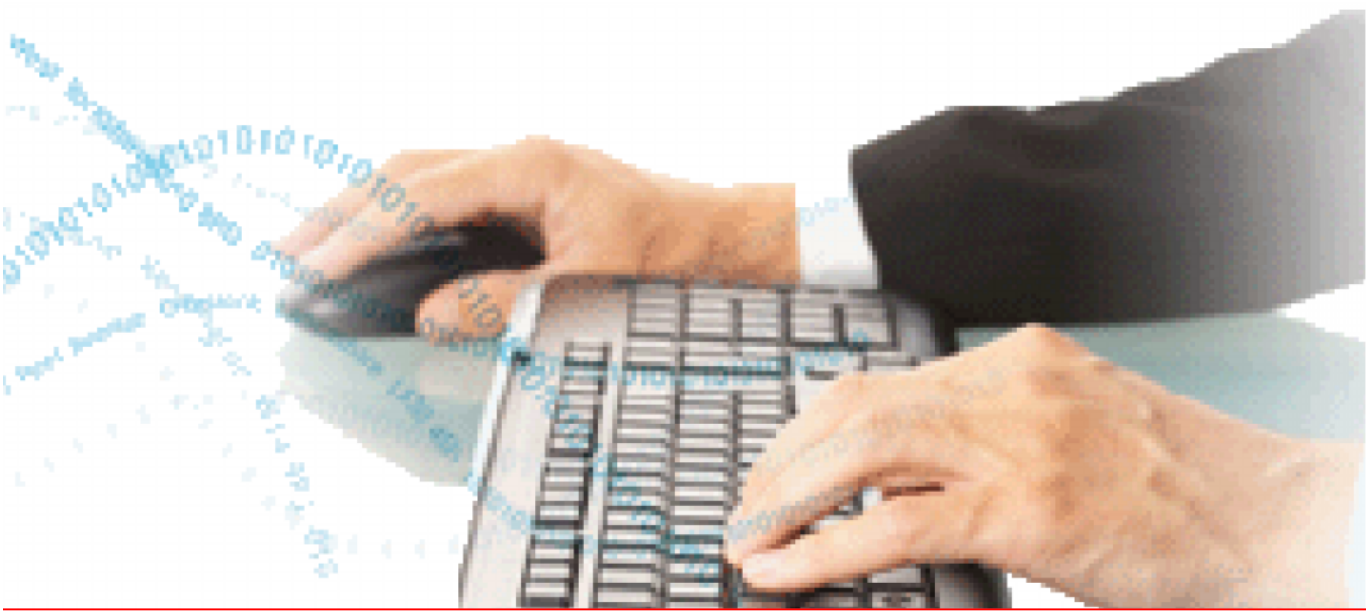
Data Cleaning and Record Linkage Research in Asia

Xiangmin Emily Zhou, Daniel McMichael and John Taylor
Monday, 20 July 2009

The Department of Defence
Wing Commander Bill Compton



Australian Government
Department of Defence

Enquiries should be addressed to:

Dr. John Taylor,
Leader, Computational and Simulation Sciences, CSIRO
Tel. +61 2 6216 7077
Fax +61 2 6216 7111
Email: John.A.Taylor@csiro.au
Web: www.csiro.au

Distribution list

Bill Crompton (DOD)	Three copies
John Taylor (CSIRO)	One copy
Daniel McMichael (CSIRO)	One copy
Tracey Papandreou (CSIRO)	One copy
Xiangmin Zhou (CSIRO)	One copy

Copyright and Disclaimer

Important Disclaimer

Contents

List of Figures

1. THE AIMS AND APPROACH OF THIS REPORT

The Department of Defence requested a survey of publicly available research on new enabling topics in data mining within the Asia region. The initial pathfinder surveys in four promising areas showed that a very small number of Asian researchers are making significant contributions. The vast bulk of published Asian research in this area is derivative. It was decided to proceed to detailed surveys in two areas: *data quality* and *ontology-based methods*. This report concerns the first of these.

To provide reports of reasonable coherence, in each of the two areas, we have surveyed the key contributions that have been made from researchers across the world. The Asian contributions have been described in relation to this larger international context. We have also added sections on significant developments in hardware and some prognostications on barriers to progress and likely future developments.

2. INTRODUCTION

2.1 The significance of data quality in the modern economy

In the area of computer science and systems technology the term *data quality* (DQ) means the degree to which the data stored on computer systems accurately reflect the events they are intended to describe. Discrepancies between our data and the real world arise from many sources including (i) spelling errors, (ii) missing fields, (iii) obsolescence (e.g. after name or address changes), (iv) systematic erroneous substitutions, (iv) data entry errors, (v) migration errors arising from moving the data from one system to another, (vi) malicious errors and (vii) multiple records referring to the same underlying entity which should never have been separated. In this report, we focus on the latter issue because it is one of the most important and technically difficult.

Data quality is becoming a more and more serious issue because of the following trends: (i) the steady increase in the rate at which data is gathered via the internet and from sophisticated sensors, (ii) increasing requirements to aggregate data sets from multiple sources, (iii) increasing amounts of malicious and deceptive data (iv) increasing use of centralised data processing systems and (v) increasing analysis of data to extract greater value from it. The combined effect of these trends is to generate more lower quality data from which more information is expected. In health care for example, studies and reports spanning the last ten to fifteen years indicate that while the quantity data reported by provider organizations and the number of databases/data holdings created to manage this data have progressively increased over time, the quality of the data reported, collected and held in these databases, and therefore the usefulness of the data and information for effective evidence-based decision-making continues to be of significant concern [1].

We would expect from good quality data that it is accurate, consistent and there exits a transaction audit trail back through any transformations that have been made to it to the time,

place and detail of its capture. The availability of good quality data facilitates efficient supply chains, financial transactions, management of organisations, operation of government and other productivity improvements of all kinds.

2.2 The cost of low data quality

Real-world data is often dirty, and containing inconsistencies, conflicts and errors. A recent survey [2] reveals that enterprises typically expect data error rates of approximately 1%–5%. The consequences of dirty data may be severe. According to the recent report from the information management magazine [3], two 20-year-old "calculation errors" for pension withholding left Los Angeles County with $1.2 billion in unforeseen liabilities and will force the county to spend an additional $25 million a year to make up for insufficient contributions to the fund. Wrong price data in retail databases costs American consumers $2.5 billion in annual overcharges. Quality data is vital in any data-driven enterprise. If information is the currency of the new economy, then data is a critical raw material needed for success. Just as a refinery takes crude oil and transforms it into numerous petroleum products, companies use data to generate a multiplicity of information assets. These assets form the basis of the strategic plans and actions that determine a firm's success. Consequently, poor quality data can have a negative impact on the health of a company. If not identified and corrected early on, defective data can contaminate all downstream systems and information assets.

The problem with data is that its quality quickly degenerates over time. Experts say 2% of records in a customer file become obsolete in a month because customers die, divorce, marry and move [4]. In addition, data-entry errors, systems migrations and changes to source systems, among other things, generate bucket loads of errors [4]. As well, as organizations fragment into different divisions and units, interpretations of data elements mutate to meet local business needs. A data element that one individual finds valuable may be nonsense to an individual in a different group.

Poor quality customer data have caused serious effect on economy in many companies. The Data Warehousing Institute (TDWI) estimates that poor quality customer data costs U.S. businesses a staggering $611 billion a year in postage, printing and staff overhead [5]. Frighteningly, the real cost of poor quality data is much higher. Organizations can frustrate and alienate loyal customers by incorrectly addressing letters or failing to recognize them when they call, or visit a store or Web site [5]. Once a company loses its loyal customers, it loses its base of sales and referrals, as well as future revenue potential.

Given the business impact of poor quality data, it is bewildering to see the casual way in which most companies manage this critical resource. Most companies do not fund programs designed to build quality into their data in a proactive, systematic and sustained manner. According to TDWI's Data Quality Survey [6], almost half of all firms have no plan for managing data quality.

Part of the problem is that most organizations overestimate the quality of their data and underestimate the impact errors and inconsistencies can have on their bottom line. On one hand, almost half of the companies who responded to our survey believe the quality of their data is "excellent" or "good." Yet more than one-third of the respondent companies think the quality of their data is "worse than the organization thinks." Although some firms understand the importance of high-quality data, most are oblivious to the true business impact of defective or substandard data. Thanks to a raft of new information-intensive strategic business initiatives,

executives are beginning to wake up to the real cost of poor quality data. Many have bankrolled high-profile IT projects in recent years -- data warehousing, CRM and e-business projects -- that have failed or been delayed due to unanticipated data-quality problems. For example, in 1996, FleetBoston Financial Corp. (then Fleet Bank) in New England undertook a much publicized $38 million CRM project to pull together customer information from 66 source systems. Within three years, the project was drastically downsized and the lead sponsors and technical staff were let go. A major reason the project came unravelled was the team's failure to anticipate how difficult and time consuming it would be to understand, reconcile and integrate data from 66 different systems. According to TDWI's Industry Study 2000 survey [7], the top two technical challenges firms face when implementing CRM solutions are "managing data quality and consistency" (46%) and "reconciling customer records" (40%). Considering that 41% of CRM projects were "experiencing difficulties" or "a potential flop," according to the same study, it is clear that the impact of poor data quality in CRM is far reaching.

Data warehousing, CRM and e-business projects often expose poor quality data because they require companies to extract and integrate data from multiple operational systems. Data that is sufficient to run payroll, shipping or accounts receivable is often peppered with errors, missing values and integrity problems that do not show up until someone tries to summarize or aggregate the data. Also, since operating groups often use different rules to define and calculate identical elements, reconciling data from diverse systems can be a huge, and sometimes insurmountable, obstacle. Sometimes the direct intervention of the CEO is the only way to resolve conflicting business practices, or political and cultural differences.

Every firm can uncover a host of costs and missed opportunities caused by inaccurate or incomplete data. Consider the following US experiences [5]:

- A telecommunications firm lost $8 million a month because data-entry errors incorrectly coded accounts, preventing bills from being sent out.
- An insurance company lost hundreds of thousands of dollars annually in mailing costs due to duplicate customer records.
- An information services firm lost $500,000 annually and alienated customers because it repeatedly recalled reports sent to subscribers due to inaccurate data.
- A large bank discovered that 62% of its home-equity loans were being calculated incorrectly, with the principal getting larger each month.
- A health insurance company in the Midwest delayed a decision support system for two years because the quality of its data was "suspect."
- A global chemical company discovered it was losing millions of dollars in volume discounts in procuring supplies because it could not correctly identify and reconcile suppliers on a global basis.
- A regional bank was unable to calculate customer and product profitability due to missing and inaccurate cost data.
- Two 20-year-old "calculation errors" for pension withholding left Los Angeles County with $1.2 billion in unforeseen liabilities and will force the county to spend an additional $25 million a year to make up for insufficient contributions to the fund (Los Angeles Times, April 8, 1998).

