A Survey of Entity Resolution and Record Linkage Methodologies

David Guy Brizan and Abdullah Uz Tansel
PhD Program in Computer Science
The Graduate Center[1], Baruch College[2]
The City University of New York
365 Fifth Avenue, New York, NY 10016-4309

ABSTRACT

A great deal of research is focused on formation of a data warehouse. This is an important area of research as it could save many computation cycles and thus allow accurate information provided to the right people at the right time. Two considerations when forming a data warehouse are data cleansing (including entity resolution) and with schema integration (including record linkage). Uncleansed and fragmented data requires time to decipher and may lead to increased costs for an organization, so data cleansing and schema integration can save a great many (human) computation cycles and can lead to higher organizational efficiency. In this study we survey the literature for the methodologies proposed or developed for entity resolution and record linkage. This survey provides a foundation for solving many problems in data warehousing. For instance, little or no research has been directed at the problem of maintenance of cleansed and linked relations.

INTRODUCTION

A database is an electronic (digital) representation of a real (physical or logical) world made up of entities and their relationships (Elmasri & Navathe, 1994, p. 2). Although a relational database may consist of many relations (two dimensional tables) we will largely focus on a single relation and formulate and discuss the issues in the context of one relation.

Each physical or logical entity of interest has one or more tuples (rows), which are entries in the relation. However, as part of gathering and recording information for the relation, duplicates of real entities may be intentionally or unintentionally entered. Hipp, et. al. (2001) explain that data collection is a side-effect of an organization's operation, so databases with poor data quality should not be surprising.

Entity Resolution is the process of finding non-identical duplicates in a relation and merging the duplicates into a single tuple (record), as described by Benjelloun, et. al. (2005). Record linkage is the process of finding related entries in one or more related relations in a database and creating links among them, as described by Malin & Sweeny (2005). Entity Resolution (ER) and Record Linkage (RL) are important steps in data cleansing, which is the removal of inaccuracies in databases, and, as such, is part of populating a data warehouse. Generally, data warehouses are important repositories for organizations reporting on historical data, as shown in Rahm & Do (2000), allowing these organizations to derive accurate aggregate and trending information from the underlying data. Where this information is derived from entities in the organization's concern, it is important for the underlying data to be as accurate as possible. Additionally, because duplicate entries are not allowed in databases, entity resolution can be useful in establishing when a tuple about to be entered will be a copy of one already present. Hence, it is very useful in maintaining the integrity of a traditional database by providing accurate and consistent data.

In this paper, we survey the literature for methodologies proposed or developed for entity resolution and record linkage. We categorize these methodologies broadly into techniques for comparing tuples and applications for cleansing relations. In addition, we introduce a new term, *efficacy*, for discussing the power of these methodologies.

Our work has significance for practitioners as well as researchers. For practitioners, it establishes a basis for selecting the right methodologies in cleansing a data warehouse. It also provides a base and a framework for further research in solving the entity resolution and record linkage problems.

The remainder of the paper is organized as follows: In section 2, we give a formal definition of the problem. In section 3, we describe methodologies from the literature and discuss the efficacy of each. Section 4 discusses

additional considerations in choosing or optimizing the chosen methodologies. Section 5 is the conclusion, which discusses problems and concerns with the methodologies and indicates possible research directions.

AN OVERVIEW OF ENTITY RESOLUTION AND RECORD LINKAGE

Organizations collect information about entities in the real world interesting to them. The entity being described may be physical, such as a person or house, or could be a logical construct, such as a family, a social network or a list of people who like a particular type of music. A relation is the digital representation of this collection. This relation contains a set of entries or tuples, each one pertaining to an entity or set of entities in the real world. Each tuple may refer to a particular entity, but each entity may have one or more tuples describing it. This is described in Hernandez & Stolfo (1998) and shown in Figure 1.

