University of Louisville
ThinkIR: The University of Louisville's Institutional Repository

Electronic Theses and Dissertations

5-2008

Matching records in multiple databases using a hybridization of several technologies.

Xiaoyi Wang 1962-
University of Louisville

Follow this and additional works at: https://ir.library.louisville.edu/etd

Recommended Citation

Wang, Xiaoyi 1962-, "Matching records in multiple databases using a hybridization of several technologies." (2008). *Electronic Theses and Dissertations.* Paper 1511.
https://doi.org/10.18297/etd/1511

This Doctoral Dissertation is brought to you for free and open access by ThinkIR: The University of Louisville's Institutional Repository. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of ThinkIR: The University of Louisville's Institutional Repository. This title appears here courtesy of the author, who has retained all other copyrights. For more information, please contact thinkir@louisville.edu.

MATCHING RECORDS IN MULTIPLE DATABASES USING A HYBRIDIZATION OF SEVERAL TECHNOLOGIES

By

Xiaoyi Wang
B.S., Tongji University, 1985
M.S., University of Detroit Mercy, 1999

A Dissertation
Submitted to the Faculty of the
Graduate School of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

Department of Industrial Engineering
University of Louisville
Louisville, KY 40292

May 2008

Copyright 2008 by Xiaoyi Wang

All right reserved

MATCHING RECORDS IN MULTIPLE DATABASES
USING A HYBRIDIZATION OF SEVERAL TECHNOLOGIES

By

Xiaoyi Wang
B.S., Tongji University, 1985
M.S., University of Detroit Mercy, 1999

A Dissertation Approved on

April 8, 2008

by the following Dissertation Committee:

_____
Dissertation Director, Dr. Suraj M. Alexander

_____
Dr. William E. Biles

_____
Dr. Gerald W. Evans

_____
Dr. Hichem Frigui

_____
Dr. John S. Usher

DEDICATION

This dissertation is dedicated to my family and many friends. A special feeling of gratitude to my loving parents, Jizhong Wang and Li Zhang, is for their words of encouragement, understanding and supportive. It's they that bestow me the characters of diligence, resolution, and perseverance. Without their never-ended encouragement and support in these years, I could not finish this dissertation work.

This dissertation is dedicated to my wife, Yi Zhu, who has been proud and supportive of my work and who has shared the many uncertainties, challenges and sacrifices for completing this dissertation.

This dissertation is dedicated to my daughter, Angela, who seems to be growing into a wonderful human being, in spite of the fact that her father was less available than he should have been during the first two years of her life. This is also especially dedicated to my son, Bill, who was born on March 27th, 2008, about ten days before the day of my dissertation defense. He is a special gift and reward for completing my doctorate program.

I dedicate this dissertation to my many friends who have supported me throughout the process. I will always appreciate all they have done, especially Dazhuo Li for helping

me to develop my technology skills in data preprocessing, and sharing his knowledge and ideas with me.

I also dedicate this work and give special thanks to my best friends Dr. Jianan Huang, Dr. Bert Lin and his wife Mrs. May Lin, Dr. Abey Kuruvilla and Mr. Scott Chen whose friendship, hospitality, knowledge, and wisdom have supported, enlightened, and entertained me over the many years of our friendship. They have consistently helped me keep perspective on what is important in life. They are always there for supporting me every aspect in my life throughout the entire doctorate program. I could not go through this critical period in my life without them.

ACKNOWLEDGEMENTS

At this time I would like to recognize all those whose efforts and contributions helped make this dissertation possible. Although this is my personal work, it is also the culmination of contributions from many people and resources.

This dissertation would not have been possible without the expert guidance of my esteemed advisor, Professor Suraj M. Alexander. Many thanks go out to my mentor Dr. Alexander whose advice kept me on the straight and narrow. Not only was he readily available for me, as he so generously is for all of his students, but he always read and responded to the drafts of each chapter of my work more quickly than I could have hoped. His oral and written comments are always extremely perceptive, helpful, and appropriate.

My thanks go out to my other committee members who were more than generous with their expertise and precious time. Each guided me through hurdles, provided support, and much needed focus. Their efforts and support are gratefully appreciated. Thank you, Dr. William Biles, Dr. Gerald Evans, Dr. Hichem Frigui, and Dr. John Usher for agreeing to serve on my committee.

Special thanks go out to Dr. Barry Griffin who initially introduced me to this interesting research topic from real-world. I would like to acknowledge and thank Dr. Griffin for his countless hours of reflecting, reading, encouraging, and most of all patience throughout the entire process.

The entire process of writing this dissertation has helped me grow personally and professionally. My sincere gratitude goes to Dr. Mehmed Kantardzic who helped me see beyond my personal limitations.

I must acknowledge as well the many friends, teachers, staff and other librarians who assisted, advised, and supported my research and writing efforts over the years. My thanks from the bottom of my heart must go to my many former professors at Wayne State University, especially, Dr. Donald Falkenburg, Dr. Namyku Park, and Dr. Kai Yang who assisted, advised, mentored and supported my study and research. I am also grateful to many persons who shared their memories and experiences, especially Dazhuo Li, Aldo McLean, Dengzhi Wang, and many more. Finally, I need to express my gratitude and deep appreciation to the Writing Center at University of Louisville for their proofreading.

ABSTRACT

MATCHING RECORDS IN MULTIPLE DATABASES
USING A HYBRIDIZATION OF SEVERAL TECHNOLOGIES

Xiaoyi Wang

April 8, 2008

A major problem with integrating information from multiple databases is that the same data objects can exist in inconsistent data formats across databases and a variety of attribute variations, making it difficult to identify matching objects using exact string matching. In this research, a variety of models and methods have been developed and tested to alleviate this problem. A major motivation for this research is that the lack of efficient tools for patient record matching still exists for health care providers. This research is focused on the approximate matching of patient records with third party payer databases. This is a major need for all medical treatment facilities and hospitals that try to match patient treatment records with records of insurance companies, Medicare, Medicaid and the veteran's administration. Therefore, the main objectives of this research effort are to provide an approximate matching framework that can draw upon multiple input service databases, construct an identity, and match to third party payers with the highest possible accuracy in object identification and minimal user interactions.

This research describes the object identification system framework that has been developed from a hybridization of several technologies, which compares the object's shared attributes in order to identify matching object. Methodologies and techniques from other fields, such as information retrieval, text correction, and data mining, are integrated to develop a framework to address the patient record matching problem. This research defines the quality of a match in multiple databases by using quality metrics, such as Precision, Recall, and F-measure etc, which are commonly used in Information Retrieval. The performance of resulting decision models are evaluated through extensive experiments and found to perform very well. The matching quality performance metrics, such as precision, recall, F-measure, and accuracy, are over 99%, ROC index are over 99.50% and mismatching rates are less than 0.18% for each model generated based on different data sets.

This research also includes a discussion of the problems in patient records matching; an overview of relevant literature for the record matching problem and extensive experimental evaluation of the methodologies, such as string similarity functions and machine learning that are utilized.

Finally, potential improvements and extensions to this work are also presented.

TABLES OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURES PAGE

1.0 INTRODUCTION

There is widespread agreement that the very nature of healthcare has been unalterably changed within the last fifty years [Dwivedi et al., 2002]. A major reason for this massive change in the nature of healthcare can be traced to the coming together of the twin revolutions of Information Technology and Telecommunications, revolutions which together have synergistically opened new vistas for healthcare.

Many health care systems have multiple legacy and information systems that support health care professionals for tasks such as patient record keeping, patient assessment and monitoring, care planning and diagnosis [Turley and Connelly, 1994; Pose and Czaja 1996] and also health care administration for tasks such as billings. Data are becoming more available from different resources. To facilitate data storage and retrieval, majority of useful data is stored in large databases under certain database management system (DBMS). However, these systems contain a great deal of redundant, summarized, and overlapping data objects that are often interdependent [Verykios et al., 2000 and Georgakopoulos et al., 1997]. The lack of a common data model, errors in data flows, errors during data entry, or situations where updates are not reflected into the database cause inconsistencies to arise. Kukich [Kukich 1992a; 1992b] found that the average error rate is 1-3% in typed data, 1-6% in optical character recognition (OCR) processed data, and 5-6% in data obtained by voice

communication, respectively. Today, these inconsistencies are common in systems and are the cause of significant revenue loss. Elmagarmid et al. [1996] reported that up to 25% of customer records are erroneous in a typical billing system.

One of these problems is that data objects can exist in multiple variations of patients' contacts or inconsistent text formats across multiple sources. For instance, a patient record is in a database as "Kate Simpson, Louisville, KY 40217" and as "Kate Simson, Louisville, KY 40217" in the other. This may cause duplicates in database systems and significantly increase the costs directly on mailing. In addition, such inconsistencies may cause incorrect patient records linkage. Data quality problems block recording of real-world objects correctly and have been fully realized. However, locating matches across a pair of lists not having unique identifiers such as social security number is often difficult. Typically available identifiers such as first name, last name, date of birth, gender and address components may not uniquely identify matches because of legitimate variations [Winkler 1990, 1995, 1999]. Numerous research efforts have been directed at the problem of record linkage or matching. This dissertation presents an overview of record linkage, especially on approximate matching, and proposes a solution using a hybridization of several technologies to address patient record matching issue.

Each patient record in databases typically contains last and first name, gender, date of birth, health insurance code (HIC) and the other attributes. Additionally, there are some variations or typographic errors in these attributes of records in databases. In

order to identify whether a pair of records refers to the same entity (records matching or not) in databases, each corresponding attribute of these pairs of records must be compared. By using an exact matching methodology, a pair of records is matched only if each corresponding attribute is the same character by character. However, a number of matched records may be missed due to variations and typographic errors in the attributes of the records. In order to circumvent this problem, in this research approximate matching methodology is proposed. The approximate matching relies on basic quantitative comparisons between corresponding attributes of a pair of records. Various string comparators are applied and evaluated to quantity the similarity of the elements of a pair of records. The matching decision can be made based on the overall similarity of a pair of records.

Two decision-making approaches are proposed in this research. One is a multiple valued logic approach – that uses fuzzy set theory. Multiple valued logic relies on quantitative similarities of each attribute of a pair of records, membership functions to qualitatively describe the overall favorability of a match, and an inference engine that aggregates conditional rules to reach a generalized conclusion on the match. The other approach is a machine learning approach. The records matching problem can be viewed as a pattern classification problem. Predictive models, such as decision tree induction, neural networks, and clustering can be applied to record matching problems [Gu et al., 2004]. Decision tree induction, supervised learning, is adapted in this research to address the patient records matching problem. Decision tree predictive models are constructed by learning the classification pattern in a training

data set in which the matching status, whether “matched” or “not-matched”, is known. Once constructed, the predictive models can be used to predict the class of each unclassified pattern.

This research project presents methodologies that can be applied to match patient records in multiple databases and eliminate duplicate records in a single database, which eventually is a process of data cleaning. Data cleaning problems are frequently encountered in many areas, such as knowledge discovery in databases, data warehousing, system integration, business intelligence, and risk management. So the framework developed in this research might be extended and applied to these fields to address real-world problems.