- Wrong price data in retail databases costs American consumers $2.5 billion in annual overcharges (Information Week, September 14, 1992).
- A $2 billion company discovered it was not invoicing four percent of its orders, leaving $80 million in uncollected revenue.
- In August 1997, Hudson Foods lost its largest customer, Burger King, due to E. coli bacteria contamination that caused several illnesses. The plant had two problematic practices: poor record-keeping and the mixing of one day's leftover hamburger into the next day's production. The information quality problem of not knowing which batches were mixed caused the largest meat recall in U.S. history: 25 million pounds. Without its largest customer, Hudson Foods was not able to be profitable, having to sell the plant. The rest of Hudson Foods was subsequently acquired by Tyson Foods (The Tennessean, August 24, 1997).

The information resource has two distinctions that demand understanding and attention at all levels in the organization:

1. Information is intangible and non-consumable. As such it has not been taken as a serious object for management. With every other resource, managers are held accountable. Few organizations have information quality accountability written into a manager's job description.
2. Information is the resource required to manage every other enterprise resource. Data is the electronic representation of objects and events the enterprise must know about and manage. Financial management is not managing currency; it is managing the information about the enterprise's financial assets.

2.3 How does data get dirty?

The published literature on data quality can be divided into problems dealing with centralized systems (storing data in one logical database but operated in a multi user environment), and problems where the data is distributed at multiple sites either generated locally or imported (inserted) from multiple independent sources to the integrated structure, such as a data warehouse, a BPM engine or a scientific grid. In centralized systems, the predominant data quality problems are: duplicate removals/identification[50], errors due to concurrent data access, missing/incomplete data for attributes with mandatory occurrence constraints, inaccurate data entry (errors in spelling), violations of integrity constraints [9, 10], and infringement of other conceptual constraints such as frequency occurrence, subset constraints and subtype constraints. For systems operating on integrated data, all problems with maintenance of data quality listed above are present, but in addition errors arise due to varying standards, schemas and naming conventions and varying interpretation of them, together with increased opportunity for creation of duplicates.

There are numerous sources of poor quality data. The most frequent data quality problems are caused by poor data-entry processes and user interfaces and human error in data entry. As indicated by survey respondents, the most common source of data entry errors include misspellings, transposition of numerals, incorrect or missing codes, data placed in the wrong fields and unrecognizable names, nicknames, abbreviations or acronyms etc. Customer entered data is especially error-prone and, with the rise of e-business, this source is becoming very

significant. Eckerson provides a collection of scenarios that cause poor data quality problems are presented [8]. We summarize them as follows:

- ***Lack of validation routines.*** Many data-entry errors can be prevented through the use of validation routines that check data as it is entered into Web, client/server or terminal-host systems. Respondents to the TDWI survey mentioned a "lack of adequate validation" as a source of data defects, noting this grievance in the "Other" category.

- ***Valid, but not correct.*** But even validation routines cannot catch typos where the data represents a valid value. Although a person may mistype a telephone number, the number recorded is still valid -- it is just not the right one. The same holds true for social security numbers, vehicle identification numbers, part numbers and last names. Database integrity rules can catch some of these errors, but firms need to create complex business rules to catch the rest.

- ***Mismatched syntax, formats and structures.*** Data-entry errors are compounded when organizations try to integrate data from multiple systems. For example, corresponding fields in each system may use different syntax (first-middle-last name vs. last-first-middle name), data formats (6 byte date field vs. 4 byte date field), or code structures (male-female vs. m-f vs. 1-2). In these cases, either a data cleansing or ETL tool needs to map these differences to a standard format before serious data cleanup can begin.

- ***Unexpected changes in source systems.*** Perhaps a more pernicious problem is structural changes that occur in source systems. Sometimes these changes are deliberate, such as when an administrator adds a new field or code value and then neglects to notify the managers of connecting systems about the changes. In other cases, front-line people reuse existing fields to capture new types of information that were not anticipated by the application designers.

- ***Spiderweb of interfaces.*** Because of the complexity of systems architectures today, changes to source systems are easily and quickly replicated to many other systems, both internal and external. Most systems are connected through a spiderweb of interfaces to other systems. Updating these interfaces is time-consuming and expensive, and many changes slip through the cracks and "infect" other systems. Thus, changes in source systems can wreak havoc on downstream systems if adequate change management processes are not in place.

- ***Lack of referential integrity checks.*** It is also true that target systems do not adequately check the integrity of the data they load. For example, data warehouse administrators often turn off referential integrity when loading the data warehouse for performance reasons. If source administrators change or update tables, this can create integrity problems that are not detected.

- ***Poor system design.*** Source or target systems that are poorly designed can create data errors. As companies rush to deploy new systems, developers often skirt fundamental design and modelling principles, which leads to data integrity problems down the road.

- ***Data conversion errors.*** In the same vein, data migration or conversion projects can generate defects, as well as ETL tools that pull data from one system and load it into another. Although systems integrators may convert databases, they often fail to migrate business processes that govern the use of data. In addition, programmers may not take the time to understand source or target data models, and may therefore write code that

introduces errors. One change in a data migration program or system interface can generate errors in tens of thousands of records.

- ***The fragmentation of definitions and rules.*** A much bigger problem comes from the fragmentation of organizations into a multitude of departments, divisions and operating groups, each with its own business processes supported by distinct data management systems. Slowly and inexorably, each group begins to use slightly different definitions for common data entities -- such as "customer" or "supplier" -- and apply different rules for calculating values, such as "net sales" and "gross profits." Add mergers, acquisitions and global expansion into countries with different languages and customs, and you have a recipe for a data-quality nightmare.

  The problems that occur in this scenario have less to do with accuracy, completeness, validity or consistency, than with interpretation and protecting one's "turf." That is, people or groups often have vested interests in preserving data in a certain way even though it is inconsistent with the way the rest of the company defines data. For example, many global companies squabble over a standard for currency conversions. Each division in a different part of the world wants the best conversion rate possible. And even when a standard is established, many groups will skirt the spirit of the standard by converting their currencies at the most opportune times, such as when a sale was posted vs. when the money was received. This type of maneuvering wreaks havoc on a data warehouse that tries to accurately measure values over time.

- ***Slowly changing dimensions.*** Similarly, slowly changing dimensions can result in data-quality issues depending on the expectations of the user viewing the data. For example, an analyst at a chemical company wants to calculate the total value of goods purchased from Dow Chemical for the past year. But Dow recently merged with Union Carbide, which the chemical company also purchases materials from. In this situation, the data warehousing manager needs to decide whether to roll up purchases made to Dow and Union Carbide separately, combine the purchases from both firms throughout the entire database, or combine them only after the date the two companies merged. Whatever an approach the manager takes, it will work for some business analysts and alienate others.

  In these cases, data quality is a subjective issue. Users' perception of data quality is often coloured by the range of available data resources they can access. Where there is “competition” -- another data warehouse or data mart that covers the same subject area “knowledge workers tend to be pickier about data quality”, said Michael Masciandaro, director of decision support at Rohm & Haas.

Data quality problems are expensive and pervasive. They cost hundreds of billion dollars each year. Resolving them is often the biggest effort in a data mining study. It typically requires a very large investment of time and energy **often 80% to 90% of a data analysis project is spent in making the data reliable enough** that the results can be trusted.

To repair unclean data the aim to find a repair that is close to the original and satisfies the required constraints. This is the data cleaning approach that US national statistical agencies, among others, have been practicing for decades [11, 12]. Manual editing is unrealistic when the database is large. Indeed, manually cleaning a set of census data can take dozens of clerks many months [12].

3. COLUMN SEGMENTATION

Frequently, legacy data is stored in large text files. To reinsert such data into databases its structure needs to be reconstructed. Usually these files contain the data relating to a single table. The data is a string (sequence) of characters. Record and column boundaries are marked using short sequences of characters, typically <CR><LF> between records and a comma or tab between fields. The structural reconstruction problem is one of identifying the rows and separating out the columns. Of course there are many ways that this simple scheme can go awry. For example, record and field separating characters can occur as part of field values, some fields may not have been serialised properly for storage in text files resulting in spurious separator characters within binary dumps of field values. Empty records may be marked in the text file as spaces, rather than as nulls.

Current techniques for automatically segmenting input strings into structured records either apply manually programmed rules or employ supervised learning techniques. Rule-based approaches require a domain expert to design a number of rules and maintain them over time. This approach does not scale as each new application may require designing, crafting, deploying, and maintaining a new set of rules. Supervised approaches alleviate this problem by automatically learning segmentation models from training data consisting of input strings and the associated correctly segmented tuples[25, 70]. However, it is usually hard to obtain training data, that is comprehensive enough to illustrate all features of the data to be segmented.

4. RECORD MATCHING AND LINKAGE

The goal of record matching is to identify records in the same or different databases that refer to the same real-world entity, even if the records are not identical. Reviews of this topic have been provided in [13, 14, 15]. Existing approaches for this task mainly focus on the use of similarity metrics (functions) on string pairs to measure how likely strings are to represent the same underlying entity. The arguments of similarity metrics are a pair of strings and they take values between 0 and 1.The higher the value the greater the similarity, with the 1 corresponding to equality. Commonly used similarity metrics include edit distance, cosine similarity, Jaccard similarity and the generalized edit distance [16]. Having code for evaluating the metrics given two input strings is less useful that one might first imagine. Normally the problem is one of *retrieval* where one seeks to retrieve all the records that have one or more fields that are similar to each other. This idea can be formalized in the form of a search criterion (query): *find all record pairs such that the weighted sum of similarity metrics of their fields is greater than a threshold*. Referring back to the database world, the results of such queries are termed *approximate* joins. Classically, a *join* is an operation on a database whereby one or more tables are linked, so that one or more records in a table are linked to others in the same or other tables because they satisfy some exact match criterion. In approximate joint, the joining criterion does not require an exact match.