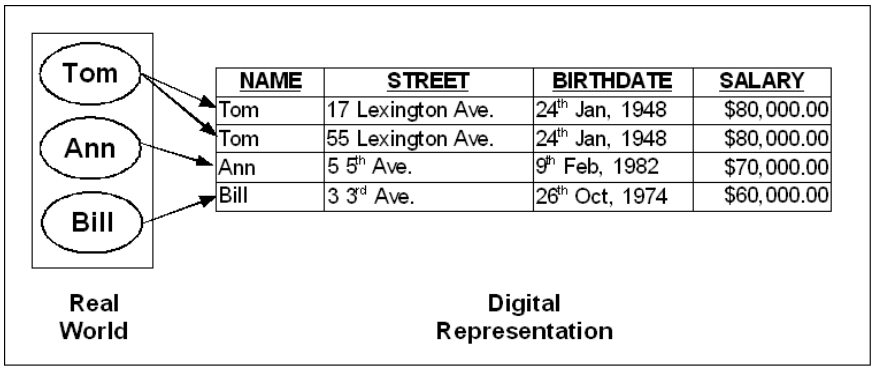

Figure 1: Correspondence Between Entities and Tuples

In Figure 1, three entities (in this case, people) are represented by four tuples, with one duplicate entry for the person named Tom. It is often desirable to remove duplicate entries in a relation, either by merging the duplicates into a single tuple, or by linking each duplicate tuple. However, identifying duplicates may be non-trivial for a number of reasons, as described by Muller & Freytag (2003). For example, a nickname or alias may be incorrectly recorded as a person's proper name, making identification of the duplicate difficult.

Benjelloun et. al. (2005) show that the decision about whether two tuples refer to the same entity can be described by a single function, which we refer to as δ. This function requires at least two inputs – the candidate tuples – and returns a Boolean output (true or false). Entity Resolution is the process of finding tuples in a relation which describe the same entity and merging them. Record Linkage also finds co-referent tuples, but links the tuples rather than merging them. The shared goal of entity resolution is the correct identification of duplicate or co-referent tuples in a relation and mis-identification of none of the duplicates. Once identified, what is done with duplicates distinguishes entity resolution from record linkage.

Efficacy

Because the aim of both entity resolution and record linkage is the accurate identification of co-referent tuples, it is interesting to quantify the effectiveness of a methodology or a solution. In practice, however, accurate identification may not be possible, as is shown in Hernandez & Stolfo (1998). In that study, the authors discovered a number of duplicates in a supposedly cleansed relation and failed to identify a number of other duplicates.

We define the *efficacy* of an entity resolution or record linkage methodology as its success rate, with *theoretical efficacy* referring to its estimated or expected success rate and *real efficacy* as the efficacy of the methodology shown under experimental conditions.

Efficacy is measured as shown in Formula 1, with:

- ***n*** representing the number of real-world entities for which there are tuples in the relation
- ***precision***$_i$ representing the number of tuples correctly identified by the methodology divided by the number returned by the methodology for entity E_i
- ***recall***$_i$ representing the number of tuples correctly identified by the methodology divided by the number of tuples in the relation representing the entity E_i

Efficacy scores range from 0 to 1. Higher efficacy scores describe methodologies more effective at correctly identifying entities in a relation and associating those entries with entities.

$$\frac{\sum_{i=1}^{n} \frac{2 * precision_i * recall_i}{precision_i + recall_i}}{n} \quad (1)$$

Entity Resolution

The purpose of entity resolution is to identify the digital representation of a unique entity of the real world in a single relation and to ensure that only one tuple represents it. Once entities are identified, the next step in entity resolution is the process merging or nesting all tuples which refer to the same entity E_i. This merging may be destructive – i.e. may select one of tuples' values to represent all related tuples – or the merging may retain the values of all related tuples.

Formally, we define the process of entity resolution by the function, δ (the decision to merge any pair of tuples), and by a merge function, μ, which merges two tuples according to specified rules in the relation, R as follows:

$$\forall t_i\, t_j\, (t_i \in R) \wedge (t_j \in R) \wedge \delta(t_i, t_j) \rightarrow R = R - t_i - t_j + \mu(t_i, t_j) \quad (2)$$

Record Linkage

The purpose of record linkage is to explicitly record a link, or relationship, between related tuples where no such link has been made explicit. Here, the real entity being described may be physical, such as a person, a social construct, such as a group of family members or a network of friends, or any other object in the real or logical world which can be described.

Figure 2 shows a graphical representation of record linkage employed across two relations.

Address	City	State	Links
11 Crestline Dr.	SF	CA	
1073 Tennis Ct.	Bklyn	NY	

Links	Address	Zip
	1073 Tennis Court	11233
	11 Crestline Dr.	94131

Figure 2: Record Linkage Across Two Relations

Once entities are identified, record linkage creates links among all related tuples representing an entity, E_i. Where record linkage finds co-referent tuples, i.e. duplicates, in a single relation, we find it to be almost equivalent to entity resolution.