In the following sections, the objectives of the research are listed, related literature is reviewed, the methodology is described in details, extensive experimental evaluations are conducted, and the results of the performance and the other matching quality matrix are compared. The limitations, potential improvements and extensions of the current approach are discussed in the future work section.

2.0 OBJECTIVE

The problem of matching service recipients to third party payer eligibility can be stated succinctly: given a service recipient's record from a hospital database which of several, possibly numerous, similar records in an appropriate payer's database match that of the recipient? The above problem gives rise to several questions, as shown below:

1. Are some matches better than others?
2. Which is the best match?
3. Will drawing on several selected input sources increase the likelihood of matches?
4. Can a preferred "identity" be constructed from the input sources; and if so, how?

When the requirement is to link records, it should be possible to link them using a unique personal identification number. In many cases, however, encountered in practice, the identification number is neither unique nor error-free. In some of these cases, the evidence presented by identification codes, such as, primary key, object id, etc., may point out that the records correctly correspond or correspond to different or unknown identities [Verykios et al., 2003]. Therefore other methods such as the use

of last name, first name, date of birth, gender and address, have been necessary to identify different records relating to the same person. The methodology developed through this research is focused on matching patients' records when there is no unique identification matching number.

There are two approaches to record matching. The first one is called exact or deterministic and it is primarily used when there are unique identifiers for each record. Deterministic algorithms employ a set of rules based on exact agreement or disagreement results between corresponding fields in record pairs. The second approach to record matching is classified as approximate or probabilistic. Approximate methods commonly use likelihood scores calculated from rates of identifier agreement and disagreement among fields from potentially matched and not-matched records [Grannis et al., 2004, Verykios et al., 2003, Fellegi and Sunter, 1969]. The methods evaluated in this research, fall under the second category. The two principle steps in the record matching process are the searching of potential pairs of records, the searching step, and the decision whether a given pair is correctly matched, the matching step [Verykios et al. 2003]. For the searching step, the goal is to reduce the number of failures to bring linkable records together for comparison. For the matching step, the goal is to let the computer score the closeness of a match when some attributes of records match exactly and others do not. There is also a speed issue in the matching process because the searching step is computing intensive, especially when there are millions of records located in several databases.

Even though there are some commercial software packages, none of the existing software routines include a complete solution that uses all the available technology to solve the problem [Bell and Sethi 2001]. In this research, the methodology developed is focused on approximate matching in third party payer databases. Fuzzy logic and machine learning approaches are applied, as appropriate, for the patient records matching problem. Fuzzy logic is applied to use expert rules in the matching process and machine learning is used when training datasets are available. The main objectives of this research effort are to achieve the highest possible accuracy in object identification with minimal user interaction and provide an approximate matching tool that can draw upon multiple input service databases, construct an identity, and match to third party payers. The main objectives of this research are summarized below:

- To develop a framework and methodology for matching records;
- To create a set of matching rules;
- To improve matching quality metrics such as precision, recall, F-measure, accuracy, which are commonly used in Information Retrieval, and also to increase ROC index, and reduce mismatching rate.
- To generate evaluation tools to analyze string comparator functions and decision models;
- To compare performance metrics and matching results for different approaches;
- To design system architecture and integration to legacy system;

This research establishes a framework and methodology for patient records matching in third party payer databases. The methodology is extensively evaluated and validated using synthetic datasets. Performance evaluation tools are generated for the model comparison purpose.

3.0 LITERATURE REVIEW

There is widespread agreement that the very nature of healthcare has been unalterably changed within the last fifty years [Dwivedi et al 2002]. The cause of this massive revolution in the nature of healthcare can be traced to the coming together of the twin revolutions of Information Technology and Telecommunications, revolutions which together have synergistically opened new vistas for healthcare.

In response to rapidly increasing health care costs, and the need to improve the quality of health care, the decision makers in the federal government and private sector are promoting the utilization of health information technology (HIT) [Dhillon and Forducey 2006]. Healthcare enters the information age and professionals are finding an ever-growing role for computers in the daily practice of medicine and medical record management. Computers are used for research, education, medical record keeping, communications, reference resource and decision support amongst other. The amount of medical knowledge generated by clinical trials is rapidly growing, but that information is not being incorporated into practice with a satisfactory pace, which causes the emerging need for highest quality medical data management [Koncar M. and Gvozdanovic D. 2006]. At the2004 meeting of the World Health Care Congress, in Washington D.C., leaders of health insurance

companies, hospitals, pharmaceutical companies, and large employers, emphasized information technology's ability to improve the quality of care and reduce costs. In October 2004, President Bush stated that utilization of information technology is the most promising option for controlling health care costs [Dhillon and Forducey, 2006].

3.1 Impact of Information Technology on Healthcare

As a result of the rapid advancement and wide application of computer hardware, software and network, Healthcare Information Systems (HISs) have entered all hospitals and are becoming more important and covering more parts in daily hospital operations. Most functions in a HIS provide the users an easier and faster way of doing their medical tasks with graphic user interface [Sakamoto and Norihiro 1998].

The spectrum of potential applications of information technology to healthcare system is extraordinarily broad. The applications of information technology to the care of individual patients and groups of patients with an emphasis upon the interfaces that will be critical to enhance the quality of care, manage resources, enhance access, and control the rate of rise of health care costs [Shine 1996]. The modern Health Information System (HIS) may integrate three aspects: 1) patient data management, through an Electronic Patient Record (EPR); 2) medical decision support, though a Guideline Management System (GLMS); 3) organizational support, through a Workflow Management System (WfMS) [Ciccarese et al., 2005] . Medical errors can be reduced by the sharing of medical information and the correct application of medical information. A wealth of medical information exists in the

form of published medical algorithms. Application of such algorithms can generate information crucial to the clinical process [Johnson et al.]**.** The options could be considered for the development of a knowledge management tool. In fact, in a traditional information system there is no separation between information level and knowledge level and it is common that users adapt themselves to the system and vice-versa [Ciccarese et al. 2005].

3.1.1 Computerized Patient Records

Increased concern about the cost and quality of health care service delivery had led to dramatic changes in the organization of medical environments. Coupled with these changes is the increased deployment of computer and communication technologies within health care settings. In recent years with the emergence of new technologies and the advances in computer power, computers are increasingly being used by health care professionals for tasks such as patient record keeping, patient assessment and monitoring, care planning and diagnosis [Pose, and Czaja 1996, Truly and Connelly, 1994]

In 1991, the Institute of Medicine published a report on the computer-based patient record as essential technology for health care. The committee defined a computer-based patient record as "an electronic patient record by providing accessibility to complete and accurate data, alerts, reminders, clinical decision support system, links to medical knowledge, and other aids." [Shine 1996]. Computerized patient record

(CPR) is representation of a generally accepted belief that the paper record can no longer meet the demands of modern health care. Even clinicians who are not looking forward to change do understand much of the added potential of the CPR [Ginneken 2002]. A well-designed computer based patient record can be available to any authorized health care provider regardless of location. In addition to providing previous historical information about the patient, diagnoses, medications, and treatment parameters, such systems can alert the practitioner to allergies, idiosyncratic responses to treatment previously administered, and even include relevant citations in the medical literature that apply to the management of that particular individual [Shine 1996]. The advantages or expectations of computerized records are given by Kaihara [1998], are shown in Table 1.

Numerous publications explain the potential benefits of the CPR. Ginneken [2002] briefly summarized the (potential) benefits of CPRs as follows:

- Accessibility: computer-stored data can be viewed at multiple locations at all times. There are two forms of availability that are often mentioned separately, shared records and electronic data interchange (EDI)
- Readability: Scanned documents can be made available at multiple locations, but freehand may be difficult to read. Typed information, often acquired through transcription, is easy to read, but susceptible to errors.
- Reporting: Data in well-organized CPRs can be used to generate reports for institutional, regional or national repositories, and reduces the need for redundant recording.

- Completeness: computers can actively prompt for data. This is useful for improvement in the quality of data in CPRs, especially in the context of decision support, data analysis, and reporting

	Recipient of the benefit	Effect of CPR
Intra hospital	Patient care	Efficient access to the medical records Easy generation of required documents Report to other doctors for referral Various certification letters
	Hospital ad-ministration	Advanced information and decision support to doctors Small size of storage space Easy conversion to various documents such as Insurance claim Administration report Efficient analysis of the medical record for administration Easy access to his/her own medical record Efficient analysis of the medical record for clinical research
Intern Institutional	Patient care Administrators Researchers	Efficient data exchange among medical institutions Efficient collection and analysis of the medical record data for administration Efficient collection and analysis of the medical record data for clinical research

Table 1 Summary of Benefits of CPRs.

- Decision support: this is a broad area of functions that support diagnosis making and treatment policy, which often involve both assessment of health parameters, and treatment, which includes diagnostic support, treatment support, protocol support, and critiquing systems.
- Access to external knowledge sources: searches of databases with reference knowledge can be performed on the basis of CPR contents.
- Data analysis: the aforementioned benefits were mainly related to one particular patient. Research often involves data extractions beyond the

boundaries of one patient. Data analysis can be performed in the context of clinical research, but also for the purpose of quality assessment.

As with any information technology, the quality of the input data is extremely critical. In this regard information systems in departments of radiology, pharmacy, and clinical laboratories have developed at a rapid pace and can input individual patient record promptly and accurately [Shine 1996]. Collecting all data regarding one patient gives one the opportunity to present the diagnosis problem list, therapy, drugs and underlying symptoms in a comprehensive manner [Adelhard et al., 1995].

Lorence and Churchill [2005] showed that as early as 1991, consideration of the various barriers to CPR development, the interest and resources of individuals and organizations able to effect change, and the concerns of individuals who would be affected by implementation of CPRs prompted a national U.S. summit to identify eight critical activities to help advance CPR development: 1) identification and understanding of CPR design requirements; 2) development of standards; 3) CPR and CPR systems research and development; 4) demonstration of effectiveness, costs and benefits of CPR systems; 5) reduction of legal constraints for CPR uses as well as enhancement of legal protection for patients; 6) coordination of resources and support for CPR development and diffusion; 7) coordination of information and resources for secondary patient record databases; 8) education and training of developers and users. Despite these recommendations, relatively few advances have taken place in any of

these areas. What is needed is positive action to bring the U.S. up to the level of computerization currently existing in most of the developed world.

In order to harvest the benefit from the CPR, Ginneken [2002] discussed the efforts are required. A significant portion of these requirements can roughly be divided into requirements related to consultation of records and requirements related to contents. Other requirements are more related to the barriers for actual implantation of a CPR.

3.1.2 Population – based Information Technology

Rapid changes in technology and in the health care delivery system now allow for better attention to the health status and management of populations of patients. With health care information systems, computerized patient record (CPR) and internet, it is now feasible for a physician to identify all of the patients with high blood pressure within the practice to determine which ones have good, fair, or inadequate hypertension control, and relate this is the therapeutic programs being used. In this way, the physician learns from the population of hypertension, diabetics, cardiac, or others within the practice about what is working and what does not work in the specific practice [Shine 1996].

