sh, shr, hri, riv, iva, vas, ast, sta, tav, av$, v$$}

Inverted index

SE	IDs
##s	r2, r1, r3
#sr	r1
#sh	r2, r3
sri	r1
shr	r2, r3
hri	r2, r3
riv	r2, r1, r3
…	…
tav	r2, r1, r3
ava	r2, r1
…	…
v$$	r3

- Process skewed lists in increasing size order

Probe-Cluster: Set Joins [SK04]

- Goal: generic algorithm for set join based on similarity predicate

- Example:

ID	SVA
r1	{##s, #sr, sri, riv, iva, vas, ast, sta, tav, ava, va$, a$$}
r2	{##s, #sh, shr, hri, riv, iva, vas, ast, sta, tav, ava, va$, a$$}
r3	{##s, #sh, shr, hri, riv, iva, vas, ast, sta, tav, av$, v$$}

Inverted index

SE	IDs
##s	r2, r1, r3
#sr	r1
#sh	r2, r3
sri	r1
shr	r2, r3
hri	r2, r3
riv	r2, r1, r3
…	…
tav	r2, r1, r3
ava	r2, r1
…	…
v$$	r3

- Process skewed lists in increasing size order

SSJoin: Relational Operator [CGK06]

- Goal: generic algorithm for set join based on similarity predicate
- Background: Probe-Cluster, dimension hierarchies, q-gram join
- Methodology: domain-independent, weighted set similarity
  - Compare strings based on sets associated with each string
  - Problem: Overlap(s1, s2) ≥ threshold
  - Optimization: high set overlap → overlap of ordered subsets
  - SQL implementation using equijoins, cost-based plans
- Lesson: Generic algorithms can be supported in DBMS

SSJoin: Relational Operator [CGK06]

- Goal: generic algorithm for set join based on similarity predicate
- Example:

ID	Name
r1	Srivastava
r4	Srivastav

SELECT Q1.ID, Q2.ID
FROM Q AS Q1, Q AS Q2
WHERE Q1.Qg = Q2.Qg
GROUP BY Q1.ID, Q2.ID
HAVING COUNT(*) > 8

Q

ID	Qg
r1	##s
r1	#sr
r1	sri
r1	riv
r1	iva
r1	vas
r1	ast
r1	sta
r1	tav
r1	ava
r1	va$
r1	a$$

ID	Qg
r4	##s
r4	#sr
r4	sri
r4	riv
r4	iva
r4	vas
r4	ast
r4	sta
r4	tav
r4	av$
r4	v$$

SSJoin: Relational Operator [CGK06]

- Goal: generic algorithm for set join based on similarity predicate

- Example:

ID	Name
r1	Srivastava
r4	Srivastav

```
SELECT Q1.ID, Q2.ID
FROM Q AS Q1, Q AS Q2
WHERE Q1.Qg = Q2.Qg
GROUP BY Q1.ID, Q2.ID
```
~~HAVING COUNT(*) > 8~~

Q

ID	Qg
r1	tav
r1	ava
r1	va$
r1	a$$

ID	Qg
r4	##s
r4	#sr
r4	sri
r4	riv
r4	iva
r4	vas
r4	ast
r4	sta
r4	tav
r4	av$
r4	v$$

- Optimization: use any 4 q-grams of r1 with all of r4

SSJoin: Relational Operator [CGK06]

- Goal: generic algorithm for set join based on similarity predicate
- Example:

ID	Name
r1	Srivastava
r4	Srivastav

```
SELECT Q1.ID, Q2.ID
FROM Q AS Q1, Q AS Q2
WHERE Q1.Qg = Q2.Qg
GROUP BY Q1.ID, Q2.ID
```

~~HAVING COUNT(*) > 8~~

- Optimization: use any 3 q-grams of r4

Q

ID	Qg
r1	##s
r1	#sr
r1	sri
r1	riv
r1	iva
r1	vas
r1	ast
r1	sta
r1	tav
r1	ava
r1	va$
r1	a$$

ID	Qg
r4	sri
r4	av$
r4	v$$

SSJoin: Relational Operator [CGK06]

- Goal: generic algorithm for set join based on similarity predicate

- Example:

ID	Name
r1	Srivastava
r4	Srivastav

SELECT Q1.ID, Q2.ID
FROM Q AS Q1, Q AS Q2
WHERE Q1.Qg = Q2.Qg
GROUP BY Q1.ID, Q2.ID
~~HAVING COUNT(*) > 8~~

Q

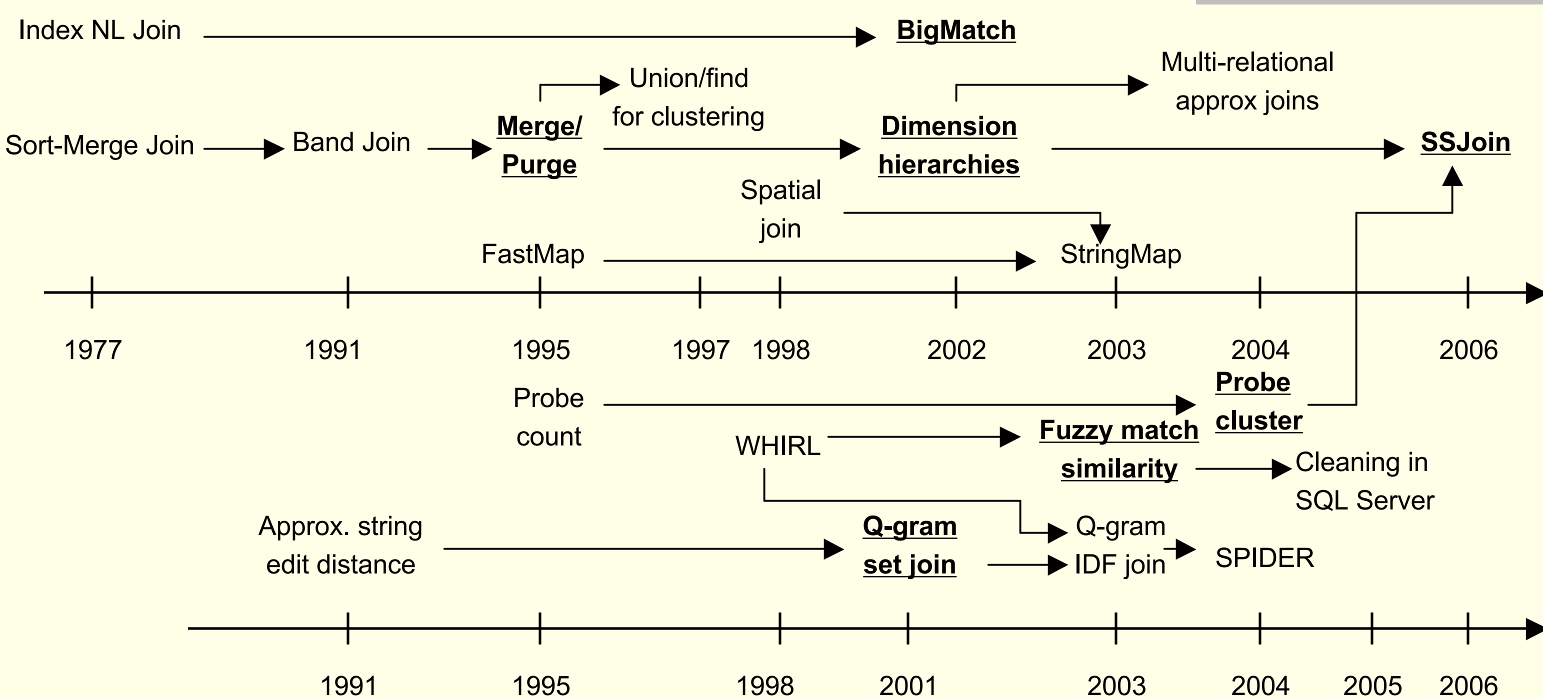
ID	Qg
r1	iva
r1	ast
r1	ava
r1	a$$

ID	Qg
r4	iva
r4	ast
r4	av$

- Optimization: use ordered 4 q-grams of r1 and 3 q-grams of r4

- Suggested ordering: based on decreasing IDF weights

Historical Timelines



Commercial Systems: Comparisons

Commercial System	Record Linkage Methodology	Distance Metrics Supported	Domain-Specific Matching	Additional Data Quality Support
SQL Server Integration Services 2005	Fuzzy Lookup; Fuzzy Grouping; uses Error Tolerant Index	customized, domain-independent: edit distance; number, order, freq. of tokens	unknown	unknown
OracleBI Warehouse Builder 10gR2 "Paris"	match-merge rules; deterministic and probabilistic matching	Jaro-Winkler; double metaphone	name & address parse; match; standardize: 3rd party vendors	data profiling; data rules; data auditors
IBM's Entity Analytic Solutions, QualityStage	probabilistic matching (information content); multi-pass blocking; rules-based merging	wide variety of fuzzy matching functions	name recognition; identity resolution; relationship resolution: EAS	data profiling; standardization; trends and anomalies;