Formally, we define the process of record linkage by the function, δ (the identification of a related pair of tuples) and by a link function, λ, which links two tuples in the relation, R as follows:

$$\forall t_i t_j\, (t_i \in R) \wedge (t_j \in R) \wedge \delta(t_i, t_j) \rightarrow \lambda(t_i, t_j) \quad (3)$$

METHODOLOGIES

Broadly, we divide the techniques for performing entity resolution or record linkage can be divided into: 1) establishing good match criteria between any pair of tuples, and 2) applying that match criteria over one or more relations. This section describes both types of methodologies.

Establishing Match Criteria

Both entity resolution and record linkage have one or more criteria for deciding when two tuples refer to the same entity. Where more than one criterion is used, it must be connected to others with Boolean AND or OR. This decision defines an entity. For example, a decision may be made that all people who share an address and a last name are related to each other, and people who share an address but do not have the same last name are co-habitants (roommates).

Many entity resolution and record linkage decisions may be performed by comparing the values of one or more attributes found in a pair of tuples. A match function compares attributes of two tuples and produces a dichotomous response (*matches* or *does not match*), as described by Benjelloun, et. al. (2005). This match function may involve a user-defined threshold, with this threshold determining the point dividing matching tuples from non-matching ones. Classically, as defined by Fellengi and Sunter and reported by Winkler (2003), this function returns one of three responses (*matches*, *does not match* or *needs more review*).

Much of the literature's focus on matching criteria for entity resolution and record linkage employ text-based matches. This is not surprising given that a great deal of data in an organization's databases is in text form. A brief description of various matching techniques found in the literature follows.

Exact Match. Perhaps the easiest match criteria to establish or implement is an exact match. Where tuples have the same values for a particular attribute, they can be said to refer to the same entity or to be related to each other. For example, if a relation contains two tuples with identical last names, first names and driver's license numbers, it can be said that the two are identical. Certain criteria may seek only partial matches for certain attributes, as described by Cohen & Richman (2002). In performing exact matches, the assumption is that two tuples are free from data entry errors (or have similar entry errors).

Distance Match. One criterion for establishing a match between tuples can be implemented by a deciding on a threshold over one of a group of functions used to describe distance between any two strings, as cited by Cohen & Richman (2002), Bilenko & Mooney (2003) and others. Distance matches must be given a threshold. The implementations are as follows:

1. *String Edit Distance*: the absolute number of inserts, deletes and substitutions required to change one string into the other. Computationally, this absolute number is often established by using a table, such as the one in Figure 3, comparing the word "pine" to the word "payne." In this case, the final edit distance of 3 is shown in the bottom right-hand corner of the table.

		p	i	n	e
	0	1	2	3	
p	1	2	3	4	
a	2	3	4	5	
y	3	4	3	4	
n	4	5	4	3	
e					

Figure 3: Edit Distance Table for "pine" vs. "payne"

2. *Keyboard Distance*: the number of inserts, deletes and substitutions required to change one string into the other, with greater distance between letters incurring a heavier penalty. Here, the number of spaces between letters on a QWERTY keyboard defines distance. This match has high theoretical efficacy when used to correct data entry (transcription) errors.

Soundex: The two strings are converted to one of a class of phonetic representations, and those representations are compared to each other, sometimes with thresholds. Sarawagi & Bhamidipaty (2002) briefly discuss the use of Soundex functions to establish tuple similarity. This criterion has high theoretical efficacy when used with words or names with alternate spellings, such as "Debra" and "Deborah," which would otherwise be difficult to discern by comparing the texts of each string.

Cosine Similarity Matches: This measure allows match criteria independent of word ordering in the two strings being compared. Essentially, the similarity of two sentences can be established by the sum of the pairs of matches between the words in the sentence. However, as shown by Gravano, et. al. (2003), this comparison is not limited to establishing similarity of sentences based on the component words; it can compare any string of characters by its substrings. Like many other matching techniques, Cosine similarity must be used with an established threshold.