Bortolan [2000] performed a study of the influence of gender and age on QT-dispersion. A population based (aged from 65 to 85 years old) ECG database has been used. Three groups have been identified by clinical data: healthy subjects, patients

with hypertension and with cardiac diseases. Two QT-dispersion indices have been considered, and analyzing various subgroups, the influence of gender and age has been investigated.

Advancements in telecommunication, computers, networking, and information technologies present opportunities for analyzing the results of population-based disease screening. Tishelman et al.[2002] performed a study on population-based cervical cancer screening, used the model for quality of care systematically developed by Wilde et al. applied information technology and statistical data analysis, and found that generally high perceptions of quality of care, with particularly high ratings of perceived gynecological knowledge and medical information provision. Low perceptions of quality were found regarding several aspects of psychosocial care.

Shine [1996] also discussed that managed care systems require substantial information with regard to utilization of services, costs, and revenues. The population of patients now extends beyond that of an individual's practice to many thousands or ten of thousands of patients in a managed care system. Information systems are essential to the management of these entities. Moreover, most of these managed care organizations operate on fixed annual budgets so that activities that promote health and prevent disease are economically advantageous. The improvement in prenatal care, immunizations, appropriate application of mammography or cervical smears for early detections of cancer becomes cost-effective interventions to the organization.

3.1.3 Clinical Decision Making

A critical element for the future of America's health care system is well-informed patient/doctor joint decision making. This concept applies not only to individual patients and individual doctors, but also to groups of patients and to groups of physicians [Shine 1996]. Almost half of the executives surveyed by Gartner in March 2001 indicated they planned to add clinical decision support, and most 60% planned to add physician order entry. Another study by McKinesy [McKinesy et al.] makes clear that these enhancements are taking priority: the increase for overall hospital spending on OT was to increase 6-7% per year though 2004, while clinical spending would grow 13-15% annually [Ball 2003].

Given the explosion of medical knowledge, clinical practitioners find it literally impossible to keep up-to-date in the latest information about diagnosis, prognosis, therapy and related health issues. This has prompted the need to provide means for clinicians to receive the relevant research-supported evidence necessary for safe, effective and efficient clinical decision making. The emergence of 'evidence-based medicine' is an attempt to address the information needs of physicians and other health care workers. The recent development of communication technologies such as decision support tools can facilitate such a project, providing means to deliver health evidence that is scientifically valid and up-to-date [Allen et al., 1998].

In computer-aided decision-making, it is necessary to recompile medical knowledge so that it could be represented and organized in computational forms in a computer. A computer system with the encoded domain knowledge can then, to some extent, emulate a physician's decision-making process by reasoning on the domain knowledge. In some cases, a decision-making process for some restricted domains may be well formulated by incorporating numerical schemes such as categorical models (e.g., using flow charts) or statistical models (e.g., using Bayes' theorem) [Jones et al., 1995]. Many decision classification systems applied to Medical Diagnosis problems have been reported in recent years. Such systems can generally be interpreted as comprising a knowledge base and a method of reasoning from that knowledge base. Generally, there are two basic approaches to compile the available knowledge base: 1) in the first approach, patient records are compiled as tables of probabilities and decisions are taken using Bayesian probability inference, clustering algorithms, or discriminate function analysis; 2) the alternative approach depends on the rule-based Expert System in which the knowledge base of facts and rules is compiled by questioning expert clinicians [Mohamed 1992].

Information systems that rapidly provide data with regard to the outcomes of care are critical to the physician who is involved in decision making so that the best advice is provided to the patient. Information is critical to the patient, who must understand the therapeutic options and risks, participate in a joint process of decision making [Shine 1996].

The traditional tools for clinical decision making are statistical analysis, and knowledge based systems. Neural network models, hybrid systems and hyper merge. Schmidtke et al**.** [1996] showed how to apply statistics for critical clinical decision making based on readings of pairs of implanted sensors. They showed that the clinical accuracy of in vivo glucose readings is significantly improved by using sensor pairs and applying a likelihood ratio test. When only those readings that pass the test are used for calibration, the likelihood of a clinically significant error is substantially reduced.

Hudson et al. [2000] discussed use of internet-based information. In the last few years, numerous clinical wet sites have appeared. Many biomedical databases are now publicly available on the web, the best known of which is the human genome databases. Another source of domain information is generated by listservers to which individuals with like interests subscribe. An example of a successful listserve is PediHeart that brings together pediatric cardiologists to discuss difficult cases and emerging issues in the field. While these listservers are narrative in nature, they offer sources of potentially valuable domain information.

3.1.4 Geography and Information

Nationally, the distribution of physicians and other health care providers is very uneven. Rural communities continue to have major deficiencies in such personnel, and small rural hospitals continue to close at a rapid rate [Shine 1996].

Despite the existence of a highly advanced medical care system in the United States, large segments of population living in rural and sparely populated regions of the U.S. continue to be denied access to adequate health care services because of the small numbers of primary and secondary health care facilities and personnel available to serve them [Dhillon and Forducey 2006].

Telemedicine is one example of such solutions. This technology is not only available so that a consultant can visualize a patient at a distant site, review laboratory radiologic data, and discuss the problem with both patient and provider at the remote location, but also eventually will allow actual supervision of procedures and treatments. With the spread of the internet and the addition of other technologies, it should be possible for a physician anywhere in the world to obtain the most recent information regarding diagnostic or therapeutic options or modalities [Shine 1996]. A great promise of telemedicine has been to help isolated or scattered populations gain access to health service [Martinez et. al., 2004], [Field 1996]. In industrialized countries, telemedicine has proven to be a good tool for enabling access to knowledge and allowing information exchange, and showing that it is possible to bring good quality healthcare to isolated communities [Martinez et. al., 2004], [Kyedar 2003]. Telemedicine can also (and must) be used to deliver healthcare to poor area in countries with scarce infrastructure and to developing countries [Martinez et. al., 2004], [Wootton 1997], [Wright 1997].

Zach [1996] summarized telemedicine goals as:

- Improve patient care.
- Improve access to health care for rural areas and underserved areas.
- Give physicians better access to tertiary consultation.
- Give physicians access to conduct remote examinations.
- Reduce health-care costs.
- Provide health care services of a physician or facility to a larger audience (larger geographic regions and populations).
- Reduce patient transfers to secondary and tertiary care centers.
- Build an atmosphere of managed-care at hospitals and health-care facilities**.**

Ingenerf [1999] discussed three different issues with respect to "telemedicine and terminology":

- Better manage the communication (exchange and integration) of electronic patient data between disparate organizations.
- Enable access to external literature and knowledge bases for integration with patient-data-driven decision-support systems.
- Support the collection and analysis of clinical processes and outcomes (medical research).

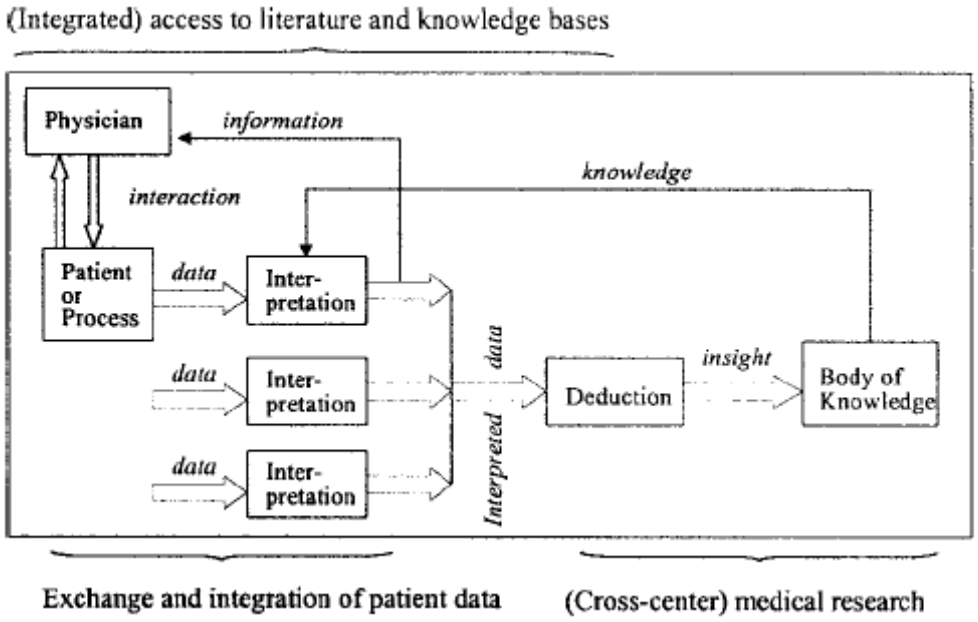


Figure 1 Data, Information and Knowledge [Ingenerf 1999]

Zach [1996] summarized barriers to telemedicine implementation as:

- **Infrastructure Planning and Development**: Failure by state policy makers to consider needs and solutions **across** the range of state activities (education, criminal justice, health and social services, etc.) can result not only in missed opportunities for capacity and cost sharing, but also can lead to costly redundancies and incompatibilities.
- **Telecommunications Regulation:** Limited competition for telecommunications services in rural areas and regulatory distortions created by arbitrary boundaries, such as Local Access and Transport Areas (LATAs), result in prohibitively high costs for transmission services needed to support high bandwidth applications like interactive video. In many rural communities,

prices for intra-LATA calls are unusually high and there is no local access to the Internet.

- **Reimbursement for Telemedicine Services:** Reimbursement policies for telemedicine services by HCFA, private insurers, and state Medicaid programs are currently limited and inconsistent.
- **Licensing and Credentialing**: Practitioners are understandably reluctant to use multi-state telemedicine networks because of the costs and administrative burdens of complying with multiple censure and credentialing rules compared to the expected frequency of network use.
- **Medical Malpractice Liability**: There is significant uncertainty regarding whether malpractice insurance policies cover services provided by telemedicine. Telemedicine networks that cross state lines create additional uncertainties regarding the state where a malpractice lawsuit may be litigated and the law that will be used.
- **Confidentiality**: Patients wary of electronic data may be reluctant to use telemedicine systems that result in the creation or transmission of confidential information Physicians and other health care practitioners with these perceptions may be reluctant to use electronic systems which they believe may increase the risk of breaching patient confidentiality.

Although there are some issues and barriers with respect to telemedicine, it is so important that it has still been applied to many fields in health care system. Stamford et al. [1999] have demonstrated that an ED-ED telemedicine consultation service can

be quite useful. In roughly one-fourth of the teleradiology cases, results show that either the diagnosis and/or the treatment is altered. Although the total number of telediagnostic cases was small, they found that in 18% of the cases, diagnosis was changed, and in over 50% of the cases, treatment was changed. Even when care was not modified, confirmatory advice supported delivery of care at higher confidence levels. In about one-third of the cases, decisions on patient transfer were altered. From these changes, they concluded that telemedicine can make a significant improvement in health care provided at remote EDs.