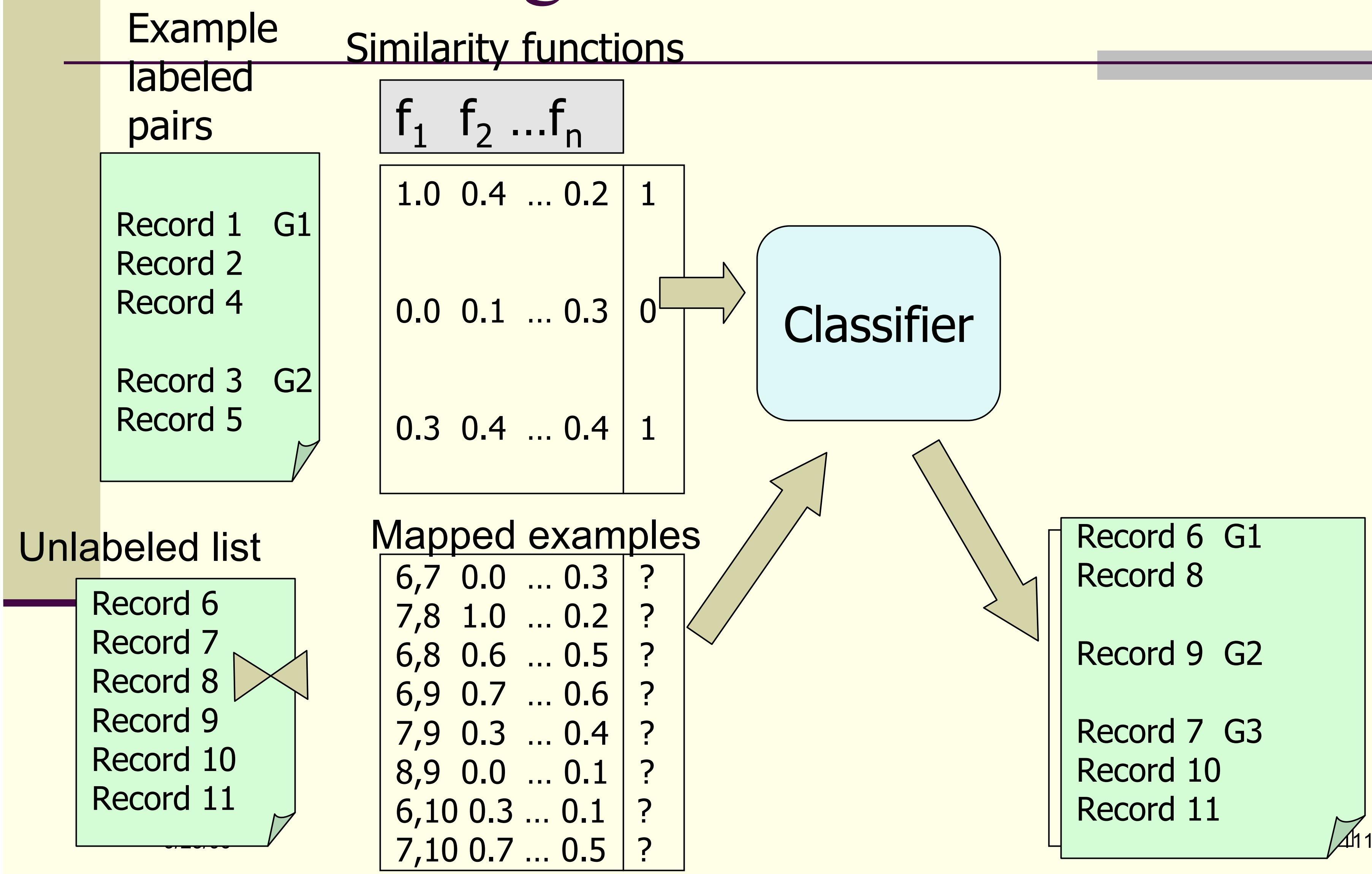
Outline

- Part I: Motivation, similarity measures (90 min)
- Part II: Efficient algorithms for approximate join (60 min)
- Part III: Clustering/partitioning algorithms (30 min)