The term *approximate join* has a similar meaning to the terms:

- **Deduplication** uses approximate joins to the same table to eliminate near duplicated records corresponding to the same underlying entity (Section 4.1);
- **Record linkage** uses approximate joins between tables;

- **Coreference resolution** is concerned with clustering references within and between documents; and
- **Entity resolution**, a generalized term for the task of clustering records that refer to the same underlying entity together.

Many approximate join methods have been proposed to improve the efficiency of record matching, such as the BigMatch [17], SSjoin [18], FastSS [70] and NGPP [71].

4.1 Deduplication

When information from multiple sources of data is integrated, it invariably leads to erroneous duplication of data when these sources store overlapping information. For example, both ACM and DBLP store information about publications, authors and conferences. Owing to data entry errors, varying conventions and a variety of other reasons, the same data may be represented in multiple ways an author's name may appear as "Jeffrey Ullman" or "J. D. Ullman". A similar phenomenon occurs in enterprise data warehouses that integrate data from different departments such as sales and billing that sometimes store overlapping information about customers. Hence, a significant amount of resources are spent today on the task of detecting and eliminating duplicates. This problem of detecting and eliminating multiple distinct tuples representing the same real world entity is traditionally called the deduplication problem. The problem is challenging since the same tuple may be represented in different ways thus rendering duplicate elimination by using "select distinct" queries inadequate.

The task of deduplication is to translate this pair-wise information into a partition of the input relation. While duplication in the real world is an equivalence relationship, the relationship induced by the similarity function is not necessarily an equivalence relation. The input to the deduplication problem is a relation (or view) $R(T, N_1, \ldots, N_k)$ with a text field T and numeric fields N_i. The output is a partition of the records in R which we capture through a GroupID column that is added as a result of deduplication. We refer to each equivalence class in the partition as a group of tuples.

Many approaches have been proposed for deduplication [19, 20, 21, 22, 23, 50]. One of the most common sources of mismatches in database entries is the typographical variations of string data. Therefore, duplicate detection typically relies on string comparison techniques to deal with typographical variations. Therefore, techniques that have been applied for matching fields with string data can be also used in the duplicate record detection context.

4.2 Similarity measures

4.2.1 Edit based measures

Edit distance is a commonly used string matching that counts the minimal number of updates (including insertion, deletion and substitution operations) to transform one string into another [29]. Given two strings, s, t, *edit(s, t)* is the Minimum cost sequence of operations to transform s to t. For example: *edit(Error, Eror)* = 1, *edit(great, grate)* = 2. Generally, the edit distance between two strings is computed using Folklore dynamic programming algorithm. Edit distance based measure provides a distance measure for two strings that reflects the number of errors

that may have been made by an operator intending to enter one string but actually entering the other. However, evaluating the edit distance is in the worst case quadratic in the length of the strings.

Edit distance is widely used in approximate string matching to handle the data inconsistency existing in matching [30, 34, 47, 48]. In [47], the similarity between two strings is measured with the constraint of edit distance for the approximate queries. To improve the performance of approximate queries on string collections, the frequencies of variable-length grams in the strings is first analysed, and a set of grams is selected to form a gram dictionary D. Based on the string gram dictionary, each string is first transformed into a set of variable-length grams based on the preselected grams. When using a gram dictionary D to generate a set of variable-length grams for a string s, we still use a window to slide over s, but the window size varies, depending on the string s and the grams in D. Then the relationship between the similarity of the gram sets of two strings and their edit distance is studied, and used to prune the unqualified strings from the string collection. In [48], authors studied the relationship between the gram dictionary and the performance of queries. A tighter lower bound is developed using a dynamic programming algorithm. To tackle the problem of dynamic updating, the issue of how adding a new gram to an existing gram dictionary affects the index structure of the string collection is analysed. A new algorithm is developed to find a high-quality gram dictionary for the string collection. Comparing with the dictionary generation approach in [47] which requires several manually-tuned parameters, the new algorithm does not require some of the parameters and is cost-based.

4.2.2 Token based similarity

The main idea behind token-based similarity is to view the strings to be compared as sets of tokens and evaluate the similarity of the operand sets. If the similarity is high enough, the string pair is flagged as being of interest (e.g., potential duplicate). Tokens can be words, such as “CMIS/ICT Center” → “CMIS/ICT”, “Center”, or q-grams[35, 36], for instance “CMI”, “MIS”, “IS/”, “S/I”, “/IC”, “ICT”, “T_C”, “_Ce”, “Cen”, “ent”, “nte”, “ter”. The similarity between strings can be assessed by manipulating sets of tokens. Given two sets of tokens S, T, we can decide the similarity between them using Jaccard similarity which is computed by $Jaccard(S, T) = |\mathbf{S} \cap \mathbf{T}|/|\mathbf{S} \cup \mathbf{T}|$ [30]. This expression says that the Jaccard distance between strings S and T is the number of tokens that occur in both the string expansion of *S* (denoted **S**) and the string expansion of *T* (denoted **T**) divided by the number of tokens that occur in either **S** or **T**.

Set similarity operators can also be used to evaluate similarity of set-valued attributes in general (e.g., in an Object Relational DBMS). Tokens that appear very frequently in the database (like ‘Main’ or ‘St.’) carry small information content, whereas rare tokens (like ‘Maine’) are more important semantically. Hence, the more important a token is, the larger the role it should play in overall similarity. For that reason, weighted similarity measures (for example TF/IDF) use the Inverse Document Frequency (IDF) as token weights. The IDF of a token is the inverse of the total number of times that this token appears in the data collection. In addition, weighted measures also use a Term Frequency (TF) component, i.e., each token is also weighted with respect to the total number of times it appears in the multi-set.

Different term weighting systems, such as vector length normalization, TF and TF/IDF, have been introduced in [40]. In the TF/IDF similarity measure, the Inverse Document Frequency (IDF) is used as token weights. The IDF of a token is the logarithm to the base 2 of the normalised inverse of the total number of times that this token appears in the database.

Consider a database *D* of strings {*s*} where a string can be interpreted as a document, a sentence, a field value, sequence of genes etc.. From each string *s*, a set of tokens is derived T(*s*) = {*t*}. The set of all distinct tokens over the whole database is **T** = $\cup_j$T(s_j). If the number of documents in which the token t_i occurs is N(t_i), the IDF is

$$IDF(t_i) = \log_2(1 + N / N(t_i))$$

Denote the term frequency of token t_i in string *s* by TF(t_i, s). The normalized length of set s is computed as

$$len(s) = \sqrt{\sum_{t_i \in s} TF(t_i, s)^2 . IDF(t_i)^2}$$

The length normalized TF/IDF similarity of sets q and s is:

$$I(q, s) = \sum_{t_i \in T(q) \cap T(s)} \frac{TF(t_i, s).TF(t_i, q).IDF(t_i)^2}{len(s).len(q)}$$

Length normalization restricts similarity in the interval [0, 1]. If q = s, the IDF score is equal to 1. Other normalisations are sometimes used [73].

Cosine similarity is an important similarity measure used in document retrieval [37, 38, 39]. In cosine similarity, a string *s* is represented as a (potentially sparse) weighted vector *W* of high dimensionality. *W*(*s*) has an element for each possible token in a string in D; i.e. |*W*| = |**T**|. Each element of *W*(*s*) takes a value reflecting the frequency of the token to which it corresponds. The values of *W*(*s*) could be TFIDFs, TFs, etc. The cosine metric is

$$Cosine(a, b) = \frac{W(a).W(b)}{\|W(a)\|\|W(b)\|}$$

The cosine metric is so-called because it is the cosine of the angle in high-dimensional space between the vectors *W*(*a*) and *W*(*b*). If q-gram tokens are used, the cosine similarity can capture small typing mistakes, such as “jaccard” vs “jacard” → {jac, acc, cca, car, ard} vs {jac, aca, car, ard}. The common tokens “jac”, “car”, “ard” would be enough to result in high value of Cosine (“jaccard”, “jacard”). The similarity between them is always between zero and one, since every document vector has unit length. Two documents are similar if they share many “important” terms. The cosine similarity metric has been applied in many literatures using TFIDF values [37, 38, 39]. The TFIDF approach assigns higher weights to terms that occur infrequently in the collection, and which may therefore be more significant.

4.2.3 FMS similarity

String edit distance is used to measure the similarity between tuples. However, the edit distance has severe limitation when handling the fuzzy matching operations. Suppose that we have the relations shown in the following tables 1 and 2. The edit distance function would consider the input tuple I3 in Table 2 to be closest to R2 in Table 1. But in fact, the intended target is R1. edit distance fails to find the real match because it considers transforming ‘corporation’ to ‘company’ more expensive than transforming ‘boeing’ to ‘bon’. Consider another example, the edit distance considers I4 closer to R3 than to its target R1.

ID	Org. Name	City	State	Zipcode
R1	Boeing Company	Seattle	WA	98004
R2	Bon Corporation	Seattle	WA	98014
R3	Companions	Seattle	WA	98024

Table 1: Organization Reference Relation

ID	Org. Name	City	State	Zipcode
I1	Boeing Company	Seattle	WA	98004
I2	Beoing Co.	Seattle	WA	98004
I3	Boeing Corporation	Seattle	WA	98004
I4	Company Beoing	Seattle	NULL	98014

Table 2: Input Organization Tuples

The edit distance is not effective because of the following points. First, it does not make used of informative tokens. Within a field, edit distance can not distinguish between more and less informative tokens. Intuitively, we know Boeing is more informative than Corporation. Hence, matching on Boeing should mean high similarity. It means it should be expensive to transform input token into Boeing than into Corporation. Then it does not take into account the token transposition error. As shown in the tables 1 and 2, we know I4 should be matched with R1.
FMS similarity is proposed to attach weights to transformation costs for each token, and transform input tuple into reference tuple [16]. In FMS similarity matching, clean tuples are stored in reference table. Fuzzy matching is done to find the best matching clean tuples. With FMS similarity, the reference table can be very large, and the volume of input tuples can be very large. A template for using Fuzzy match is shown as Figure 1.