Takizawa et al. [2001] demonstrated using their new integrated telemedicine diagnosis-treatment system of a spiral CT unit and telecommunication equipment; they were able to provide medical examination and early detection of lung cancer through mass screening of the population. They have also provided early diagnosis and treatment of sport-related injuries and home-based medical treatment of elderly people. They have also used the system to provide medical services to rural areas, as telemedicine support at remote area, wintertime telemedicine support to an international sports competition, and various medical services to a home-care facility.

Choi et al. [2006] concluded that telemedicine is the future of health care. Recent implications in hospitals and future trends show that technology in health care leads to better services, while also demonstrating the potential to improve the lives of health care professionals and make transactions more efficient. The implementation

of current standards and continued progress in the development of standards will make a permanent and long-lasting positive effect on the health care industries.

3.1.5 Confidentiality and Accuracy

Shine [1996] summarized that although the health science will benefit from many of the advances in information technology that are applied to a wide variety of research areas, information technology is of particular importance to health care delivery. Developments of computerized patient records will enhance the efficiency, effectiveness, and distribution of health care. While health care system benefits from advanced information technologies, such as computerized patient records(CPRs), confidentially becomes critical not only because of the privileged nature of the physician/patient relationship, which must proceed in an uninhibited and trusting manner, but also because of the potentially perverse use of such information, which can be made by insurers or employees.

Kaihara [1998] reviewed the security issues related to CPRs. CPRs can be used in many different situations for different purpose. Security issues are present in almost every situation in different perspectives. The security issues which emerge in the various uses of CPRs are summarized in Table 2. He also divided the security issues into three categories.

- Security issues related to intrahospital use of CPRs: even when a CPR is used only inside a hospital, there are security issues. The main issues are two, namely, access control and prevention of outside intrusion.
- Security issues related to interinstitutional use of CPRs: patient data re-transmitted from a hospital to another hospital or a clinic, when a patient is referred to another

	Security related issues	**Examples**	**Available technology**
Intra hospital use	Access control Outside intrusion	How to prevent file access by unauthorized persons How to prevent access by outside intruders	Password Firewall
Inter hospital use	Integrity Leakage Authentication	How to prove the integrity of data Sent via the network How to prevent leakage of data during transmission How to prevent reception of transmitted data by unauthorized persons	Message digest Encryption Public and private keys
Storage	Identity Modification Long term storage Destruction, loss, theft etc Readability	How to guarantee the stored data are the same as the original How to prevent or detect modification after data are stored. How to guarantee safe storage in the long term. How to prevent loss, destruction of theft of data stored in a small size device How to guarantee the long term readability to stored data in the light of changing technology	Digital signature Digital signature Frequent reading the archive and the long-term planning Physical security

Table 2 Summaries of the security issues related to CPRS [Kaihara 1998]

medical institution. For the transmission, electronic mails of file transfer may be used. Since the data are highly confidential data, is the security of electronic mail of the transfer sufficient? If we want to use a more secure

method, then there seem to be two ways to solve this problem. The first one is to establish a computer network only the medical doctors, called Intranet Approach. The second way is the use of encryption for the communication, called Encryption Approach.

- Security issues related to storage of electronic data: regarding the security of electromagnetically stored data, there remain many interesting but still unsolved problems. The main issues of security in storage are how to guarantee that the stored data are identical to the original data, how to prevent or detect the modification of stored data and how to prevent the loss or destruction of the data.

In order to maintain the confidentiality of patient information, Quantin et al. [1998] proposed a computerized record hash coding and linkage procedure, which allows the chaining of medical information within the framework of epidemiological follow-up. Before their extraction, files are rendered anonymous using one-way hash coding based on the standard hash algorithm (SHA) function. Once rendered anonymous, the linkage of patient information can be accomplished by means of a statistical model, taking into account several identification variables. The security of the information hashed only once must then be ensured while it is transmitted to the recipient in order to avoid a dictionary attack by another sender. In particular, a network transmission should be secured by using a reversible encryption method.

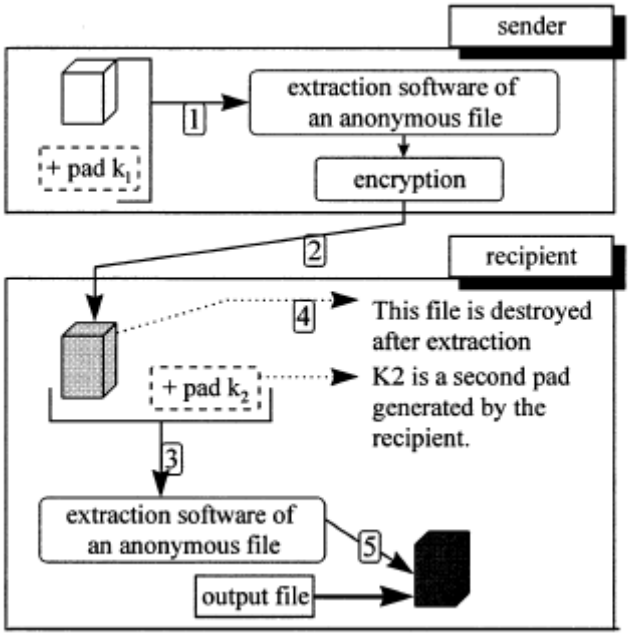


Figure 2 Outline of Hashing Procedure. [Quantin et al., 1998]

As technology leaps forward, no problem in developing information for health care systems exceeds the concerns related to confidentiality and accuracy of data. It is clear that individual rights and privacy must be protected. It is also critical that information be accurate. The potential harm to a patient of incorrect data might well be life-threatening. Regular opportunities for assessment of this information by the patient and/or the patient's primary provider as well as a methodology for rapidly correcting inaccuracies are essential if these information systems are to minimize potential adverse outcomes [Shine 1998].

We believe a fully integrated health information infrastructure holds the power to transform our health system. This transformation involves multiple dimensions, reaching into clinical research and out across healthcare delivery. It will require

information that encompasses the continuum of care, from the hospital to the enterprise, and on to include long term and home care [Ball 2003].

Although the health sciences will benefit from many of the advances in information technology that are applied to a wide variety of research areas, information technology is of particular importance to health care delivery. Developments of computerized patient records will enhance the efficiency, effectiveness, and distribution of health care.

However, healthcare systems today suffer from common problems of inefficient data management, uncontrolled resources spending, insufficient collaboration between various points of care, and inequity in access to high quality care [Koncar and Gvozdanovic 2006]. Some of the challenges in getting the correct patient information – both demographic and financial – are common among healthcare organizations. Others vary according to the size and location of the health system [Bell 2006].

3.2 The Patient Record Matching Problem

Several different specialized computer systems are used to deal with the various aspects of clinical practice today. These systems are tailored to the tasks they perform. This dedicated use ensures that the tasks are accomplished efficiently, but a large amount of technical and organizational adaptation is required for the data to be exchanged between systems. There are normally two types of systems in hospitals:

administrative patient management systems and those systems that handle medical and scientific patient data. [Sachs et al. 2000]. Although clinical and administrative databases serve their respective functions, linking records from the two types of databases serves valuable purposes and provides greater richness and potential benefit than either system separately. Such database "mergers", for example, can guide healthcare planning and resource utilization and lead to new discoveries related to incidence of disease, case-finding, risk or cause of death, expenditures, or ways to improve quality of healthcare without increasing costs [Weiner et al., 2003].

However, databases frequently contain approximately duplicate (but not identical) fields and records that refer to the same real-world entity, as illustrated by the following example:

A hospital has a database with thousands of patient records. Every year it receives new patient data from other sources, such as the government or local organizations. It is important for the hospital to link the records in its own database with the data from other sources. However, usually the same information (e.g., name, SSN, address, telephone number) can be represented in different formats. For instance, a patient name can be represented as "Dan Ford" or "Ford, D." or other forms. In addition, there could be typos in the data [Jin et al., 2003].

Bell and Sethi [2001] discussed two instances where matching must be take place. The first is when a new record is entered into the system. This record needs to be

correctly linked to any existing patient data. An effective matching algorithm must be able to handle comparison of records within a database of several hundred million records. When a new record is inserted into the database, automatic matching will be necessary. The second requirement for matching is for retrieval of patient data from a query to the system. Query results must be provided to the user without noticeable delay of operation (within two seconds). An effective matching algorithm should be able to properly match records containing fields that contain equivalents that are not exact matches. Examples of this include cases of two-letter designations for states or the use of St. for Street, and so forth.

A typical patient record has fields that can be parsed from encounter data. The fields are likely to include: first name, last name, address, telephone number, social security number, gender and data of birth [Bell and Sethi 2001]. Patients are entered into the system and identified on the basis of personal details provided by the patients themselves. In the optimal case patients are identified by referring to their ID-card. However, in some cases (e.g. loss of card, accident) the identification process has to rely on conventional techniques. This is done either orally or from a registration from. Registration in this way is by nature extremely susceptible to error because of the vagaries of human communication [Sachs et al., 2000].

Patient data in a hospital can come from various sources. Most hospitals have a central admission and discharge office, but only for admitted patients. Outpatients are entered in the outpatient clinics [Sachs et al., 2000]. Some data comes from the

electronic input such as Health Care Financing Administration forms, laboratory results, or pharmacy reports. There are a variety of input errors, including phonetic errors, incorrect data entry (added spaces, missing spaces, invalid characters), and reversal of first and last name (especially in Asian names) are common. Fields are sometimes misused and address either run on or are broken at inappropriate places. Nicknames, abbreviation, encoded information (such as 1= married, 2=single), and the use of Jr., Sr., III, and hyphenated names may lead to mismatches. Valid changes of address, married status, in addition to name changes can cause matching to be missed. Fraud and missing data are also a potential problem [Bell and Sethi 2001].

Where information is provided orally, the phonetic characteristics of a name can be misheard, while errors with written forms can occur because of legibility problems or spelling mistakes [Sachs et al. 2000]. Bell [2006] found that Minneapolis-based health system, like many large health systems across the country, used to receive hundreds of return mails items per month, most of which were patient bills that had not found their home. Sometimes, the address would be off by a single digit, whether in the street address or the patient's zip code. Other times, the address was so completely off base that it left patient financial services employees wondering whether the patient had intentionally supplied an incorrect address in hopes of eluding payment.

It is very important to choose identifiers for record linkage. Quantin et al. [2004] found that linkage using fewer informative identifiers could lead to linkage errors, it

is essential to quantify the information associated with each identifier. The aim of their study was to estimate the discriminating power of different identifiers susceptible to be used in a record linkage process. They showed the interest of three identifiers when linking data concerning a same patient using an automatic procedure based on the method proposed by Jaro; in this method, the date of birth, the first and the last names seemed to be the more appropriate identifiers. Including a poorly discriminating identifier like gender did not improve the results. Moreover, adding a second Christian name, often missing, increased linkage errors. On the contrary, it seemed that using a phonetic treatment adapted to the French language could improve the results of linkage in comparison to the Soundex. However, whatever the method used it seems necessary to improve the quality of identifier collection as it could greatly influence linkage results.