Partitioning/collective deduplication

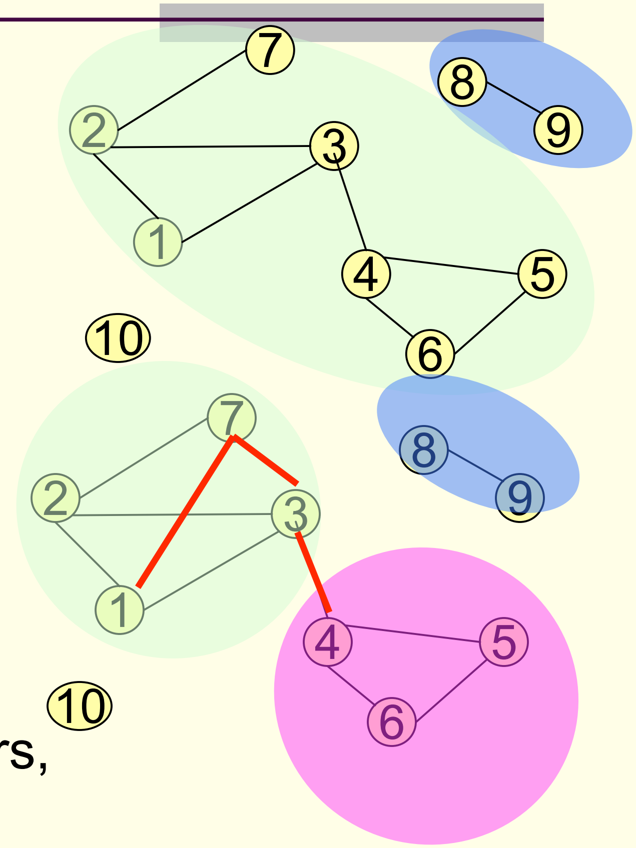
- Single-entity types
  - A is same as B if both are same as C.
- Multiple linked entity types
  - If paper A is same as paper B then venue of A is the same as venue of B.

Partitioning data records



Creating partitions

- Transitive closure
  - Dangers: unrelated records collapsed into a single cluster
- Correlation clustering (Bansal et al 2002)
  - Partition to minimize total disagreements
    - Edges across partitions
    - Missing edges within partition
  - More appealing than clustering:
    - No magic constants: number of clusters, similarity thresholds, diameter, etc
  - Extends to real-valued scores
  - NP Hard: many approximate algorithms



3 disagreements

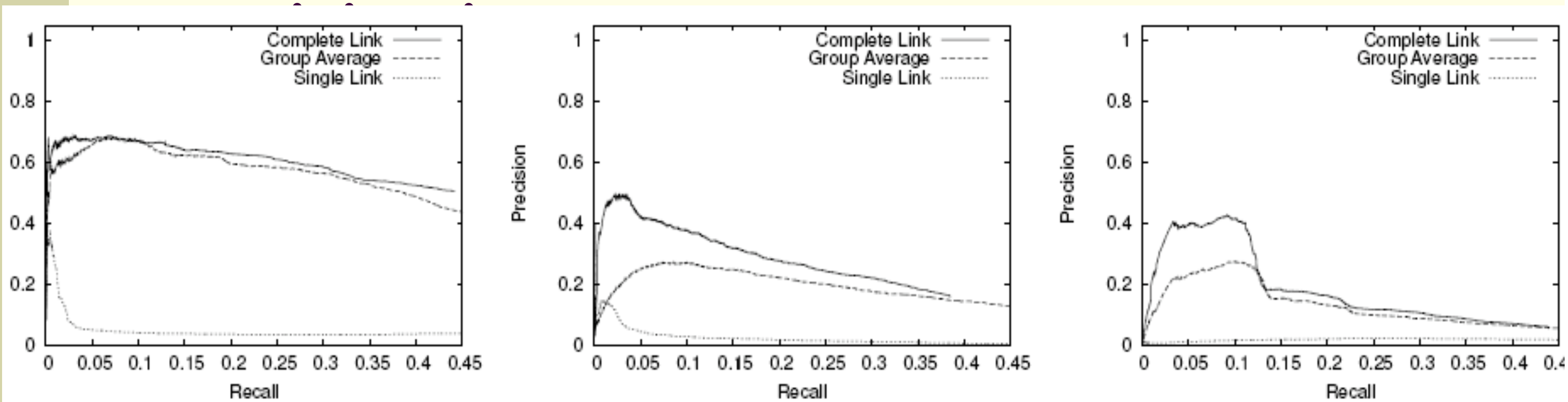
Algorithms for correlation clustering

- Integer programming formulation (Charikar 03)
  - X_{ij} = 1 if i and j in same partition, 0 otherwise

$$\min \sum_{ij \in edges} (1 - x_{ij}) + \sum_{ij \notin edges} x_{ij}$$
$$\text{such that} \quad x_{ij} \in \{0,1\}$$
$$x_{ij} + x_{jk} \leq 1 + x_{ik} \qquad \text{(partitioning constraint)}$$