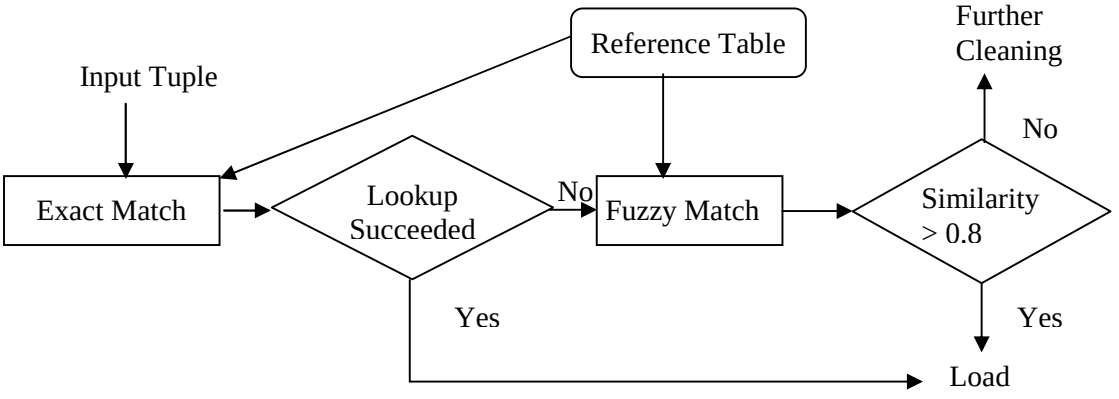


Figure 1: A template for using Fuzzy Match

To develop a domain-independent method, a fuzzy match similarity (FMS) function views a string as a sequence of tokens and recognizes varying “importance” of tokens by explicitly associating weights quantifying their importance. Tuples matching on high weight tokens are more similar than those matching on low weight tokens. Meanwhile, the token weights are considered effectively in combination with data entry errors. The IDF weight function is

adapted to the relation domain by treating each tuple as a document of tokens. Given a token t and the schema of a relation R, let the frequency of token t in column I, denoted *freq(t,i)*, e the number of tuples v in R such that tok(v[i]) contains t, then when *freq(t,i)*>0, the IDF value of the i^{th} column in the schema is computed by

$$w(t,i) = IDF(t,i) = \log \frac{|R|}{freq(t,i)}$$

Informally, the similarity between an input tuple and a reference tuple is the cost of transforming the former into the latter; the lower the cost the higher the similarity. Given an input tuple u, a reference tuple v with the schema R[A_1, … A_n], three transformation operations, token replacement, token insertion and token deletion, are considered. The cost of token replacement is ed(t_1,t_2)*w(t_1, i), where i is the column number, t_1 is token in u[i] and t_2 is that in v[i]. The cost of inserting a token t into u[i] is c_{ins}*w(t, i), where c_{ins} is token insertion factor, between 0 and 1. The cost of deleting a token t from u[i] is w(t, i).

The costs associated with inserting and deleting the same token may be different. This asymmetry is useful for the scenarios where it is more likely for tokens to be left out during data entry that for spurious tokens to be inserted. For two tuples u and v, transforming u into v requires each column u[i] to be transformed into v[i] through a sequence of transformation operations, whose cost is the sum of costs of all operations in the sequence. The transformation cost tc(u[i],v[i]) is the cost of the minimum cost transformation sequence for transforming u[i] to v[i], which is computed by.

$$tc(u,v) = \sum_i tc(u[i], v[i])$$

For example, given an input tuple u = [Beoing Corporation, Seattle, WA, 98004], and a reference tuple v = [Boeing Company, Seattle, WA, 98004], the transformation cost is tc(u[1],v[1]) = 0.33 (beoing to boeing) + 0.64 (corporation to company) = 0.97. The FUZZY SIMILARITY MATCH - similarity between u and v is defined as:

$$fms(u,v) = 1 - \min\left(\frac{tc(u,v)}{w(u)}, 1.0\right)$$

where w(u) is sum of weights of all tokens in token set tok(u). Eg: w(u) = 5.0 (considering unit weight on each token) , and fms(u,v) = 1 – 0.97/5.0 = 0.806

4.3 Efficient algorithms for approximate join

A fundamental operation in record matching is similarity join, which identifies all pairs of similar strings (records). Given two tables R and S, a similarity join between them returns all pairs of records, one each from R and S, such that the similarity function when applied to them is greater than a threshold. In this part, we will review some typical similarity join approaches. The methods are divided into two groups: traditional join methods and the extended join methods. The traditional approaches focus on the approximation join of sets containing records

that can be stored in conventional databases, while extended join methods process the approximation join of string sets that can not be supported by relational databases.

4.3.1 Traditional join methods

Traditional join methods identify all similar record pairs from two databases. Given two tables, each comprising a set of records $\underline{A} = \{A_1, \ldots, A_n\}$ and $\underline{B} = \{B_1, \ldots, B_m\}$, an approximate join of them is a subset of their Cartesian Product[1]. The records in both tables are assumed to have a set of *k* common fields numbered *1…k*. An approximate join is used to jointly match these *k* fields such that the joint similarity of these fields is above a specified threshold.

Naïvely, we could compute the similarity of each possible pair of records, and return those meeting the requirement of the similarity threshold. However, the naïve method is I/O and CPU intensive. It requires $n \times m$ matching calculations, each of which is involved, and so is not scalable to millions of records. To perform efficient approximation join over large databases, it is necessary to reduce the computational complexity of it from $O(n^2)$ to $O(n*w)$, where $w << n$, or less. The way to do this is to avoid ever attempting matching calculations between very different records.

A number of computationally efficient approaches have been proposed. In [41], the task of merging data from multiple sources in as efficient matter as possible is studied, while the accuracy of the result is maximized. The sorted neighbourhood method is used to solve the problem of merge/purge. The matching records are brought close together by sorting the records over the most important discriminating key attribute of the data (the *blocking key*). Then, the comparison of records is restricted to a small neighbourhood within the sorted list. The sorted neighbourhood method for solving the problem of merge/purge is performed in the following three phases. First, a discriminating key is computed for each record in the list by extracting relevant fields or portions of fields. Then the records in the data list are sorted using the produced keys. Finally, a fixed size window is moved through the sequential records limiting the comparisons for matching records to those records in the window. If the size of the window is w records, then every new record entering the window is compared with the previous w–1 records to find “matching” records. The first record in the window slides out of the window.

This approach is a generalization of band joins based on the duplicate elimination algorithm described in [22]. The duplicate elimination algorithms take advantage of the fact that matching records will come together in the moving window during the final phase of sorted neighbourhood method. To improve the efficiency of the method, an alternative method based on multiple blocking keys. Copies are made of the set of all the records are made, one for each blocking key. Each set is sorted in order of its corresponding blocking key, and the records are divided into blocks, one for each value of the blocking key. The sorted-neighbourhood method is then applied to each block independently. Figure 2 shows an example of merging databases using multiple compound blocking key method.

[1] The set of all possible pairs of records, one taken from each of the two tables.

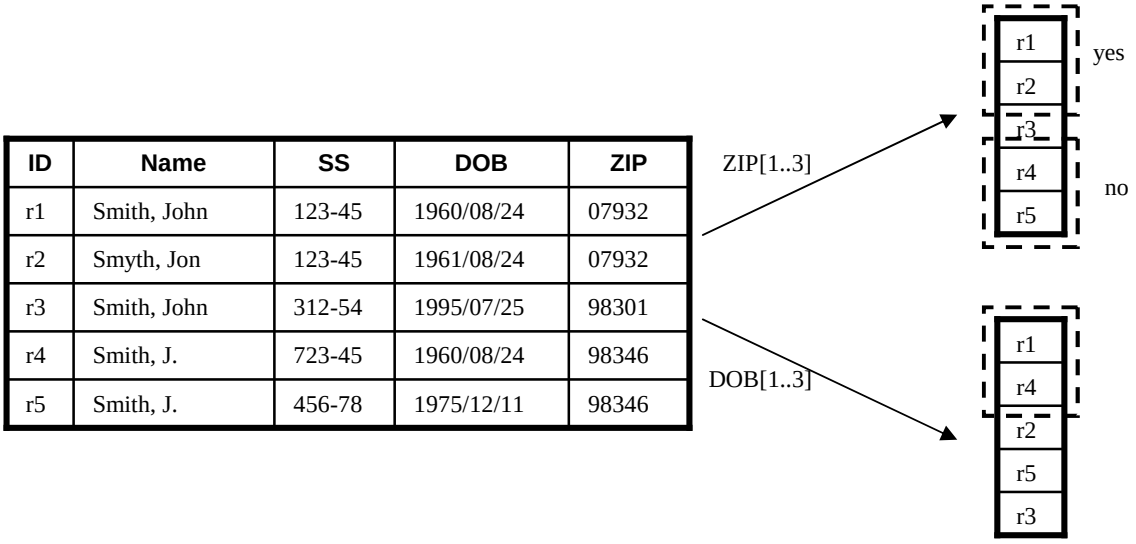


Figure 2: An example of cluster method

As shown in the figure, when we sort by the first three letters of the ZIP field ZIP[1..3], the records will be sorted as r1, r2, r3, r4 and r5. Meanwhile, if the DOB[1..3] of each record is used as its key, the sorted records will be r1, r4, r2, r5 and r3. The comparison of records to determine their equivalence is a complex inferential process that considers much more information in the compared records than the keys used for sorting. For example, in figure 2, we infer r1 and r2 are the same person, while r4 and r5 are not.

Variations to this method have been proposed in [41]. For example, OPS5 rule program consisting of 26 rules was used to compare the equivalence of records during the merge phase. The multi-pass strategy is proposed to increase the number of similar records merged. This strategy executes several independent runs of the sorted neighbourhood method, each time using a different key and a relatively small window. The multi-pass approach can drastically improve the accuracy of the results of only one run of the sorted neighbourhood method with varying large windows. The multiple “cheap” passes faster than an “expensive” one.

In [17], authors presented the BigMatch program, a new record linkage tool for use in matching records in table *A* with very large number of records against a table *B* of moderate size. The BigMatch program allows the user to specify several blocking criteria and the matching field parameters. For each of several blocking criteria, the program can extract likely matching record candidates from the large file without requiring sorting of either file. Record pairs from two files are brought together to be compared based on multiple keys when they agree on a specified blocking criterion. In BigMatch, a smaller table is stored in main memory. For each set of blocking criteria, a table of the keys is created by reading the moderate file B. The program allows one to run several different blocking criteria for a very large file and a moderate size file requiring the large file to be read just once and without requiring sorting either file. The algorithm is made to run efficiently by the construction of a set of indexes, one for each blocking key. Each index stores all the records that have each value of blocking key observed in the small table (B). With the indexes are contstructed, the algorithm proceeds down the large table, processing records one at a time. For each record it retrieves from each index the set of records in the small table that lie in the same block. The union of these are candidate for merger with the current record in the large table. An example of the BigMatch approach is shown as Figure 3. In the figure, given a record from outer (large) table, the first blocking

criterion (SS[1..2]) provides two possible matches (r1 and r2), while the second (ZIP[1..4]) provides two more (r4 and r5). When we check each of these, only r1 and r4 are acceptable matches.