Weiner et al. [2003] discussed that although a social security number (SSN) might appear to represent a unique, nation-level identifier, inaccuracies occur in practice, due to patients' use of relatives' SSN for insurance-related or administrative purposes. In addition, errors, missing data, and other factors decrease the validity of this identifier as a MPI. Medicare beneficiaries receive unique identifiers, but many patients are not Medicare beneficiaries, and the identifiers may not be easily memorized by patients or available in all databases. These numbers, too, may be used by dependents in some circumstances. In fact, whenever individuals use other individual's identifiers as their own - whether intentionally, unintentionally, appropriately, or inappropriately - the identifier becomes potentially invalid, and this

is generally the case without biometrics or secret, strong passwords or tokens as identifiers. The cost of maintaining a MPI and the predominant use of paper-based records in delivering healthcare further hinder effective development and use of accurate indices.

Quantin et al. [1998] discussed that the aim of patient record linkage is to gather all information coming from different sources and concerning the same patient. Two types of linkage errors are of concern: erroneous link of notifications from two distinct patients, also called homonym errors, and failure to link multiple notifications on the same patient, also called synonym errors. Moreover, the linkage takes into account several identification available such as, for example, first and last names, data of birth, gender and zip code. However, some variables provide more information and more reliability than others.

Weiner et al. [2003] also showed that the Medicare program uses unique health insurance claim (HIC) numbers to identify its records. The HIC number corresponds to a beneficiary, rather than to a claim as the name might imply. Unique suffixes appended to HIC numbers are meant to distinguish categories of people, such as spouses or dependents, who may be covered under a single policy, based on rather complex rules established by the US Social Security Administration. This may cause difficulties when inconsistencies in these numbers cause analysts to attribute a health encounter erroneously to a beneficiary who is not the patient. HIC numbers used by patients for insurance purposes may also change, such as when a dependent spouse

remarries, for example. For their study, Medicare data were obtained from and in collaboration with Health Care Excel, Incorporated, and Medicare's Quality Improvement Organization for Indiana, after CMS approved a contractual agreement between Health Care Excel and the Regenstrief Institute. Data from CMS's separate Medicare beneficiary (N = 967,917 records) and claims (N = 596,105) files were used. The beneficiary file is actually a social security file that contains information about Medicare coverage. The CMS claims file represented all paid inpatient hospital claims for all 170,610 beneficiaries in central Indiana and the two outlying rural Indiana counties studied. Medicare part A data were assessed, but also included were some claims for outpatient surgery, which are covered under part A if the center where surgery was performed is operated by a hospital and its administrators elect part A coverage for its services. Health maintenance organizations and other risk-based programs are not represented in the data.

3.3 Potential Matching Methods

Efforts to consolidate patient data coming from heterogeneous databases with different identification schemas have a long history. Automated methods for linking patient records have been described as early as the 1970's [Smith, and Newcombe 1975, Sachs et al., 2000]. A variety of algorithmic methods can be applied to the matching problem. String comparison methods use comparison of individual letters to determine matching fields [Bell and Sethi 2001].

Record matching or linking is the process of identifying records in data store that refer to the same real world entity or object. There are two types of record matching. The first one is called exact or deterministic and it is primarily used when there are unique identifiers for each record. The other type of matching is called approximate [Verykios et al., 2002].

3.3.1 String Comparison Methods

The problem of string matching that allows errors, called approximate string matching, is that of finding the text positions that match a pattern with up to *K* errors. The problem, in its most general form, is to find a text where a text given pattern occurs, allowing a limited number of "errors" in the matches. Each application uses a different model, which defines how different two strings are. The idea for this "distance" between strings is to make the distance small when one of the strings is likely to be an erroneous variant of the other under the error model in use [Navarro 2001].

Since field matching is to identify string values in attribute domains in databases, it is to identify semantically equivalent (identifying) attribute value in syntactically different representations. As in other application areas, the equivalency of two string values is modeled by their similarity degree in the range [0, 1], with one as equivalent, zero no similarity [Wei et al., 2006].

In the literature, field matching algorithms can be classified into three categories [Wei et al., 2006]: a) character-based, b) n-gram-based, and c) token-based algorithms.

a) Character-Based Field Matching

Although designed with different strategies, character-bases string matching takes strings as sequence of characters and compares two strings character by character. The distance *d (x, y)* between two strings *x* and *y* is used to measure their closeness or similarity. Navarro [Navarro 2001] discuss that the distance *d(x, y)* between two strings *x* and *y* is the minimal cost of a sequence of operations that transform *x* into *y* (and ∞ if no such sequence exists. The cost of a sequence of operations is the sum of the costs of the individual operations. In most applications, the set of possible operations is restricted to

- Insertion: $\delta(\varepsilon, a)$, i.e. inserting the letter *a*.
- Deletion: $\delta(\varepsilon, a)$, i.e. deleting the letter *a.*
- Substitution or Replacement: $\delta(a, b)$ for $a \neq b$ i.e. substituting *a* by *b*
- Transposition: $\delta(ab, ba)$ for $a \neq b$ i.e. swap the adjacent letters *a* and *b.*

The most commonly used distance functions in the literature (although there are many others.) are:

- The Hamming distance: Hamming distance is the number of positions with different characters between two words of the same length [Bell and Sethi 2001]. Amir et al. [2004] defined the Hamming distance as

(1) Let $a, b \in \sum$. Define

$$neq(a,b) = \begin{cases} 1, & if \quad a \neq b; \\ 0, & if \quad a = b. \end{cases}$$

(2) Let $X = x_0, x_1 \ldots x_{n-1}$ and $Y = y_0, y_1 \ldots y_{n-1}$ be two strings over alphabet $\sum$. Then the hamming distance between X and Y $(ham(X,Y))$ is defined as

$$(ham(X,Y)) = \sum_{i=0}^{n-1} neq(x_i, y_j)$$

The Hamming distance allows only substations. The distance is symmetric, and it is finite whenever $|x| = |y|$. In this case it holds $0 \leq d(x, y) \leq |x|$ [Navarro 2001].

- The Levenshtein distance: or edit distance allows insertion, deletions, and substitutions [Navarro 2001]. The Levenshtein distance measures the number of morphological changes required to make one string into the matching string (not necessarily of the same length) [Bell and Sethi 2001]. The distance is symmetric, it holds $0 \leq d(x, y) \leq \max(|x|, |y|)$ [Navarro 2001].

  For instance: quickly

  qucehkly

  A simple character-wise comparison suggests that all letters after the "u" are incorrect. However, the final three ("kly") appear correct, despite misalignment. The minimum string distance is 3. In fact, there are often multiple answers, because more than one minimum set of transformation may exist for the computed MSD. Each transformation is called an "alignment", and represents a possible explanation of the error made [MacKenzie and Soukoreff 2002].

  The edit distance of two strings can be denoted as ED (*str1, str2*). The Levenshtein algorithm is often applied by its normalized variation [Navarro 2001],

[Hernandez and Stolfo, 1995]. The most popular variation is to normalize the edit distance of two strings by their maximum length:

$$sim(s_1, s_2) = 1 - \frac{ED(s_1, s_2)}{\max(|s_1|, |s_2|)}$$

- Episode distance allows only insertions. In the literature the search problem in many cases is called "episode matching", since it model the case where a sequence of events is sought, where all of them must occur within a short period. This distance is not symmetric, and it may not be possible to convert *x* into *y* in this case. Hence, *d(x, y)* is $|y| - |x|$ or ∞ [Navarro 2001]
- The longest common subsequence distance allows only insertions and deletions. This distance refers to the fact that it measures the length of the longest paring of characters that can be made between both strings, so that the pairings respect the order of the letters. The distance is the number of unpaired characters. The distance is symmetric, and it holds *d(x, y)* is $0 \leq |y| + |x|$ [Navarro 2001].
- Jaro [1985] introduced a string comparator that accounts for insertions, deletions, and transpositions. The basic steps of this algorithm include computing the strings and the number of common characters in the two strings and the number of transpositions. Jaro's definition of "common" is that the agreeing character must be within the half of the length of the shorter string. Jaro's definition of transposition is that the character from one string is out of order with the corresponding common character from the other string. The string comparator value is given by the follow formula:

$$sim(s_1, s_2) = \frac{1}{3} * (\frac{N_{common}}{L_{s1}} + \frac{N_{common}}{L_{s2}} + 0.5 * \frac{N_{common} - N_{transposition}}{N_{common}})$$

Where s_1 and s_2 are the two strings to be compared, with lengths L_{s1} and L_{s2}, respectively. *Ncommon* and *Ntranspostions* are the numbers of common characters and transpositions.

- Winkler [1997] modified the original string comparator introduced by Jaro in the following three ways:

1. A weight of 0.3 is assigned to a 'similar' character when counting common characters. Winkler's model of similar characters includes those that may occur due to scanning errors ("1" versus "l") or key punch errors ("V" versus "B").
2. More weight is given to agreement at the beginning of a string. This is based on the observation that the fewest typographical errors occurs at the beginning of a string and the error rate then increase with character position through the string.
3. The string comparison value is adjusted if the strings are longer than six characters and more than half the characters beyond the first four agree.

b) N-Gram-Based Field Matching

N-grams are often regarded as one kind of language model [Lloyd-Thomas et al., 1995]. The use of bigrams or trigrams for comparison of field value can be used to measure closeness of match. Bigrams are the two-letters consecutive combinations while trigrams are the three-letter consecutive combinations within the test value. For example, the word "receive" has the trigrams "rec", "ece", "cei", "eiv", and "ive". A

count of the number of bigrams or trigrams that words have in common and do not have in common can provide a quality of match [Bell and Sethi 2001] [Pfeifer et al., 1996].

String matching using *n*-grams is based on following observation: if two strings are similar to each other then they share a large number of *q*-grams in common [Ukkonen, 1992]. The similarity degree of two strings is usually modeled as the ratio of common q-grams to the total number of distinct q-grams in two strings. Combined with database management systems, n-grams can be efficiently applied in string matching, as reported in Gravano et al. [2001]. N-gram-based methods employ the sequential substrings to consider the similarity of two strings. Positional *n*-grams consider the out-of-order or word transposing problems in string matching. But the *n*-gram is an inherent space expensive technique, with (*m*-*n*+1) n-grams for a string with length of m. High space consumption means high computational cost in database systems [Wei et al., 2006]. Phonetic matching can be achieved by translating each field value into an equivalent phonetic code and comparing the phonetic codes for matching [Bell and Sethi 2001].

a) Token-Based Field Matching

Wei et al. [2006] discussed that Token-based string matching considers strings as consequences or sets of words, as in $S_1 = \{w_1\ w_2\ ...w_m\}$ and $S_2 = \{w_1\ w_2\ ,...,\ w_m\}$ respectively; where each word as a sequence of characters as in a character-based ring matching. This adds great flexibility in string comparison for two obvious reasons:

1) Stop words and punctuation, which have no significant meanings in content representation, can be easily removed from strings, and only meaningful strings are left to compare;

2) Abbreviations can be taken care of by expressing words as sequences of characters, as in St → Street and U of L → University of Louisville

Navarro et al. [2003] addressed the importance and requirement of token-based field matching in resolving field matching problems. Wei et al [2006] summarized token-based field matching algorithms in the literature:

- The simplest token-based field matching algorithm is the Jaccard similarity metrics, which counts the number of common words N_c and the number of distinct words N_d of two strings in comparison, and take the ratio of N_c/N_d as the similarity degree of two strings. The simple field matching algorithm is very similar to the Jaccard metrics, in which the similarity degree is calculated by a simple formula:

$$sim(str_1, str_2) = \frac{N_c}{|str_1| + |str_2|} \times 2$$

where $|str_1|$ is the number of words in *str1*, and $|str_2|$ is the number of words in str_2. Given proper thresholds, these algorithms are capable to resolve some equivalence errors without introducing high false positives (e.g., irrelevant string values identified), but they generate a large number of true negatives (e.g., semantically equivalent string values unidentified).