  - Impractical: $O(n^3)$ constraints
- Practical substitutes (Heuristics, no guarantees)
  - Agglomerative clustering: repeatedly merge closest clusters
    - Efficient implementation possible via heaps (BG 2005)
    - Definition of closeness subject to tuning
      - Greatest reduction in error
      - Average/Max/Min similarity

Empirical results on data



Digital cameras — Camcoder — Luggage

(From: Bilenko et al, 2005)

- Setup: Online comparison shopping,
  - Fields: name, model, description, price
  - Learner: Online perceptron learner
- Complete-link clustering >> single-link clustering(transitive closure)
- An issue: when to stop merging clusters

Other methods of partitioning

[Chaudhuri et al ICDE 2005]

- Partitions are compact and relatively far from other points
- A Partition has to satisfy a number of criteria
  - Points within partition closer than any points outside
  - #points within p-neighborhood of each partition < c
  - Either number of points in partition < K, or diameter < θ

Algorithm

Consider case where partitions required to be of size < K ➔ if partition P_j of size m in output then

- m-nearest neighbors of all r in P_i is P_i
- Neighborhood of each point is sparse

For each record, do efficient index probes to get

- Get K nearest neighbors
- Count of number of points in p-neighborhood for each m nearest neighbors

Form pairs and perform grouping based on above insight to find groups

Summary: partitioning

- Transitive closure is a bad idea
- No verdict yet on best alternative
- Difficult to design an objective and algorithms
- Correlation clustering
  - Reasonable objective with a skewed scoring function
  - Poor algorithms
- Greedy agglomerative clustering algorithms ok
  - Greatest minimum similarity (complete-link), benefit
  - Reasonable performance with heap-based implementation
- Dense/Sparse partitioning
  - Positives: Declarative objective, efficient algorithm
  - Parameter retuning across domains
- Need comparison between complete-link, Dense/Sparse, and Correlation clustering.

Collective de-duplication: multi-attribute

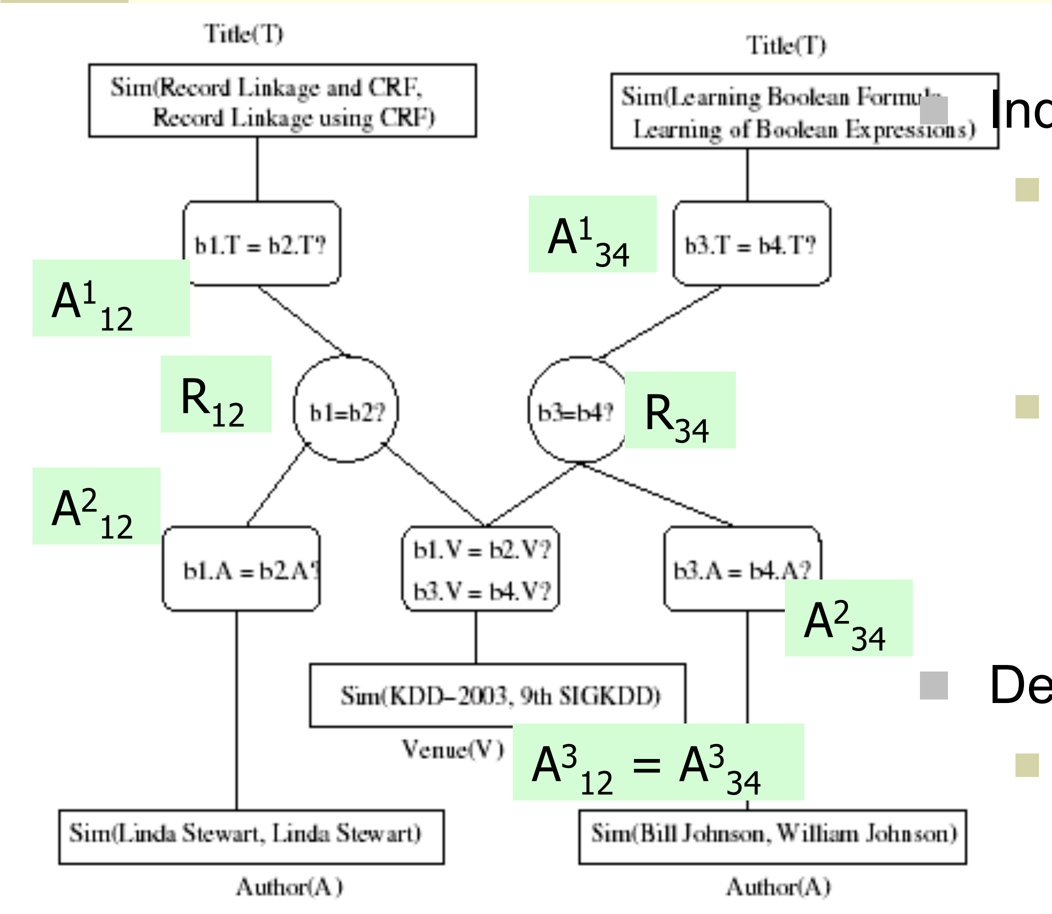
Record	Title a^1	Author a^2	Venue a^3
b_1	"Record Linkage using CRFs"	"Linda Stewart"	"KDD-2003"
b_2	"Record Linkage using CRFs"	"Linda Stewart"	"9th SIGKDD"
b_3	"Learning Boolean Formulas"	"Bill Johnson"	"KDD-2003"
b_4	"Learning of Boolean Expressions"	"William Johnson"	"9th SIGKDD"