TF/IDF: Term Frequency / Inverse Document Frequency is a measure from Speech and Language Processing and discussed in Cohen & Richman (2002), Bilenko & Mooney (2003) and others. The idea is to determine the

frequency of a string in the relation and to favor matches of less common strings, penalizing more common strings. This match requires knowledge or derivation of the frequencies of each of the attribute being considered.
For example, when transcribing an address, it may be common for an "Avenue" to be mis-recorded as a "Street." If so, the matching criteria may choose to ignore the most common words in this field, i.e. "Street," "Avenue," "Lane," etc., instead concentrating on the more important number and name.

Clustering or Feature Extraction: As shown in Malin & Sweeney (2005), Cohen & Richman (2002) and Benjelloun, et. al. (2005), a group of tuples with similar features may be extracted from a relation. This group may be established by many criteria, including co-presence, the existence of two people in the same location at the same time. As such, tuples may belong to more than one cluster, especially for a record linkage methodology. Clusters may also be used to establish the identity of an entity, in which case it is useful for entity resolution.

Secondary Source Matches: A source outside the relation may be used for its domain-specific knowledge. For example, a knowledge base of common first names and their nicknames allows a match between two names, for example, "William" and "Bill," despite generating little or no match from the other match criteria. This may be a particularly difficult example for methods which employ sorting, since tuples for "William" and "Bill" may not be close together for comparison by techniques above.

Secondary sources may be slow and expensive to access. Because of this, this match technique is used to impose a Boolean decision only when the other criteria produced no definite response, i.e. when the decision yields "need more review" (Michalowski, et. al. 2004).
In the literature, Christen & Churches (2006) describe a system which employs geopositioning data to determine correct addresses, despite flawed data. Michalowski, et. al. (2004) describe a dynamic system built on an ontology of classes in an object oriented system This system has a mechanism for automatically connecting to secondary sources and gleaning external data when needed.

Applications over Relations

The above match criteria may be applied to compare the attributes of two or more tuples to determine whether they are co-referent or otherwise related. The manner in which the techniques are applied to the target relations may vary producing different running times and coverage of the constituent tuples. Once similarity techniques above have been established, the manner in which those techniques are applied must be chosen. A number of applications are described below and are summarized in Table 1.

Brute Force Application: Using this application, each tuple in a relation is compared to every other tuple. When a tuple matches, it is changed (for example, if it is determined to form an entity and is merged), this changed tuple must again be compared to every other tuple. This application is simple, to implement and completely covers the relation, but it requires the greatest running time – $O(n^2)$ – of those discussed here. The formula for brute force methods is the same as Formula 1, exhaustively applied against all pairs of tuples in the relation.

Technique	Area of Complete Coverage	Time Complexity	Notes
Brute Force	Entire relation	$O(n^2)$	--
Canopy	Window	O(n lg n)	Windows overlap, so coverage is essentially 2(w)
Bucketing	Entire relation	O(n)	Requires clustering technique
Hierarchical	Entire relation	O(n)	Requires data to be in a hierarchy
Data Mining	Entire relation	O(n)	Item threshold may have to be set low
Mutual Decision	Entire relation?	?	Requires prior associations among entities

Table 1: Applications over Relations, Showing Coverage and Time Complexity

Canopy / Sliding Window Applications: With this application, the tuples in the relation are sorted along some criterion, for example, by last name, and a fixed window size, w, is chosen. All pairs of the first w tuples are checked against each other along the established match criteria, and matching tuples are linked or merged. When all the tuples within the window are checked against each other, the next w tuples are considered. The window is moved progressively down the list of tuples until the list is exhausted. An example of a window scan is shown in Figure 4. During scan "w_n," all six possible pairings of the tuples, "Christopher" to "Elizabeth," are compared to each other.

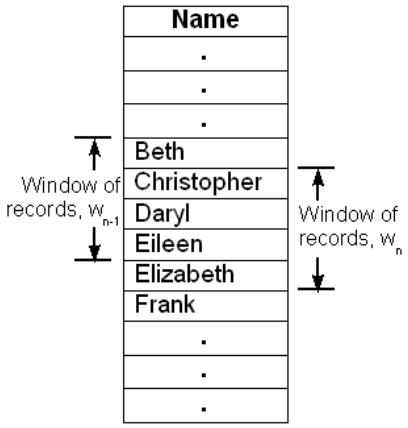

Figure 4: Canopy Window Scan

A successful application of the Sliding Window application on real-world data is reported in Hernandez & Stolfo (1998). Because the number of comparisons performed in each window is constant, determined only by the size of the window, and the number of windows is linear with respect to the number of tuples, this application is dominated by the need to sort the list as the first step. It therefore has a running time of O(n lg n).