- In Lee et al. [1999], an algorithm was proposed to improve the field matching accuracy. First, it sorts tokens in each attribute value in the selected domain, and then sorts records based on selected domain(s) to bring duplicate records as close as possible. The concept of Record Similarity (RS) was proposed to calculate the similarity of records by combining the similarity of tokens and similarity of field values. By defining some string matching patterns, this algorithm improved the field matching accuracy significantly. a token set S can be viewed as samples from an unknown distributions P_s of tokens, and a distance between S and T can be computed based on these distributions. Letting $KL(P \mid Q)$ be the Kullback-Lieber divergence and Letting $Q(w) = \frac{1}{2}(P_s(w) + P_T(w))$, the Jensen-Shannon distance between P_s and P_T is

$$Jensen - Shannon(S,T) = \frac{1}{2}(KL(P_S \mid Q) + KL(P_T \mid Q))$$

- A more sophisticated field matching algorithm was presented in Mongen and Elkan [1996, 1997], the recursive field matching algorithm. This algorithm recursively compares substrings of two strings str_A and str_B to obtain the maximum degree of similarity, and calculates the overall similarity by averaging the total similarity degrees, as shown in formula

$$sim(str_A, str_B) = \frac{1}{|str_A|}\sum_{i=1}^{|str_A|}\sum_{j=1}^{|str_B|} sim(str_{Ai}, str_{Bj})$$

In the recursive string matching algorithm, strings are considered as sequences of words, and best matching is achieved by comparing substrings iteratively.

3.3.2 Probability Methods

When pairs of records are brought together for comparison, a decision must be made as to whether these are to be regarded as linked, not linked, or possibly linked, depending upon the various agreements and disagreements of items of identifying information. For example, if we are linking patient records, a possible measurement would be to compare family names on two records, and assign the value of 1 for those pairs where there is an agreement and 0 for those pairs where there is a disagreement. These measurements will yield a vector of observations on each record pair [Verykios et al., 2000]. Once a field agreement (match) or field disagreement is determined, probabilities are used to measure the value of determination [Bell and Sethi 2001]. Most of the works proposed by statisticians have been influenced by the pioneering work of Fellegi and Sunter [1969]. The Fellegi Sunter probability approach uses the probability that a field agrees given the record pair examined is a matched pair (m probability). It also uses the probability that a field agrees given the record pair being examined is an unmatched field (u probability). The values of u are calculated by occurrence of field entries within a database. Thus, u needs to be recalculated after entry of new data or segments of new data. Guessing what the occurrence of error is in various fields approximates the value of m. The weight of matching field is computed as $\log_2$ (m/u) and the weight of a field disagreement is $\log_2$ (1-m)/(1-u). The composite weight of record matching is the sum of the weights for individual fields. The greater the composite weight, the greater is the probability the records are the same [Bell and Sethi 2001].

The overall probabilistic method for record linkage has been developed for more than 30 years and the original pioneers in this field are Fellegi and Sunder [1969], and Newcombe et al. [1959]. This method requires knowledge of the distribution of data in the existing database. Training data is used to develop the statistics associated with the incidence of expected matches and mismatches [Bell and Sethi 2001].

Fellegi and Sunter [1969] made the concepts introduced by Newcombe et al. [1959] rigorous by considering ratios of probabilities of the form:

$$R = P(\underline{x} \in X | M) / P(\underline{x} \in X | U)$$

where $\underline{x}$ is an arbitrary agreement pattern in the comparison space X. The theoretical decision rule is given by:

(a) If R >UPPER, then designate pair as link.

(b) If LOWER $\leq R \leq$ UPPER, then designate the pair as a possible link and hold for clerical review.

(c) If R <LOWER, then designate the pair as non-link.

The UPPER and LOWER cutoff thresholds are determined by a priori error bounds on false matches and false nonmatches. Fellegi and Sunter showed that the decision rule is optimal in the sense that for any pair of fixed upper bounds on the rates of false matches and false non-matches, the manual/clerical review region is minimized over all decision rules on the same comparison space X. If now, one considers the costs of the various actions, that might be taken, and the utilities associated with their possible outcomes, it is desirable to choose decision rules that will minimize the costs of the operation [Verykios et al., 2003]. With the Fellegi-Sunter algorithm, the results are

based on the input statistics regarding the probability of error (m value) that is not a known quantity and must be estimated for any given term [Bell and Sethi 2001].

3.3.3 The Other Methods

Besides the string comparison and probability methods for records approximate matching, which we have reviewed, there are some the other approaches. Bell and Sethi [2001] summarized those approaches as follow:

3.3.3.1 Neural Network

Cases can be fed into a neural network such that a weighted system of neurons can be developed to determine matching patient records from those that do not match. A series of training cases would be developed to train the neural network. Then the neural network would be tested using a series of test cases with known solutions. The neural network has the potential for development of nonobvious algorithms but has drawbacks of case specific applicability or wide generalizations. This method has the positive benefit that a conversion process of characters to numerical values that have true distance meaning is not required. The neural network can usually derive rule-based knowledge from training instances, and the quality of the neural network is largely determined by the quality of the training data. A set of real data that has been correctly evaluated for matches and mismatches has not presently been located.

3.3.3.2 Signal Processing

The use of signal processing could be implemented if the characters are converted into numerical values. Each character/symbol can be assigned a digital level (or analog level) that is related to the likeness of characters (keyboard location, similar sound or similar physical shape). For example, the letter “A” may be assigned a number one, the letter “B” may be assigned a two, and so on. This may require around 50 levels for numbers, letters, and some symbols. The levels will then correlate to an analog signal sampled over time. Thus, the level of the signal (analogy to voltage) is related to character symbol and the order of the level is related to placement (analogy to time). The individual fields or combined fields can then be correlated with same field types of other records. The level of correlation will correspond to the likeness of records. If the fields are separated and correlated, then a combined weighted function of correlation will be computed.

The new “signals” can be shifted in “time” and correlated. Variance of the matching records will be similar to signals with added noise. Signals can be converted to the frequency domain” by the fast Fourier transform or to the wavelet domain if working with signals in a different domain adds computational speed.

3.3.3.3 Clustering Approach

Another approach is clustering. Each entry in a field can be plotted in n-dimensional space and the distance between similar entries will be smaller than dissimilar entries. The values of entry distance for fields can be weighted to obtain an overall likeness value for records. A threshold value can be set for determination of merging of records. This method has the drawback that an entry in a field that has an

Comparison of matching techniques													
	Letter-Related Mismatches					Word- or Field-Related Mismatches							
	Transposition	Omission	Typing error	Misspelling(few letters)	Misread characters	Change of address	Change of name	Nicknames	Sex change	Fraud	Change of zip code	Change of phone number	Incorrect data entry
Damerau-Levenshtein	X		X	X	X								
Jaro Weighting	X												
Fellegi-Sunter SOUNDEX/ NYSIS		X	X	X	X		X						
Neural Network						X	X	X	X		X	X	
Correlation		X	X	X	X								
Rule-based Approach						X	X	X	X		X	X	
Fuzzy Logic Approach						X	X		X		X	X	
Clustering	X	X	X	X	X								
Matched Filter		X	X	X	X								

Table 3 Comparison of matching techniques

offset in character/symbol space (for example: an entry with the first character missing) will have a large distance determination. A space shift and distance recalculation could be performed to catch such errors.

3.3.3.4 A Rule-Based Approach

A rule-based approach would involve the use of heuristics associated with common errors based on similarity of letters, word sounds, letter proximity on keyboard, importance of name versus address, nicknames, abbreviations, date of encounter relation to age of patient, zip code relation to location, and so forth. Such an effort would incorporate the expertise involved in making a record comparison. An expert comparator would be able to differentiate between distinguishing features and probable errors.

3.3.3.5 Fuzzy Logic Approach

Fuzzy logic would provide for such options as mostly matches or almost doesn't match and other such conditional expressions. All records would be members of the sets matching records and mismatching records to varying degrees. Each of the fields would have membership in the sets of matching fields and mismatching fields to varying degrees. The rules could be made to state "if the last name matches poorly and the first name matches well and the address matches well and sex is female then ..." These types of rules could be made to make final determination of degree of membership of a recording in the set of matching records. Other measures of importance in matching such as such as length of name, commonness of name could also be used in the rules.

3.4 Existing Matching Algorithms

Bell and Sethi [2001] conducted a search of existing software for record matching. Table 4 outlines a comparison of the various commercial software packages. The ideal approach of comparing existing algorithms is to use a test database and test cases to measure type I and type II errors. A type I error occurs if an unmatched comparison is erroneously linked while a type II error occurs if a matched comparison is erroneously not linked [Gu et al. 2004; Winkler W. E. et al., 1990; Winkler and Thibaudeau 1990; Winkler 1995]. The speed of operation also needs to be considered in algorithms comparison. However an accepted set of test cases dose not exist nor have all the manufacturers agreed to participate in such a test. Thus, the best that can be done at this time is to compare the algorithms on a theoretical basis.

The U.S. Bureau of the Census has an algorithm based on Fellegi-Sunter that only works with simple flat databases [Bell and Sethi 2001]. Fellegi and Sunter [1969] gave a formal model for record linkage that involves optimal decision rules that divide a product space $A \times B$ of pairs of records from two files A and B into matches and nonmatches, denoted by M and U, respectively. The main issue is the accuracy of estimates of probability distributions used in a crucial likelihood ratio. When estimates are sufficiently accurate, decision rules are (nearly) optimal. The optimality is in the sense that, for fixed bounds of the proportions of false matches and false nonmatches, the size of the set of pairs on which no decision is made is minimized. The expectation-Maximization algorithm was used to obtain maximum likelihood

estimates [Winkler 1990; Winkler and Thibaudeau 1990]. The use of Fellegi-Sunter is beneficial due to the application of knowledge of the database entries and estimates on the validity of data in a field [Bell and Sethi 2001]. The Language Analysis Systems Company has a unique position in the matching analysis in that it has the only ethnic detection algorithm but it focuses only on names. The Search Software America Company does not use the Fellegi-Sunter model, and has a multi-step processes with the ability to put in rules for heuristics [Bell and Sethi 2001].