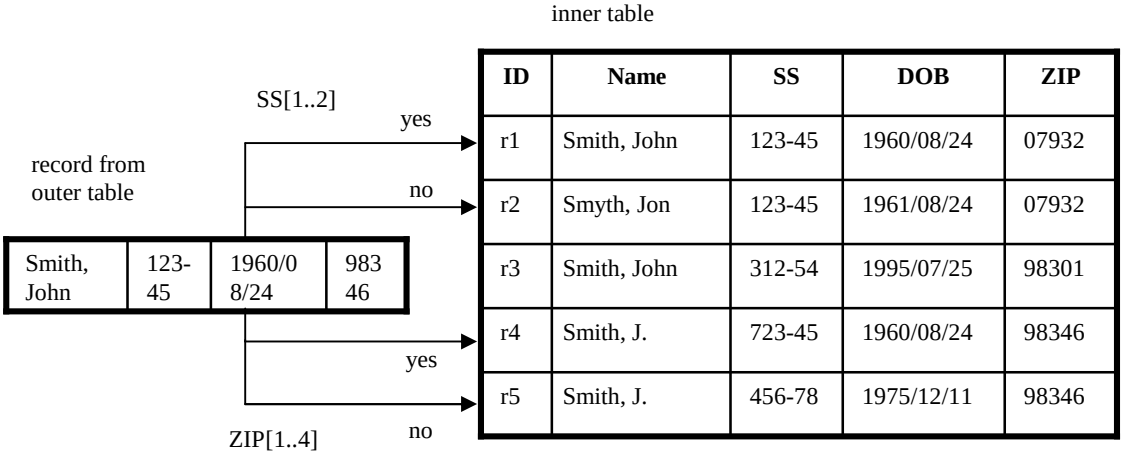


Figure 3: An example of BigMatch approach

4.3.2 Extended join methods

Since commercial databases do not support approximate string similarity joins directly, it is a big challenge to efficiently implement this functionality with user-defined functions (UDFs). Approaches have been proposed for efficient approximate string similarity join with various distance function constraints. The typical examples include the q-gram set join[30], Probe-Cluster[42], SSjoin[18], ppjoin [50] and Ed-Join [49] etc. In this part, we will review several typical approaches.

In [30], a technique was developed for building approximation string join capabilities on top of commercial databases by exploiting facilities already available in them. The functionality of string join is efficiently implemented by computing the thresholded edit distance on string attributes. Given two tables R_1 and R_2 with string attributes (fields) $R_1.A_i$ and $R_2.A_j$, and an integer k, approximate string joins retrieve all pairs of records $(t, t') \in R_1 \times R_2$ such that the edit distance $ED(R_1.A_i(t), R_2.A_j(t')) \leq k$. The proposed technique relies on matching short substrings of length q, called q-grams. Given a string s, a set of all overlapping q-grams are first extracted from s. To devise effective algorithms for approximate string join using the concept of q-grams, both the positions of individual matches and the total number of matched q-grams are taken into consideration. Thus a set of candidate string pairs with a few false positives is identified. Meanwhile, no false dismissals under edit distance metric as well as its variants are guaranteed. This approach applies to both approximate full string matching and approximate substring matching, with a variety of possible edit distance functions. The approximate string match predicate, with a suitable edit distance threshold, can be mapped into a vanilla relational expression and optimized by conventional relational optimizers.

The techniques for approximate string processing in databases share a principle common in multimedia and spatial algorithms. A set of candidate answers is first obtained using a cheap, approximate algorithm that guarantees no false dismissals. All false positives are then eliminated by checking the edit distance between each candidate string pair using an expensive,

in-memory algorithm. This filtering process is performed by a join on the q-grams. The q-gram properties are utilized to develop filtering techniques for efficiently identifying candidate answers. Three cheap filters (length, count, position) are used to prune non-matches. When applying edit distance metric in approximate string joins, the following inequality holds:

$$ED(s_1, s_2) \le d \rightarrow \left|Q(s_1) \cap Q(s_2)\right| \ge \max(\left|s_1\right|, \left|s_2\right|) - (d-1) * q - 1$$

Table 3 shows an example of filtering based on this property. Suppose we have 3 strings whose ids are r_1, r_2 and r_3 respectively. The Name attribute of each can be transformed into a set of 3-grams as shown in the table. We can easily compute the edit distance between r1 and r2 ED(r1,r2)=1, that between r1 and r3 ED(r1,r3)=2. Meanwhile, based on the 3-grams of them, we know $|Q(r1) \cap Q(r2)| = 10$ and $|Q(r1) \cap Q(r3)| = 7$. Clearly, we can easily verify (1) $ED(s_1,s_2) \le d \rightarrow |Q(s_1) \cap Q(s_2)| \ge \max(|s_1|,|s_2|) - (d-1)*q-1$ and (2) $ED(s_1,s_3) \le d \rightarrow |Q(s_1) \cap Q(s_3)| \ge \max(|s_1|,|s_3|) - (d-1)*q-1$.

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	##s, #sh, shr, hri, riv, iva, vas, ast, sta, tav, av$, v$$

Table 3: Positioning q-grams of records

The most interesting thing about these filters is that they can be naturally expressed as an SQL expression on the augmented database. The approximate string join is based on the cost of substring matching, which reduce approximate join problem to aggregated set intersection. Take the records in Table 3 as an example. When we match r_1 and r_3, the process can be expressed as using SQL expression as follows.

Q:

```
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
```

Figure 4 shows this process of using this SQL expression over r1 and r3.

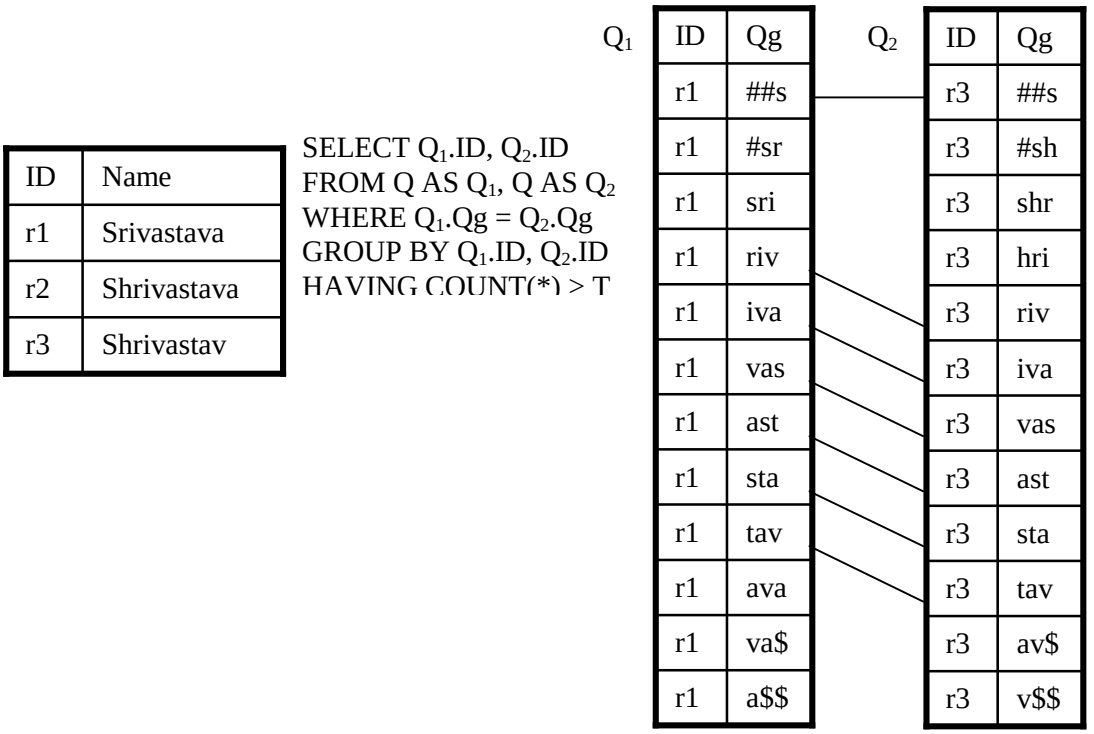


Figure 4: Computing thresholded edit distance join on string attributes using SQL expression

In [42], an efficient algorithm called Probe-Cluster was proposed to compute similarity joins over set/text attributes. This work aims at presenting a generic algorithm for set join based on similarity predicates involving various similarity measures like intersect size, Jaccard-coefficient, cosine similarity, and edit-distance. The Probe-Count algorithm is derived from the way keyword queries are answered during Information Retrieval using an inverted index. The index maps words to the list of record identifiers that contain that word. Such an index can be constructed in memory in one sequential scan of the data by inserting each scanned record into the record list associated with all words the record contains. In Probe-Count, a string is first mapped to a set of tokens (words, q-grams, etc.). An index is then constructed for each token type (word, q-gram etc.).

Figure 5 shows an example of a set join based on similarity predicates. Given three strings r1, r2 and r3, a set of 3-grams are derived from each string. The 3-grams record the information of the set of strings, as shown at the left of the figure. The corresponding index is shown on the right.

ID	3-grams
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$$}

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

Figure 5: set join based on similarity predicate

The aim is to find likely duplicates. After constructing the index, the data in the record table (left) is scanned again. For each *active record r*, the rows in the index table (right) corresponding to each 3-gram are retrieved. The value of each row is the set of records possessing the corresponding 3-gram. The tokens may then be weighted by their corresponding IDF weight and the weights summed. The records with sums of weights greater than the threshold are merged with the active record *r*. Several optimization strategies have been used to improve the efficiency of this calculation, two of which are presented below.

In order to produce a sorted list derived by merging a set of lists (e.g. as above, where there is a list for each token value) the component lists are presorted by weight. The records in each list the value of which are greater than the threshold are accepted for merge and need not be considered further. This can be done efficiently with a doubling binary search or, better still incorporated within the sorting process. A record that satisfies the threshold condition must appear in at least one of the remaining lists. Records are popped from the remaining lists in descending order of the maximum value over all lists and checked. After each check the remaining lists are checked to see if the sum of their maximum values is above the threshold. If not, the search is stopped.