Bucketing / Feature Grouping Applications: Bucketing or feature grouping applications do not compare tuples pair wise against each other. Instead, slots are created for each relationship or entity in the real world, and a function is applied to each tuple to determine which slot matches. Any tuples in the same slot are related; therefore, they should be linked (for record linkage) or merged (for entity resolution). A good example of this approach is in

Benjelloun, et. al. (2005). These applications have linear O(n) running time, because the tuples are scanned once for the purpose of bucketing and once more, at most, to perform merges.

The most efficient applications for ranging over a relation or set of relations must touch each tuple in all participating relations, preferably once. Bucketing applications, therefore, are the most efficient since they exhibit linear running time. However, we believe that feature grouping applications are not always possible. If they were, it would be possible to assign a unique ID to each entity in a relation, making the problem moot.

Data Mining / Data Quality Mining Applications: Data mining is the process of finding patterns in relations which are not explicitly obvious. These applications may be used to find patterns of association or co-occurrences within the relation as a whole. These patterns may be applied to the relation to generate a set of tuples which share an interestingness factor. Each of those tuples may be considered for relatedness or for merging in entity resolution and record linkage. One of the more efficient Data Mining algorithms is described by Han, et. al. (1999), with linear time complexity.

An application of Data Mining is described by Hipp, et. al. (2001). In the work, the authors suggest the possibility of mining the relation to discover the quality of the underlying data. The mining application would also suggest places where similar, perhaps near-duplicate, data exists. While the authors discuss their experiments on proprietary data, they fail to mention what thresholds were used for establishing similarity of tuples. Indeed, some relations may require counts as low as 2 to be considered.

Hierarchical Applications: Ananthakrishna, et. al. (2002) describes a general, error-tolerant application for handling hierarchal data such as addresses. When the data is arranged in a hierarchy, the *top* (or container) level may be cleansed based on the presence of duplicates beyond some significant threshold at the level immediately contained. Addresses are hierarchical because they specify a number of "containers" and "sub-containers." For example, a city contains many streets, and a street contains many houses (each with unique numbers). Using domain knowledge about the hierarchy, this application can detect duplicates by reasoning from knowledge of one level to the next. This application seems to be directly inspired by research from the data mining community and, as such, exhibits similar running time.

City	State
New York	NY
New York	N.Y.
New York	New York
San Francisco	CA
SF	CA

Figure 5: Cities and States Arranged in a Hierarchy

Figure 5 shows city and state data arranged in a hierarchy. Because the city named *New York* occurs in states *New York, N.Y.* and *NY*, the application may correctly detect the states containing the duplicate cities as one state with three possible spellings or representations. In their implementation, Ananthakrishna, et. al. (2002) employ TF/IDF techniques to determine when to discount the significance of certain text-based matches. The authors use additional optimizations, such as low thresholds for generating potential match candidates at a certain level in the hierarchy. However, they fail to demonstrate the usefulness of their application outside strongly hierarchal address data, as is found in American and Western Europe.

Mutual Decision-Making Applications: In certain circumstances, it is easier to make a group decision about the relatedness of a set of tuples. This is clearly seen in Bhattacharya & Getoor (2006), an implementation which uses known past relationships to reason about future ones. In this work, the authors favorably discuss their system for establishing identities of authors of articles on Citeseer. In addition to the mutual decision application, their system contains a number of optimizations, and the degree to which each has contributed to their results remains unclear. A demonstration or proof of running time is missing from their work.

ADDITIONAL CONSIDERATIONS

Achieving Completeness using Multi-Pass Applications

The applications above may be insufficient for solving all problems related to entity resolution or record linkage. For example, the Canopy application is cited as being fast but has low theoretical efficacy for certain datasets when the matching criterion is poorly chosen.

To overcome this, Hernandez & Stolfo (1998) chose a multi-pass implementation of the Canopy application. Specifically, the relation was sorted along one attribute, and a set of match criteria was established, and cleansed according to that criteria. Subsequently, the relation was re-sorted and an unrelated second set of match criteria was employed to mitigate any deficiencies associated with the first. While Hernandez & Stolfo report satisfaction after two passes, their experiments tested up to three successive passes, each with increases in running time, number of false positives (analogous to low efficacy) and percentage of duplicates found (analogous to high efficacy).