Existing matching software		
Software/Company	**Evaluation/Cost/Integration**	**Technology**
Advanced Linkage Technologies of America (ALTA)	Has exclusive licensee of software to LinkSoft Technologies, Inc., MPIscan	Fellegi-Sunter and fuzzy
U.S. Bureau of the Census	Has working version of software	Fellegi-Sunter
Care Data Systems	Matching algorithms is bundled with MPI	Statistical, configurable
DeNovo Technologies	Uses high-dimensional space and distributed representations for matching, bigram discussion; gone out of business	Unclear
Electronic Digital Documents	Short write-up on Web site, have demo version downloadable from Web, requires Informix database	Rule-based
Intelligent Search Ltd	Strong string comparison, API made to be integrated into systems	Rule-based
Group I SmartMatch	Not meant for MPI application	Rule-based
Language Analysis Systems	Names resolution software, uses cultural and rule-based analysis to rank name matches	Linguistic rules
LinkSoft Technologies	Has exclusive license with ALTA for the matching algorithm, claim to have enhanced ALTA algorithms	(See ALTA)
MatchWare (part of Vality Technology Inc.)	Has API for integration, international usage	Fellegi-Sunter and rule-based parsing
Search Software America	Uses multistep processing	Fuzzy Keys
Statistics Canada (GRLS)	Requires Oracle database, Unix server/ PC client	Fellegi-Sunter
University of Oxford (OX-LINK)	Used on National Health Register (60 million records)	Fellegi-Sunter

Table 4 Existing matching software

MatchWare is a commercial software product developed by MatchWare Technologies Inc. The capabilities of MatchWare were incorporated into the software product Integrity after Vality Technologies Inc acquired MatchWare in July 1999. MatchWare uses the Fellegi-Sunter model and rule-based parsing, and provides two services: standardization and matching. These services are available via a set of C function calls through MatchWare Callable Libraries. MatchWare provides the means for performing targeted searches and intelligent probabilistic matching through its Standardization and Matching Service. MatchWare provides standardization services by which name and address are broken into distinct components. These standardized components provide an efficient means of searching the database by using names and address information. In the case of Name Standardization, MatchWare removes commonly used words in business names like “The”, “Of”, “Inc”, “Company”, etc., that do not provide any additional information to establish the identity by name. MatchWare then converts the first five significant words of the remaining words of the name to their root form to eliminate variations in spelling or usage. Words like Robert, Robbie, Bob, Rob would all be converted to the same root word by this process. Finally, MatchWare provides the capability of converting those token to SOUNDEX, REVERSE SOUNDEX and NYSIS forms, which are based on the way a word sounds. These can compensate for spelling errors in a query when searching.

The matching capability is based on a string comparator, introduced by Jaro [Winkler 1985; Gu et al., 2004], that gives values of partial agreement between two strings. The string comparator accounts for length of strings and partially accounts for the

types of errors typically made in alphanumeric strings by human beings. It is used in adjusting exact agreement weights when two strings do not agree on a character-by-character basis.

Specifically, if $c > 0$, then the Jaro string comparator is

$$\Phi = W_1 \bullet c/d + W_2 \bullet c/r + W_t \bullet (c - \tau)/c,$$

Where

W_1 = weight associated with characters in the first of two files,

W_2 = weight associated with characters in the second of two files,

W_t = weight associated with transpositions,

d = length of string in first file,

r = length of string in second file,

τ = number of transpositions of characters, and

c = number of characters in common in pairs of strings.

If $c = 0$, then $\Phi = 0$.

Table 5 provides examples of string comparator values for pairs of last name and for pairs of first names. The abroms-abrams example with string comparator value .9333 in contrast to the lampley-campley with value .9048 shows that the string comparator gives higher values to the pair that differs by a single character further from the first position. The martha-marhta example with value .9667 in contrast to the jonathon-jonathan example with value .9583 shows that transposition of two characters causes less of a dwonweighting than differing by one [Winkler and Thibaudeau].

Gill [1997] described the major features of the Oxford record linkage system (OX-LINK), which uses Fellegi-Sunter model, with its use of the Oxford name comparison algorithm (ONCA), the calculation of the names weights, the use of

Shackeford	Shackelford	.9848
Cunningham	Cunnigham	.9833
Campell	Campbell	.9792
Nichleson	Nichulson	.9623
Massey	Massie	.9444
Abroms	Abrams	.9333
Galloway	Calloway	.9167
Lampley	Campley	.9048
Dixon	Dickson	.8533
Frederick	Fredrick	.9815
Michele	Michelle	.9792
Jesse	Jessie	.9722
Marhta	Martha	.9667
Jonathon	Jonathan	.9583
Julies	Juluis	.9333
Jeraldine	Geraldine	.9246
Yvette	Yevett	.9111
Tanya	Tonya	.8933
Dwayne	Duane	.8578

Table 5 Example of String Comparator Values for Various Pairs

orthogonal matrices to determine the threshold acceptance weights, and the use of combinational and heuristic algebraic algorithms to select the potential links between pairs of records.

Most matching software has string-matching algorithms, although they are different and often proprietary. Different software companies use blocking, phonetic tokenization, and fuzzy keys for partitioning of the database. There are many fields of interest of which about four are name fields. Care should be taken to properly analyze

the other fields as well. None of the existing software were routines include a complete solution that uses all available technology to solve the problem [Bell and Sethi 2001].

3.5 Fuzzy Set theory and Applications

Bezdek [1993] discussed the difference between a fuzzy set and crisp set. Fuzzy sets are a generalization of conventional set theory that were introduced by Zadeh in 1965 as a mathematical way to deal with vagueness related to the way people sense things (e.g., "big" versus "small") [Zadeh, 1965; Mahmoud and Venkateshwar 1988]. Fuzzy interpretations of data structures are a very natural and intuitively plausible way to formulate and solve various problems. Conventional (crisp) sets contain objects that satisfy precise properties required for membership. The set of numbers H from 6 to 8 is crisp; we write $H = \{r \in R \mid 6 \leq r \leq 8\}$. Equivalently, H is described by its membership (or characteristic, or indicator) function (MF), $m_H : R \rightarrow \{0,1\}$, defined as

$$m_H(r) = \begin{cases} 1 & 6 \leq r \leq 8 \\ 0 & otherwise \end{cases}$$

Since m_H maps all real numbers $r \in R$ onto the two points (0, 1), crisp sets correspond to two-valued logic: is or isn't, on or off, black or white, 1 or 0. In logic, values of m_H are called truth values with reference to the question, "Is r in H?" the answer is yes if and only if $m_H = 1$; otherwise, no.

Consider next the set F of real numbers that are close to 7. Since the property "close to 7" is fuzzy, there is not a unique membership function for *F*. Rather, the modeler must decide, based on the potential application and properties desired for *F*, what m_F should be. Properties that might seem plausible for this F include (i) normality (MF(7) = 1), (ii) monotonicity (the closer *r* is to 7, the closer $m_F(r)$ is to 1, and conversely), and (iii) symmetry (numbers equally far left and right of 7 should have equal memberships). One of the biggest differences between crisp and fuzzy sets is that the former always have unique MFs, whereas every fuzzy set has an infinite number of MFs that may represent it. This is at once both a weakness and a strength; uniqueness is sacrificed, but this is gives a concomitant gain in terms of flexibility, enabling fuzzy models to be "adjusted" for maximum utility in a given situation.

García Valdez and Flores-Fonseca described the design and implementation of an Inference Engine for the execution of Fuzzy Inference Systems (FISs). And discussed that Fuzzy production rules use fuzzy logic sets to characterize the variables and terms used in the propositions of the rules. Fuzzy production rules or fuzzy IF-THEN rules are expressions of the form IF *antecedent* THEN *consequent*, where the antecedent is a proposition of the form "*x* is A" where *x* is a linguistic variable and A is a linguistic term. The truth value of this proposition is based on the matching degree between *x* and A. Propositions are connected by AND, OR and NOT operators. Some implementations of fuzzy rule-based systems also include other kinds of data types in their propositions, for example the FLOPS system includes fuzzy numbers, hedges, and non fuzzy data types (integers, strings and float) [Siler and Buckley

2005], Depending on the form of the consequent, two main types of fuzzy production systems are distinguished:

- Linguistic fuzzy model: where both the antecedent and consequent are fuzzy propositions.
- Takagi-Sugeno fuzzy model: the antecedent is a fuzzy proposition; the consequent is a crisp function.

Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic. The mapping then provides a basis for decision-making. The process of fuzzy inference involves all of the pieces like membership functions, fuzzy logic operators, and if-then rules. There are two types of fuzzy inference systems that can be implemented in the Matlab Fuzzy Logic Toolbox: Mamdanitype and Sugeno-type. In the purposed scheme Mamdani's fuzzy inference method has been used [Saini 2004]

García Valdez and Flores-Fonseca [García-Valdez and Flores-Fonseca] also discussed three main inference systems:

Tsakumoto: The output is the average of the weights of each rule numeric output, induced by the degree of support of each rule, the min-max or min-product with the antecedent and the membership functions of the output. The membership functions used in this method must be non-decrease monotonic.

Mamdani: The output is calculated by applying the min-max operator to the fuzzy output (each equal to the minimum support degree and the membership function of the rule). Several schemes have been proposed to choose the numeric output based on the fuzzy output; these include the centroid area, area bisection, maximum mean, maximum criteria.

Sugeno: The fuzzy production rules are used. The output of each rule is a linear combination of the input variables plus a constant term, and the output is the average of the support degree of each rule.

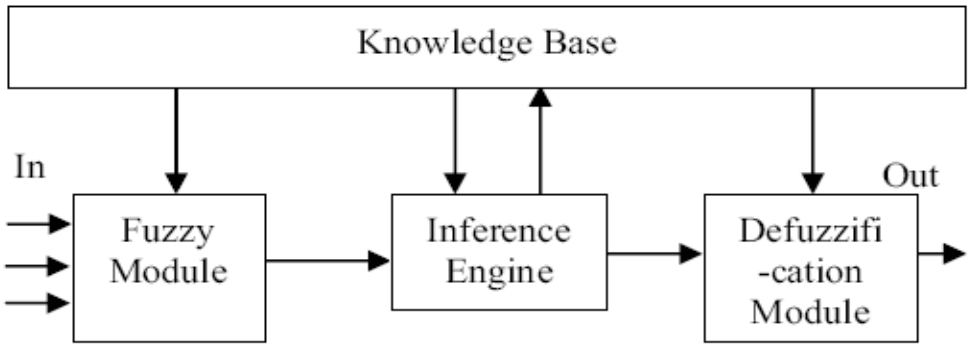


Figure 3 Fuzzy Inference System Model

Saini discussed more details that Mamdani's fuzzy inference method is the most commonly seen fuzzy methodology. It expects the output membership functions to be fuzzy sets. After the aggregation process, there is a fuzzy set for each output variable that needs defuzzification. It's possible, and in many cases much more efficient, to use a single spike as the output membership functions rather than a distributed fuzzy set. It enhances the efficiency of the defuzzification process because it greatly simplifies the computation required by the more general Mamdani method. The fuzzy model contains following four modules as shown in Figure. 3.

- Fuzzification Module (FM)
- Inference Engine (IE)
- Knowledge Base (KB)
- Defuzzification Module (DM)

The Fuzzification module transforms the crisp inputs into fuzzy values. Then these values are processed in the fuzzy domain by Inference Engine, which, is based on the rule base provided by the Knowledge base (KB). Here some appropriate fuzzy operators are also applied, implication processes are done and outputs are aggregated. Finally at last stage, the Defuzzification Module (DM) maps the fuzzified domain values into corresponding defuzzified domain crisp values.