The merge time can be reduced even further by pre-sorting the records in decreasing order of the number of tokens in the record. This ensures that records with a large number of words get processed faster. Since the running time has a log t factor, it helps to process long records (with large t) when the size of each ID-list in the index is smaller. IR query optimizations are very useful for approximate joins. Figure 6 shows an example of the optimization approach by pre-sorting. The index structure is derived from that in Figure 5, but sorts the lists in decreasing order. As shown in the figure, for each 3-gram, the strings containing it, any one or more of r1,

r2 and r3, are sorted by their length. Here, for example, r2 has most tokens, and will be put to the first position of the corresponding record in the index once it contains a certain 3-gram.

Figures 7-9 show an example of optimization on how to process the skewed lists. In the first step, as shown in Figure 7, the shortest lists, the lists to #sr, sri and v$$, are processed. Then the lists that are to the 3-grams appearing in two strings, #sh, shr, hri and ava, are processed. This is shown in Figure 8. Finally, shown in Figure 9, the longest lists that are correspond to the 3-grams, ##s, riv and tav, are processed.

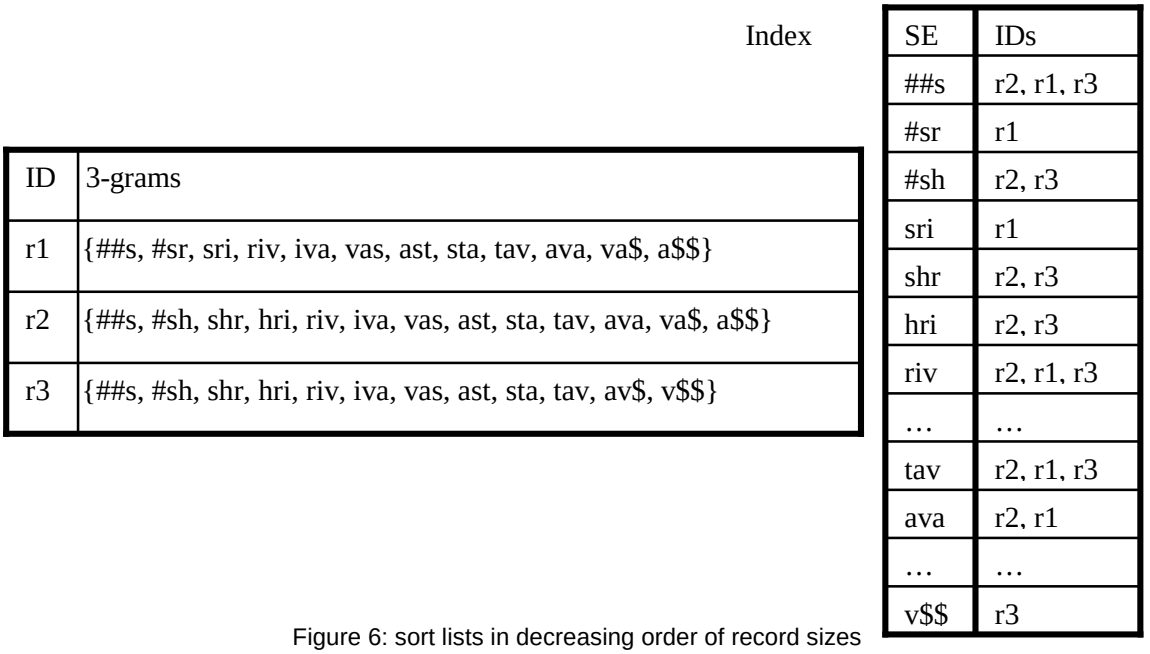


Figure 6: sort lists in decreasing order of record sizes

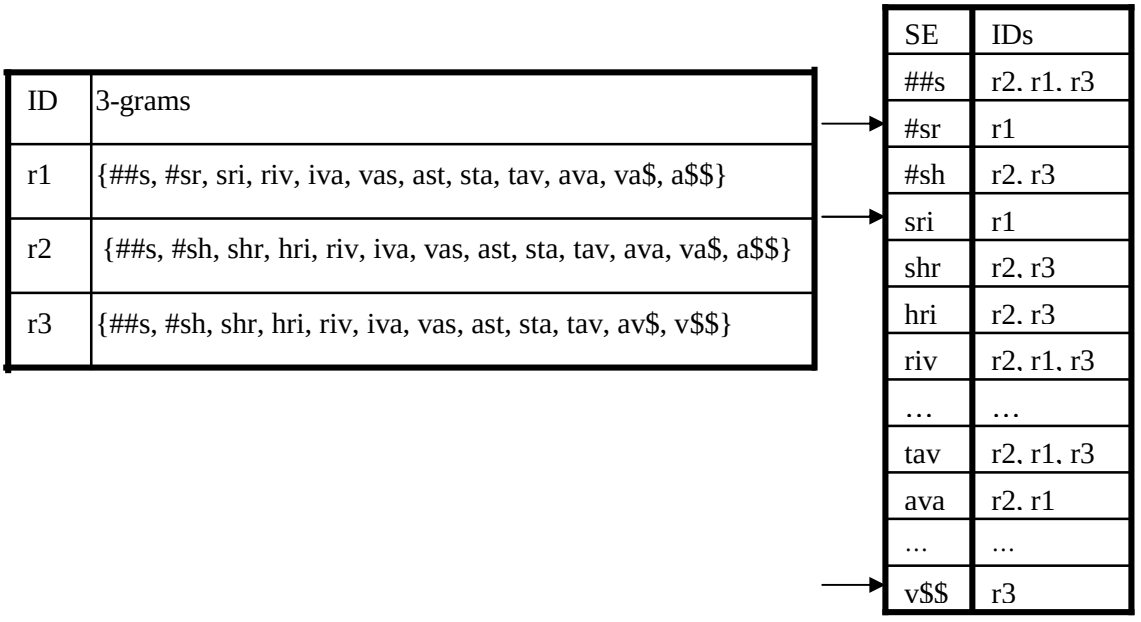


Figure 7: an example of processing skewed lists in increasing size order (step 1)

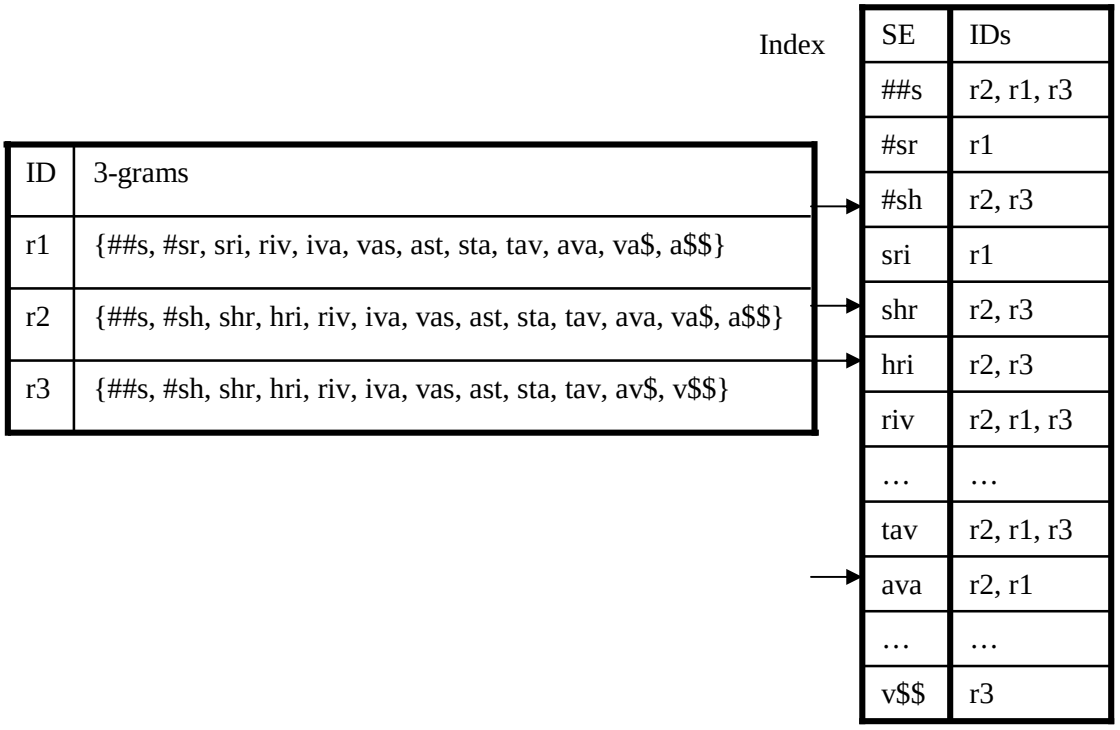


Figure 8: an example of processing skewed lists in increasing size order (step 2)

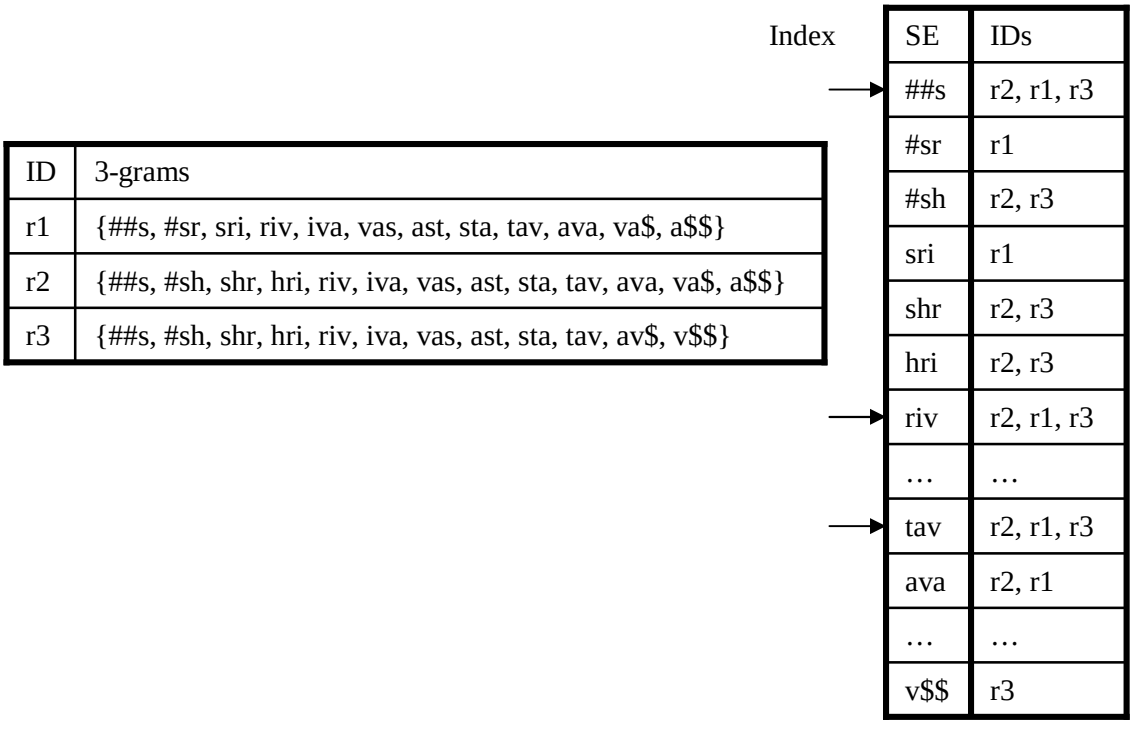


Figure 9: an example of processing skewed lists in increasing size order (step 3)