Associate variables for predictions
for each attribute k
each record pair (i,j) A^k_{ij}

for each record pair R_{ij}

from Parag & Domingos 2005

Dependency graph Scoring functions



- Independent scores
  - $s_k(A^k, a_i, a_j)$ Attribute-level
    - Any classifier on various text similarities of attribute pairs
  - $s(R, b_i, b_j)$ Record-level
    - Any classifier on various similarities of all k attribute pairs
- Dependency scores
  - $d_k(A^k, R)$: record pair, attribute pair

	0	1
0	4	2
1	1	7

Joint de-duplication steps

- Jointly pick 0/1 labels for all record pairs R_{ij} and all K attribute pairs A^k_{ij} to maximize

$$\sum_{ij}[s(R_{ij})+\sum_{k}s_k(A_{ij}^k)+d_k(R_{ij},A_{ij}^k)]$$

- When dependency scores associative
  - $d_k(1,1) + d_k(0,0) >= d_k(1,0)+d_k(0,1)$
  - Can find optimal scores through graph MINCUT
- Assigning scores
  - Manually as in Levy et. al
  - Example-based training as in Domingos et al
    - Creates a weighted feature-based log-linear model
      - $s(R_{ij}) = w_1*sim(a^1_i,a^1_j) + \ldots.+w_k*sim(a^k_i, a_j^k)$
    - Very flexible and powerful.

Other issues and approaches

- Partitioning
  - Transitive-closure as a post processing
  - Results:

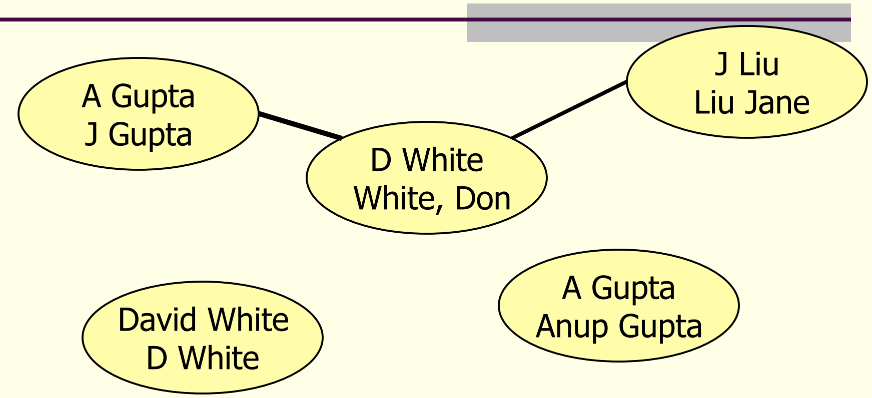
	Citation		Author		Venue	
	P	T	P	T	P	T
Independent	87	85	79	89	49	59
Collective	86	89	89	89	86	82

  - Collective deduplication
    - does not help whole citations,
    - helps attributes
  - Transitive closure can cause drop in accuracy
- Combined partitioning and linked dedup
  - Dong, HaLevy, Madhavan (SIGMOD 2005)
  - Bhattacharya and Getoor (2005)

Collective linkage: set-oriented data

(Bhattacharya and Getoor, 2005)

P1	D White, J Liu, A Gupta
P2	Liu, Jane & J Gupta & White, Don
P3	Anup Gupta
P4	David White



Scoring functions

- $S(A_{ij})$ Attribute-level
  - Text similarity
- $S(A_{ij}, N_{ij})$ Dependency with labels of co-author set
  - Fraction of co-author set assigned label 1.
- Final score:
  - $a\, s(A_{ij}) + (1-a)\, s(A_{ij}, N_{ij})$
  - a is the only parameter

Algorithm

Greedy agglomerative clustering

- Merge author clusters with highest score
- Redefine similarity between clusters of authors instead of single authors
  - Max of author-level similarity

Open Problem: Inside or Outside?