Multi-pass criteria are not limited to Canopy applications. While some applications, such as the Hierarchical application, inherently perform multiple passes across the relation, any of the other applications may be extended to allow multiple passes along different criteria. We believe this will have an overall effect of increasing the efficacy of most techniques.

Incremental Methods

Once built, a cleansed relation may be important to maintain by allowing new tuples to be added with a minimum of re-cleansing. This is discussed in Muller & Freytag (2003). Benjelloun, et. al. (2005) provides a theoretical basis for entity resolution, fitting closely to their non-incremental application, in which the new tuples are compared to already cleansed ones. Likewise, any tuples created as a result of a merge must be compared to others which have been cleansed. New tuples, therefore, are simply treated as uncleansed ones.

Hernandez & Stolfo (1998) implement a provision for an incremental method in a record linkage system. In it, the linked relation has *prime representatives* or *cluster centroids*: tuples nominated to represent the best examples of real-world entities. These prime representatives are chosen according to some criterion – for most up-to-date tuple or for syntactic completeness, for example. For speed, incoming tuples are compared only against these prime representatives for record linkage.

Multiple Relations

As discussed in Rahm & Do (2000) as well as in Doan & Halevy (2005), record linkage may operate on multiple relations, Entity resolution is poorly suited for this task, so the relations in question should be merged prior to attempting this. Tools such as Potter's Wheel, described by Raman & Hellerstein (2001) may aid the practitioner in this task.

DISCUSSION AND RESEARCH DIRECTIONS

Non-Text Comparisons

Organizations are increasingly collecting data in non-text format. Images from digital cameras used in security settings are being retained and stored as part of relations. Digital images may also record fingerprints and other biometric data. As organizations begin to place more importance on data in non-text format, we believe that non-text comparison techniques will become essential in forming data warehouses which use this data. Some of the criteria described in section 3.1 are unsuited for non-text data, so new techniques may be created as a result.

Key and Integrity Enforcement

In a relation, an entity is ideally represented by one tuple, as described by Elmasri & Navathe (1994, p. 394). One method of enforcing this is through the use of primary keys. Entity resolution, a methodology in which tuples representing the same entity are merged, can be seen as a tool in re-establishing primary keys or in enforcing the consistency of established primary keys in a relation in which an entity has acquired two or more keys.

Likewise, record linkage may be seen as a tool in enforcing referential integrity between two or more relations. Of course, record linkage may be used on a single relation, but when record linkage is applied to a single relation for the purpose of determining which tuples are co-referent, it performs the same function as entity resolution.

If entity resolution is a tool used to reclaim the primary keys of a corrupted relation and record linkage a tool to re-establish referential integrity, it is possible to view the twin problems as reclaiming the "true key" of an entity or tuple in the presence of error. Where possible in our future work, we plan to explore the efficacy of algorithms derived from this idea with respect to different techniques and applications for the relation, both in terms of the initial formation of a data warehouse and the maintenance of it. For example, a "variable canopy" application may be applied to all tuples in a feature group, resulting in linear running time with high efficacy. It remains to be seen whether this is possible.

Testing

Christen (2005) has established a dataset for testing the efficacy of techniques and applications. This dataset, while customized for the Australian continent, may contains certain universals, including a certain fuzziness about which names and addresses belong to which entities in the real world. Others, such as Bhattacharya & Getoor (2004), Bhattacharya & Getoor (2006) and Sarawagi & Bhamidipaty (2002) use citations from articles found on Citeseer as their test bed.

Our coverage of the literature has found no well-established dataset used for testing the efficacy of an application. In fact, outside Winkler's work with the U.S. Census, little research has been conducted on the types of entities represented in the relations which need to be cleansed. We believe this to be an oversight. As Bhattacharya & Getoor (2006) shows, more knowledge of the domain can result in better choices for matching criteria and optimization of the application used for cleansing.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their valuable comments. The second author's research is supported by a PSC-CUNY research award.

REFERENCES

Ananthakrishna, R., Chaudhuri, S. & Ganti, V. (2002). Eliminating Fuzzy Duplicates in Data Warehouses. *In the Proceedings of the 28th International Conference on Very Large Databases*.