In [18], the authors aim at developing generic algorithm for set join based on a similarity predicate. They propose the SSJoin operator as a foundational primitive and show that it can be used for supporting similarity joins based on several string similarity functions−e.g., edit similarity, jaccard similarity, generalized edit similarity, hamming distance, soundex, etc.−as well as similarity based on cooccurrences. In defining the SSJoin operator, the observation that set overlap can be used effectively to support a variety of similarity functions is exploited. The SSJoin operator compares values based on "sets" associated with (or explicitly constructed for) each one of them. The design and implementation of this logical operator leverages the existing set of relational operators, and helps define a rich space of alternatives for optimizing queries involving similarity joins.

More recent work focuses on the developing more efficient filtering strategies. For example, in [50], a positional filtering principle was proposed by exploiting the ordering of tokens in a record, which leads to upper bound estimates of similarity scores. The proposed positional filtering techniques are integrated into the existing filtering methods, the candidate sizes is dramatically reduced, thus the efficiency of similarity join is improved. In [49], the mismatch-based filtering methods are exploited for efficient similarity join with edit distance constraints. With the help of new edit distance lower bounds based on the locations and contents of mismatching q-grams, a substantial reduction of the candidate sizes is achieved, and thus the computation time is reduced.

4.3.3 Commercial systems

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;

5. HARDWARE ISSUES

Working with very large data sets (order Terabytes) poses challenges for existing computing hardware and software. The time to transfer data from disk to system memory lags the ability of CPU's to perform computations on the data by order 10-100 times. In data intensive computing where the number of computations on the data is small and the amount of data that must be read is large the CPU will spend most of the time waiting for data to be loaded into memory. The resulting code will be very inefficient on very large data sets (Gokhale et al., 2008).

A number of technologies that will produce improvements in performance in data intensive computing are currently under active development. These technologies are likely to produce dramatic improvements in a broad range of data mining, streaming data and data analysis applications. In summary these developments are:-

- Solid state disk drives
- FPGA's
- GPU's
- IBM system S
- Cloud computing

We provide a brief summary of these technologies and their relevance to data mining following.

5.1 Solid state disk drives

Solid state disk drives using solid state ('flash') memory deliver a significant improvement in disk access times. SSD's low latency will significantly speedup data mining applications as reading from disk is a major bottleneck in getting data from disk to system memory. SSD's are now widely available however their higher cost, relatively small capacity and reliability, limit their broad application to data mining at the moment.

5.2 Field programmable gate arrays

Field programmable gate arrays (FPGA's) perform calculations directly in hardware rather than software which results in substantial speedup in computations. FPGA's have been located near disk drives, with data streaming from disk to the FPGA. A database query can then be processed directly on the FPGA's. This avoids the transfer of data over a network, into memory and back again eliminating much of the delay caused by data movement to and from the CPU. An example of this technology is available from Netezza which claims 50-100x speedups are possible on data intensive problems. Lawrence Livermore National Laboratory has obtained such speedups.

http://www.netezza.com/data-warehouse-appliance-products/index.aspx

5.3 Graphical Processing Units

Graphics processing units (GPU's) are now available from NVIDIA in a 1U format including 4 Tesla T10 series GPU's (1 Tflop peak) which allow GPU based clusters to be deployed. CSIRO has recently deployed a 200 Tflop cluster using this technology. Data mining applications that require intensive computing such as video, image reconstruction & analysis and data fusion could benefit from GPU technology which can currently deliver 1 Teraflop in single precision (order 10-20x CPU) and can be installed in desktop computers.

http://www.nvidia.com/page/technologies.html

5.4 IBM System S

IBM System S is currently under active development by IBM. It has recently become available from IBM. IBM System S is a scalable high performance computing platform which provides the infrastructure that will underpin working with large amounts of streaming structured and unstructured data in a broad range of formats from thousands of data streams in real time. This platform should allow the rapid development of complex data analysis and/or data fusion tools.

IBM claims that "InfoSphere Streams supports high volume, structured & unstructured streaming data sources such as images, audio, voice, VoIP, video, TV, financial news, radio, police scanners, web traffic, email, chat, GPS data, financial transaction data, satellite data, sensors, badge swipes, etc"

http://www-01.ibm.com/software/data/infosphere/streams/

5.5 Parallel I/O systems

Parallel I/O systems on large clusters overcome the barrier of data transfer between disk and memory by dividing the read operations into a large number of smaller read or write operations. Data intensive applications running on large clusters can realise significant speedups. As data sets become ever larger parallel I/O systems will play an essential role in data mining. The Neteeza data mining appliances exploit this approach in combination with FPGA technology. Parallel I/O systems are now widely available.

http://www.inf.ed.ac.uk/undergraduate/projects/mathiasengvall/

5.6 Cloud Computing

Cloud computing has recently emerged as a major new platform for providing computing services. Instead of building large computational infrastructures to serve computing needs of an organisation these services can now be purchased as a service. Cloud computing offers the potential of greater flexibility and enormous scalability that is only possible at the data centres of major corporations such as Amazon, IBM, Google, and Microsoft. Haddoop

(http://hadoop.apache.org/core/) an open source software implementation of Google MapReduce allows thousands of processors to work together to perform analysis of large data sets. Google has demonstrated how their MapReduce technology can be used to efficiently search the entire content of the world wide web archived by Google. (http://labs.google.com/papers/mapreduce.html)

6. CONTRIBUTIONS FROM ASIA

The research on data cleaning and record linkage area has been active in recent years. The contributions mainly include the publications at the top level conferences and journals. For example, Kot et al. proposed to adaptively detect attribute outliners in XML documents using the correlation between attributes [43]. Cheng et al. proposed to clean uncertain data for achieving better query or service quality under a limited budget. With the query information, the set of data items to be cleaned is decided. A PWS-query metric was proposed to measure the quality of query for the probabilistic database. At the same time, the range and max queries are efficiently computed. In [44], Cheng et al. proposed several efficient approaches for evaluating the imprecise queries.

The researches in Asia have made significant academic contributions. However, the work we have reviewed on data cleaning from Asian sources is unlikely to motivate significant useful operational capability because of the limitations, such as the scalability of techniques and the dependence on other related techniques.

Casting the net more widely, there are significant numbers of researchers of Asian origin producing first class work, for instance the authors of [72], who are based in Australia. It would be natural for such talented people to move between Asia and the west in the course of their careers.

7. OPEN PROBLEMS AND FUTURE RESEARCH

The key challenges in this domain lie in the following areas: scalability, accuracy and complexity of the model relating entities to records and references. The research topics we have identified below are concerned with pursuit of these issues.

- **Benchmarks.** In recent years, many algorithms and similarity measures have been proposed for the issues of data cleaning, however, evaluation the proposed approaches is not completely solved. Only small test data sets are available and large test sets have to be provided via simulation (e.g. the deliberate injection of automatically produced near duplicates). This situation makes it difficult to convincingly demonstrate progress on accuracy. The most cost effective approaches are likely to involve boot-strap techniques, in which the results good automatic algorithms are corrected by human intervention directed at cases most likely to have been in error.

- **Accuracy.** Existing approaches are based on merge-only approaches to clustering and linkage. In the statistical clustering literature significant improvements in accuracy are

provided via the inclusion of other approaches such as switching records between clusters which facilitates optimisation.

- Indeed the entity resolution has not, so far, been cast as an **optimisation** problem. One of the authors can report from his own researches that this can be done without significantly increased computational burden. The infrastructure of optimisation can then be brought to bear on the problem, leading to massive simplifications in the design of much more complex approaches.

- Interestingly there appears to be significant **disconnect between the three literatures** of cross-document reference resolution, database entity resolution and information retrieval, which have all made valuable contributions to the area. As yet, the authors see no sign within the public or commercial domains (from the information we have on commercial systems) that such a synthesis had yet been made. The authors believe that such an approach is likely to provide rich pickings.

- **Record/Entity interrelationship models.** The record interrelationship model underlying entity resolution is particularly simple – we are interested in records that refer to the same entity. However, that model is just the simplest beginning. Two further extensions include the extraction of *histories* of entities, and of the relationships entities have with each other. More interestingly we might seek to extract the *histories of entities' interactions with each other*. There is a limited amount of literature on each of these topics, but so far as we are aware, there is no scalable operational system that absorbs its information from free text sources which provides good performance in any of these areas.

8. OPPORTUNITIES AND BARRIERS TO ADOPTION

The technology currently being created under the inconspicuous headings of *data cleaning* and *record linkage* is of great significance. The creation of effective systems in the area is likely to have a breakthrough effect on the ability of companies and other organisations to understand their data in terms of relationships between underlying entities (people and organisations) rather than snippets of partial information that currently litter their document repositories and databases. It is not beyond the bounds of possibility that the people and organisations referred to over the whole of the world wide web could be substantially disambiguated by a concerted program over the next five years. The potential opportunities, commercial and otherwise, from such technology are obvious. The consequences for organisations of not having such technology when their competitors do are likely to be unpleasant and sharply felt. It is the opinion of the authors that such a development would require further significant developments in the underlying applied mathematics and in software designed to take advantage of it and the new developments in hardware that we have described. The organisations best suited to carry out such work are the search engine providers. Were they to collaborate with world wide financial services organisations (such as Visa and Mastercard) the work would tend to be more more accurate, more highly connected and of even greater value.