- Issue: optimizable processing in a relational database
- Background
  - Declarative data cleaning in AJAX [GFS+01]
  - Q-gram based metrics, SPIDER [GIJ+01,GIKS03,KMS04]
  - SSJoin [CGK06]
  - Compact sets, sparse neighborhood [CGM05]
- Goal: express arbitrary record linkage in SQL

Open Problem: Multi-Table Joins

- Issue: information in auxiliary tables can aid matching
- Background
  - Hierarchical models [ACG02]
  - Iterative matching [BG04]
  - Graphical models [KMC05]
- Goal: efficient multi-table approximate joins

Open Problem: Benchmarking

- Issue: many algorithms and similarity measures, no benchmarks
- Background
  - Comparing quality of different similarity measures [CRF03]
- Goal: develop standard benchmarks (queries, data generation)

Conclusions

- Record linkage is critical when data quality is poor
  - Similarity metrics
  - Efficient sub-quadratic approximate join algorithms
  - Efficient clustering algorithms

- Wealth of challenging technical problems
  - Sophisticated similarity metrics, massive data sets
  - Important to work with real datasets

References

- [ACG02] Rohit Ananthakrishna, Surajit Chaudhuri, Venkatesh Ganti: Eliminating Fuzzy Duplicates in Data Warehouses. VLDB 2002: 586-597
- [BD83] Dina Bitton, David J. DeWitt: Duplicate Record Elimination in Large Data Files. ACM Trans. Database Syst. 8(2): 255-265 (1983)
- [BE77] Mike W. Blasgen, Kapali P. Eswaran: Storage and Access in Relational Data Bases. IBM Systems Journal 16(4): 362-377 (1977)
- [BG04] Indrajit Bhattacharya, Lise Getoor: Iterative record linkage for cleaning and integration. DMKD 2004: 11-18
- [C98] William W. Cohen: Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity. SIGMOD Conference 1998: 201-212
- [C00] William W. Cohen: Data integration using similarity joins and a word-based information representation language. ACM Trans. Inf. Syst. 18(3): 288-321 (2000)
- [CCZ02] Peter Christen, Tim Churches, Xi Zhu: Probabilistic name and address cleaning and standardization. Australasian Data Mining Workshop 2002.
- [CGGM04] Surajit Chaudhuri, Kris Ganjam, Venkatesh Ganti, Rajeev Motwani: Robust and Efficient Fuzzy Match for Online Data Cleaning. SIGMOD Conference 2003: 313-324
- [CGG+05] Surajit Chaudhuri, Kris Ganjam, Venkatesh Ganti, Rahul Kapoor, Vivek R. Narasayya, Theo Vassilakis: Data cleaning in microsoft SQL server 2005. SIGMOD Conference 2005: 918-920
- [CGK06] Surajit Chaudhuri, Venkatesh Ganti, Raghav Kaushik: A primitive operator for similarity joins in data cleaning. ICDE 2006.
- [CGM05] Surajit Chaudhuri, Venkatesh Ganti, Rajeev Motwani: Robust Identification of Fuzzy Duplicates. ICDE 2005: 865-876
- [CRF03] William W. Cohen, Pradeep Ravikumar, Stephen E. Fienberg: A Comparison of String Distance Metrics for Name-Matching Tasks. IIWeb 2003: 73-78