Benjelloun, O., Garcia-Molina, H., Jonas, J., Su, Q. &Widom, J. (2005). Swoosh: A Generic Approach to Entity Resolution. *Technical Report, Stanford University*.

Bhattacharya, I. & Getoor, L. (2006). A Latent Dirichlet Allocation Model for Entity Resolution. *6th SIAM Conference on Data Mining, Bethesda, MD.*

Bhattacharya I. & Getoor L. (2004). Deduplication and Group Detection Using Links. *KDD Workshop on Link Analysis and Group Detection, Aug. 2004, Seattle*.

Bilenko, M. & Raymond J. Mooney, R. J. (2003). Adaptive Duplicate Detection Using Learnable String Similarity Measures. *In the Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Bright, M. W., Hurson, A. R. & Pakzad, S. H. (1992). A Taxonomy and Current Issues in Multidatabase Systems. *Computer, v.25 n.3, p.50-60, March 1992*.

Christen, P. (2005). Probabilistic Data Generation for Deduplication and Data Linkage. *In the Proceedings of the Sixth International Conference on Intelligent Data Engineering and Automated Learning*.

Christen, P. & Churches, T. (2006). A Probabilistic Deduplication, Record Linkage and Geocoding System. *Advances in Data Mining: Theory, Methodology, Techniques, and Applications. State-of-the-Art Lecture Notes in Artificial Intelligence, Volume 3755, Springer-Verlag, 2006*.

Cohen, W. W. & Richman, J. (2002). Learning to Match and Cluster Large High-Dimensional Data Sets for Data Integration. *Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

Doan, A. and Alon Y. Halevy, A. Y. (2005). Semantic Integration Research in the Database Community: A Brief Survey. *AI Magazine, Special Issue on Semantic Integration, 2005*.

Elmasri, R. & Navathe, S. B. (1994). *Fundamentals of Database Systems*. Addison-Wesley Publishing Company, Menlo Park, CA.

Gravano, L., Ipeirotis, P. G., Koudas, N. & Srivastava, D. (2003). Text joins for data cleansing and integration in an RDBMS. *In the Proceedings of the 19th IEEE International Conference on Data Engineering, 2003*.

Han, J., Pei, J. & Yin, Y. (1999). Mining frequent patterns without candidate generation. *Technical Report TR-99-12, Computing Science Technical Report, Simon Fraser University, October 1999*.

Hernandez, M. A. & Stolfo, S. J. (1998). Real-world Data is Dirty: Data Cleansing and the Merge/Purge Problem. *In Journal of Data Mining and Knowledge Discovery*.

Hipp, J., Guntzer, U. & Grimmer, U (2001). Data Quality Mining: Making a Virtue of Necessity. *In the Proceedings of the 6th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*.

Jurafsky, D. & Martin, J. H. (2000). *Speech and Language Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ. 2000.

Malin, B. & Sweeney, L. (2005). ENRES: A Semantic Framework for Entity Resolution Modelling. Institute for Software Research International Technical Report (Carnegie Mellon Publication No. CMU-ISRI-05-134).

Michalowski, M., Thakkar, S. & Knoblock, C. A. (2004). Exploiting Secondary Sources for Automatic Object Consolidation. *In the Proceedings of the 2004 VLDB Workshop on Information Integration on the Web*.

Muller, H. & Freytag, J.-C (2003). Problems, Methods and Challenges in Comprehensive Data Cleansing. Technical Report (Humboldt-Universitt zu Berlin, Institut fr Informatik Publication No. HUB-IB-164).

Neely, M. P. (1998). Data Quality Tools for Data Warehousing - A Small Sample Survey: Using Information in Government Program. *MIT Conference on Information Quality*.

Rahm, E. & Do, H. H. (2000). Data Cleaning: Problems and Current Approaches. *IEEE Bulletin of the Technical Committee on Data Engineering, Vol 23 No. 4. 2000*.

Raman, V. & Hellerstein, J. (2001). Potter's Wheel: An Interactive Data Cleaning System. *In the Proceedings of the 27th VLDB Conference, 2001*.

Sarawagi, S. & Bhamidipaty, A. (2002). Interactive Deduplication using Active Learning. *In The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD-2002*.

Winkler, W. E. (2003). Data Cleaning Methods. *In the Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*.