REFERENCES

[1] Health Result Team for Information Management, 2005 Data Quality Report – The State of Data Quality in Ontario Executive Summary, 2005.

[2] T. C. Redman, The impact of poor data quality on the typical enterprise. Communications of ACM, 2:79–82, 1998.

[3] L. English. Plain English on data quality: Information quality management: The next frontier. DM Review Magazine, April 2000.

[4] TDWI's data quality report 2009

[5] W. W. Eckerson, Excerpt from TDWI's Research Report – Data Quality and the Bottom Line, Business Intelligence Journal. 2009

[6] W. W. Eckerson, Data Warehousing Special Report: Data quality and the bottom line, 2002

[7] TDWI's industry study report 2000

[8] W. W. Eckerson, Data Quality and The Bottom Line: Achieving Business Success through a Commitment to High Quality Data, 2002

[9] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In PODS, 1999.

[10] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. IEEE Data Engineering Bulletin, 23(4), 2000.

[11] I. Fellegi and D. Holt. A systematic approach to automatic edit and imputation. J. American Statistical Association, 71(353):17–35, 1976.

[12] W. E. Winkler. Methods for evaluating and creating data quality. Inf. Syst., 29(7):531–550, 2004.

[13] A. K. Elmagarmid, P. G. Ipeirotis and V. S. Verykios. Duplicate Record Detection: A Survey. In TKDE, Vol. 19, No. 1.pages 1-16, Jan. 2007.

[14] http://en.wikiversity.org/wiki/Duplicate_record_detection

[15] H. Garcia-Molina. Entity Resolution: Overview and Challenges. In ER, page 1-2, 2004

[16] S. Chaudhuri, K. Ganjam, V. Ganti, R. Motwani: Robust and Efficient Fuzzy Match for Online Data Cleaning. In SIGMOD, pages 313-324, 2003.

[17] W. E. Yancey: BigMatch: A program for extracting probable matches from a large file for record linkage. RRC 2007-01. Statistical Research Division, U.S. Bureau of the Census.

[18] S. Chaudhuri, V. Ganti, and R. Kaushik. A Primitive Operator for Similarity Joins in Data Cleaning. In ICDE, page 5, 2006.

[19] S. Chaudhuri, V. Ganti and R. Motwani: Robust Identification of Fuzzy Duplicates. In ICDE, pages 865-876, 2005.

[20] S. Chaudhuri, A. D. Sarma, V. Ganti and R. Kaushik. Leveraging aggregate constraints for deduplication, In SIGMOD, pages 437-448, 2007.

[21] R. Ananthakrishna, S. Chaudhuri and V. Ganti: Eliminating Fuzzy Duplicates in Data Warehouses. In VLDB, page 586-597, 2002

[22] D. Bitton, D. J. DeWitt: Duplicate Record Elimination in Large Data Files. TODS, 8(2), pages 255-265, 1983

[23] M. Bilenko, R. J. Mooney: Adaptive duplicate detection using learnable string similarity measures. In KDD, pages: 39 – 48, 2003.

[24] C. Xiao, W. Wang, X. Lin, H. Shang, Top-k Set Similarity Joins, In ICDE, pages 916-927, 2009.

[25] E. Agichtein and V. Ganti, Mining reference tables for automatic text segmentation. In KDD, pages 20-29, 2004.

[26] D.Sylwester and S.Seth, A trainable, single-pass algorithm for column segmentation, In ICDAR, page 615, 1995.

[27] M.Ozaki. Column segmentation by white space pattern matching. In ICDAR, page 134, 1995.

[28] W. E. Winkler. The state of record linkage and current research problems. Technical report, U.S. Bureau of the Census, 1999.

[29] E. Ukkonen. On approximate string matching. In FCT, 1983.

[30] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In VLDB, page 491-500, 2001.

[31] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In KDD, 2002.

[32] G. Salton. Automatic Text Processing. Addison Wesley, 1989.

[33] G. Kowalski. Information retrieval systems: theory and implementation. Kluwer Academic Publishers, 1997.

[34] G. Navarro, R. Baeza-Yates, E. Sutinen, and J. Tarhio. Indexing methods for approximate string matching. IEEE Data Engineering Bulletin, 24(4):19--27, 2001.

[35] R. Baeza-Yates and G. Navarro. A practical index for text retrieval allowing errors, In CLEI, pages 273--282, 1997.

[36] G. Navarro, E. Sutinen, J. Tanninen, and J. Tarhio. Indexing text with approximate q-grams, In CPM, LNCS 1848, 2000.

[37] W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In SIGMOD, pages 201-212, 1998.

[38] W.Cohen. Data integration using similarity joins and a word-based information representation language, TOIS, 18(3):288--321, 2000.

[39]E. Cohen and D. Lewis. Approximating matrix multiplication for pattern recognition tasks, In SODA, pages: 682 – 691, 1997.

[40]G. Salton and C. Buckley. "Term-weighting approaches in automatic text retrieval". Information Processing & Management 24 (5): 513–523, 1988

[41] M. A. Hernández and S. J. Stolfo: The Merge/Purge Problem for Large Databases. In SIGMOD, pages 127-138, 1995.

[42] S. Sarawagi and A. Kirpal: Efficient set joins on similarity predicates. In SIGMOD, pages 743-754, 2004.

[43] J.L.Y.Koh, M.-L. Lee, W. Hsu, and W. T. Ang. Correlation-based attribute outlier detection in xml, In ICDE, pages 1522-1524, 2008

[44]R. Cheng, J. Chen, and X. Xie. Cleaning uncertain data with quality guarantees. In VLDB, pages 722-735, 2008.

[47] C. Li, B. Wang, and X. Yang. Vgram: Improving performance of approximate queries on string collections using variable-length grams. In VLDB, pages 303-314, 2007.

[48] X.Yang, B.Wang, and C.Li. Cost-based variable-length-gram selection for string collections to support approximate queries efficiently. In SIGMOD, pages 353-364, 2008.

[49] C. Xiao, W. Wang, X. LIN, Ed-Join: An Efficient Algorithm for Similarity Joins With Edit Distance Constraints. In VLDB, pages 933-944. 2008.

[50] C.Xiao, W. Wang, X. LIN, J.X. Yu, Efficient Similarity Joins for Near Duplicate Detection, In WWW, pages 131-140, 2008.

[51] I. Bhattacharya and L. Getoor: Iterative record linkage for cleaning and integration. In DMKD: pages 11-18, 2004.

[52] P. Christen, T. Churches and X. Zhu: Probabilistic name and address cleaning and standardization. Australasian Data Mining Workshop 2002

[53] S. Chaudhuri, K. Ganjam, V. Ganti, R. Kapoor and V. R. Narasayya, Theo Vassilakis: Data cleaning in Microsoft SQL server 2005. In SIGMOD, pages 918-920, 2005.

[54] W. W. Cohen, P. Ravikumar and S. E. Fienberg: A Comparison of String Distance Metrics for Name-Matching Tasks. IIWeb 2003: 73-78

[55] D. J. DeWitt, J. F. Naughton and D. A. Schneider: An Evaluation of Non-Equijoin Algorithms. In VLDB, pages 443-452, 1991.

[56] I. Fellegi, A. Sunter: A theory of record linkage. Journal of the American Statistical Association, Vol 64. No 328, 1969

[57] D. Gusfield: Algorithms on strings, trees and sequences. Cambridge university press 1998

[58] H. Galhardas, D. Florescu, D. Shasha, E. Simon and C.-A. Saita: Declarative Data Cleaning: Language, Model, and Algorithms. In VLDB, pages 371-380, 2001.

[59] L. Gravano, P. G. Ipeirotis, N. Koudas and D. Srivastava: Text joins in an RDBMS for web data integration. In WWW, pages 90-101, 2003.

[60] S. Guha, N. Koudas, A. Marathe and D. Srivastava : Merging the results of approximate match operations. In VLDB, pages 636 - 647, 2004.

[61] D. Gibson, J. M. Kleinberg and P. Raghavan: Clustering Categorical Data: An Approach Based on Dynamical Systems. In VLDB, pages 311-322, 1998.

[62] L. Jin, C. Li and S. Mehrotra: Efficient Record Linkage in Large Data Sets. In DASFAA, pages 137, 2003

[63] P. Jokinen and E. Ukkonen: Two Algorithms for Approximate String Matching in Static Texts. In MFCS, pages 240-248, 1991.

[64] D. V. Kalashnikov, S. Mehrotra and Z. Chen: Exploiting Relationships for Domain-Independent Data Cleaning. In SDM,2005.

[65] N. Koudas, A. Marathe and D. Srivastava: Flexible String Matching Against Large Databases in Practice. In VLDB, pages 1078-1086, 2004.

[66] N. Koudas, A. Marathe and D. Srivastava: SPIDER: flexible matching in databases. In SIGMOD, pages 876-878, 2005.

[67] M.-L. Lee, T. W. Ling and W. L. Low: IntelliClean: a knowledge-based intelligent data cleaner. In KDD, pages 290-294, 2000.

[68] A. E. Monge and C. Elkan: The Field Matching Problem: Algorithms and Applications. In KDD, pages 267-270, 1996.

[69] Gokhale, M.; Cohen, J.; Yoo, A.; Miller, W.M.; Jacob, A.; Ulmer, C.; Pearce, R.; Computer. Volume 41, Issue 4, April 2008 Page(s):60 – 68

[70] V. R. Borkar, K. Deshmukh, and S. Sarawagi. Automatic segmentation of text into structured records. In Proceedings of the ACM SIGMOD Conference, 2001.

[71] B. S. T. Bocek, E. Hunt. Fast Similarity Search in Large Dictionaries. Technical Report ifi-2007.02, Department of Informatics, University of Zurich, April 2007.

[72] Wei Wang, Chuan Xiao, Xuemin Lin, Chengqi Zhang. “Efficient Approximate Entity Extraction with Edit Distance Constraints”, SIGMOD 2009.

[73] Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, “Introduction to Information Retrieval”, Cambridge University Press, 2008.

Contact Us
Phone: 1300 363 400
+61 3 9545 2176
Email: enquiries@csiro.au
Web: www.csiro.au

Your CSIRO

Australia is founding its future on science and innovation. Its national science agency, CSIRO, is a powerhouse of ideas, technologies and skills for building prosperity, growth, health and sustainability. It serves governments, industries, business and communities across the nation.