References

- [DJ03] Tamraparni Dasu, Theodore Johnson: Exploratory Data Mining and Data Cleaning John Wiley 2003
- [DNS91] David J. DeWitt, Jeffrey F. Naughton, Donovan A. Schneider: An Evaluation of Non-Equijoin Algorithms. VLDB 1991: 443-452
- [DWI02] Data Warehousing Institute report 2002
- [E00] Larry English: Plain English on Data Quality: Information Quality Management: The Next Frontier. DM Review Magazine: April 2000. http://www.dmreview.com/article_sub.cfm?articleId=2073
- [FL95] Christos Faloutsos, King-Ip Lin: FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets. SIGMOD Conference 1995: 163-174
- [FS69] I. Fellegi, A. Sunter: A theory of record linkage. Journal of the American Statistical Association, Vol 64. No 328, 1969
- [G98] D. Gusfield: Algorithms on strings, trees and sequences. Cambridge university press 1998
- [GFS+01] Helena Galhardas, Daniela Florescu, Dennis Shasha, Eric Simon, Cristian-Augustin Saita: Declarative Data Cleaning: Language, Model, and Algorithms. VLDB 2001: 371-380
- [GIJ+01] Luis Gravano, Panagiotis G. Ipeirotis, H. V. Jagadish, Nick Koudas, S. Muthukrishnan, Divesh Srivastava: Approximate String Joins in a Database (Almost) for Free. VLDB 2001: 491-500
- [GIKS03] Luis Gravano, Panagiotis G. Ipeirotis, Nick Koudas, Divesh Srivastava: Text joins in an RDBMS for web data integration. WWW 2003: 90-101
- [GKMS04] S. Guha, N. Koudas, A. Marathe, D. Srivastava : Merging the results of approximate match operations. VLDB 2004.
- [GKR98] David Gibson, Jon M. Kleinberg, Prabhakar Raghavan: Clustering Categorical Data: An Approach Based on Dynamical Systems. VLDB 1998: 311-322

References

- [HS95] Mauricio A. Hernández, Salvatore J. Stolfo: The Merge/Purge Problem for Large Databases. SIGMOD Conference 1995: 127-138
- [HS98] Gísli R. Hjaltason, Hanan Samet: Incremental Distance Join Algorithms for Spatial Databases. SIGMOD Conference 1998: 237-248
- [J89] M. A. Jaro: Advances in record linkage methodology as applied to matching the 1985 census of Tampa, Florida. Journal of the American Statistical Association 84: 414-420.
- [JLM03] Liang Jin, Chen Li, Sharad Mehrotra: Efficient Record Linkage in Large Data Sets. DASFAA 2003
- [JU91] Petteri Jokinen, Esko Ukkonen: Two Algorithms for Approximate String Matching in Static Texts. MFCS 1991: 240-248
- [KL51] S. Kullback, R. Liebler : On information and sufficiency. The annals of mathematical statistics 22(1): 79-86. 1959.
- [KMC05] Dmitri V. Kalashnikov, Sharad Mehrotra, Zhaoqi Chen: Exploiting Relationships for Domain-Independent Data Cleaning. SDM 2005
- [KMS04] Nick Koudas, Amit Marathe, Divesh Srivastava: Flexible String Matching Against Large Databases in Practice. VLDB 2004: 1078-1086
- [KMS05] Nick Koudas, Amit Marathe, Divesh Srivastava: SPIDER: flexible matching in databases. SIGMOD Conference 2005: 876-878
- [LLL00] Mong-Li Lee, Tok Wang Ling, Wai Lup Low: IntelliClean: a knowledge-based intelligent data cleaner. KDD 2000: 290-294
- [ME96] Alvaro E. Monge, Charles Elkan: The Field Matching Problem: Algorithms and Applications. KDD 1996: 267-270

References

- [ME97] Alvaro E. Monge, Charles Elkan: An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records. DMKD 1997
- [RY97] E. Ristad, P. Yianilos : Learning string edit distance. IEEE Pattern analysis and machine intelligence 1998.
- [S83] Gerry Salton : Introduction to modern information retrieval. McGraw Hill 1987.
- [SK04] Sunita Sarawagi, Alok Kirpal: Efficient set joins on similarity predicates. SIGMOD Conference 2004: 743-754
- [TF95] Howard R. Turtle, James Flood: Query Evaluation: Strategies and Optimizations. Inf. Process. Manage. 31(6): 831-850 (1995)
- [TKF01] S. Tejada, C. Knoblock, S. Minton : Learning object identification rules for information integration. Information Systems, Vol 26, No 8, 607-633, 2001.
- [W94] William E. Winkler: Advanced methods for record linkage. Proceedings of the section on survey research methods, American Statistical Association 1994: 467-472
- [W99] William E. Winkler: The state of record linkage and current research problems. IRS publication R99/04 (http://www.census.gov/srd/www/byname.html)
- [Y02] William E. Yancey: BigMatch: A program for extracting probable matches from a large file for record linkage. RRC 2002-01. Statistical Research Division, U.S. Bureau of the Census.