Manzoul and Rao [1988] discussed fuzzy inference that each linguistic rule is expressed as a set of fuzzy relations, (equals the number of inputs) from each of the input universe of discourse (antecedent) to the output universe of discourse (resultant action). Inference is made from the given observations, and the rules, based on the compositional rule of inference.

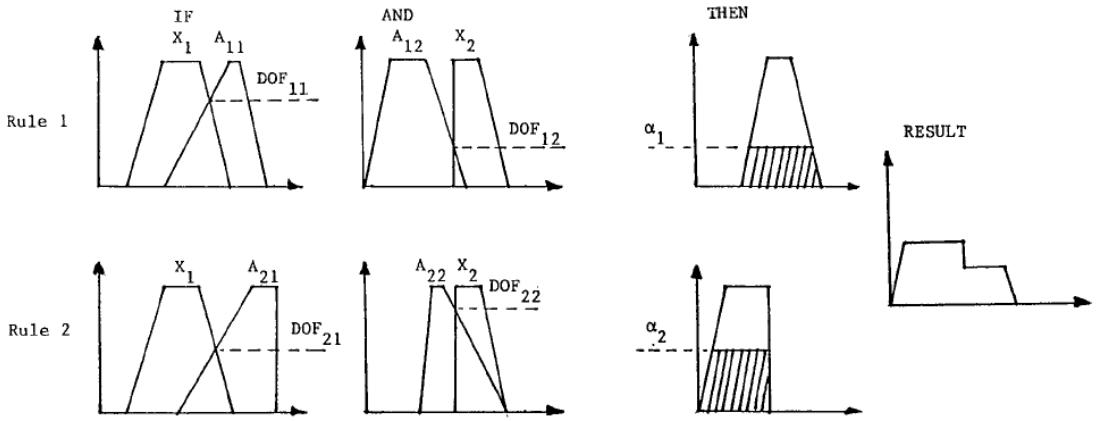


Figure 4 Fuzzy Inference Mechanisms

Fuzzy inference has been successfully implemented in some rule-based control systems such as the primitive fuel engine, and the cement kiln process. Rule-based expert systems, such as CATS, have successfully employed fuzzy inference. Fuzzy inference has also been proposed for real-time decision making in areas of command and control. Schneider et al. [2001] introduced a new heuristic technique that is based on fuzzy logic to understand words after any OCR software system generated the proper database. It is assumed that a document is scanned by some OCR system, and the result was put in a database. The algorithm matches the noisy strings in the database against an existing dictionary.

Buche et al. [2005] proposed a new system to query a relational database which retrieves the most relevant heterogeneously structured information according the most relevant heterogeneously the query. They have made the choice first to query the database using fuzzy values expressing the user's preferences and on the other hand to represent imprecise data stored in the database as possibility distributions. In this context, the central point of their work is to propose a way of helping any user to make a fuzzy query to be performed on a complex database schema.

Chen et al. [2002] developed a new technique to support the query relaxation in biological database. Query relaxation is required due to the fact that queries tend not to be expressed exactly by the users, especially in scientific database such as biological database, in which complex domain knowledge is heavily involved. To

treat this problem, they proposed the concept of the so called fuzzy equivalence classes to capture important kinds of domain knowledge that is used to relax queries.

Martin-Bautista et al. [2002] discussed the role profiles using fuzzy logic in web retrieval process. It is possible to consider the use of soft computing, e.g., linguistic qualifiers for computing with words, to help retrieval. Soft computing is seen to aid both the indexing and the querying processes, as well as the matching of a document to a query. Web mining process can be carried out by means of fuzzy clustering. Fuzzy inference can be used in order to modify queries and extract knowledge from profiles with marketing purpose within a web framework.

Kraft et al. [2000] presented an integrated approach to information retrieval which combines some techniques of fuzzy clustering and fuzzy inference in order to achieve optimal retrieval performance. To capture the relationship among index terms, fuzzy logic rules are used. They adapt several fuzzy clustering methods to the task of clustering documents with respect to the index terms.

4.0 METHODOLOGY

The problem of matching service recipients to Medicare eligibility still exists in health care providers. In this research, approximate matching in third-party payer databases is been focused on, and a framework is attempted to be developed using a hybridization of several technologies. This framework, as illustrated in Figure 5, is designed to accept users' repeated calls to third party payer databases on the basis of HIC number (Health Insurance Code), last name, first name, date of birth and gender, and to declare a result along with, potentially, a list of candidates. As the search stage is executed, the potential matches are included in the candidate pool.

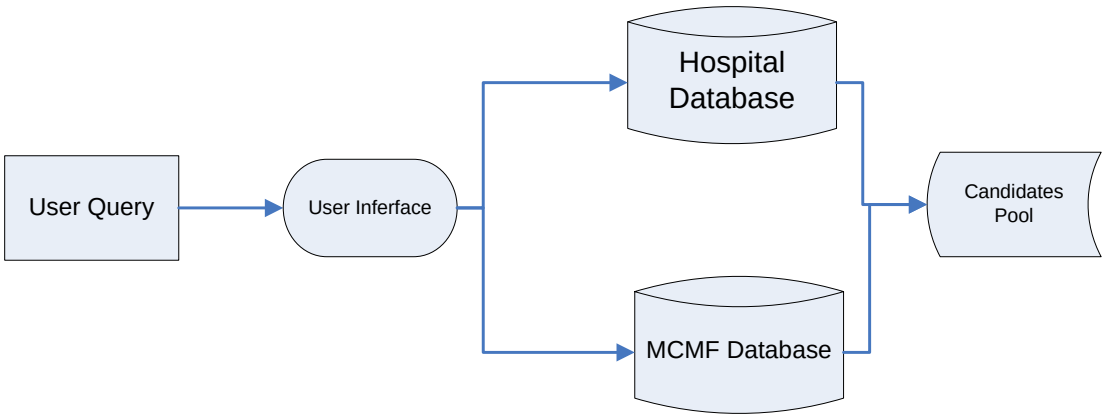


Figure 5 Patient Records Matching Application

During the matching process, the matching algorithm compares all the candidates to the presented record in the query and assigns each candidate a score that defines the

degree of similarity. Then matching logic rules are applied to identify the best matches.

4.1 Framework for the Matching Process

The process that is designed for patient records matching problem is illustrated in Figure 6 and described below:

- Parsing: after users submit a query to the databases, the first step in the matching process is to parse the fields or attributes from the record in the database. For example, the attributes could be health insurance code (HIC) number, last name, first name, date of birth and gender.
- Transformation: Included in the framework is a set of general domain-independent transformation functions to resolve the different text formats of

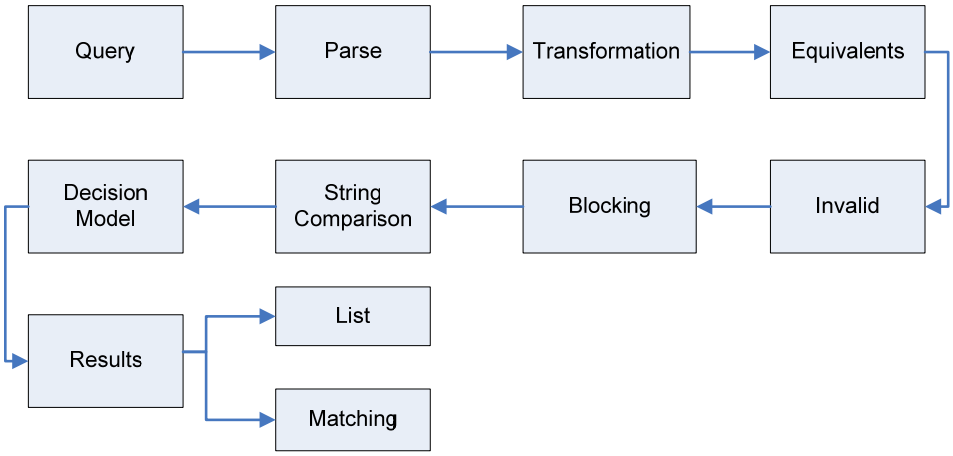


Figure 6 The Process of Record Matching [Bell and Sethi 2001].

attributes or fields in the records. For example, abbreviation transformation

replaces token with corresponding abbreviation (e.g., Blvd, Boulevard), and Soundex coverts a token into a Soundex code. Tokens that sound similar have the same code etc. The transformation functions are applied between sets of attribute values individually, i.e. first name with first name, HIC number with HIC number (health insurance code).

- Equivalents: A lookup table for equivalent names can be applied to help avoid not matching records when an equivalent name is used. The first name can be looked up in the table to determine the comparable name (e.g., Bill for William).
- Invalid entries: Here unexpected entries in a field can be flagged. For example, the fields that have the attributes of first name and last name are expected to be strings. However, if in those fields numerical values are present then an error signal is initiated. Similarly, for the HIC number field, numerical value is expected, if it is otherwise then an error signal is displayed.
- Blockings: If two data sets A and B are to be linked, the number of possible comparison equals the product of the number of records in the two data sets $|A| \times |B|$. For example, if data set A contains 1,000,000 and data set B has 50,000,000 records. The total number of possible comparisons would be $1{,}000{,}000 \times 50{,}000{,}000$. Assume each comparison takes 0.01 seconds, and then it takes 500,000,000,000 seconds. The example illustrates that it is computationally intractable to consider all pairs when the data sets are large [Christen and Churches]. To reduce the large amount of possible record pair comparisons, blocking is used to reduce the number of comparisons of record

pairs by bringing only potentially linkable record pairs together. This is achieved by using one or more record attributes to split the data sets into blocks. Only records having the same value in the blocking variable are compared. For text attributes, various phonic codes have been derived to avoid effects of spelling and aural errors in recording names [Gu et al., 2004]. This technique, however, becomes problematic if a value in the blocking variable is recorded erroneously, and the corresponding record is inserted into an incorrect block. To overcome this problem, several iterations with different blocking variables are normally performed [Christen and Churches].

- String comparison: two string attributes in two different records matching or mismatching can be analyzed by string comparison. The number of deletions, insertions, and transposition to convert one string to the other is evaluated. For example, "Alexandra" and "Alexander" differ by deleting "a" and inserting "e" to convert "Alexandra" to "Alexander". There are several methods to measure this "edit distance" as previously espoused in the literature review.
- Decision Model: Once the similarity values of individual attributes of a pair of records are obtained, they need to be combined. Form an overall degree of match to decide whether a record pair should be classified as a match, non-match, or possible match. Two methodologies are used for this task, (i) Fuzzy logic and (ii) Machine learning. In the fuzzy logic approach, membership functions link or map the basic quantitative measures onto the linguistic concepts, and define the degree of "similarity" (match) as "small, somewhat small, or large" for a given presented-contest pair. In machine leaning

approach, supervised learning is applied, a decision tree inducer. A decision tree can be constructed from a training data set to predict the matching status of the un-seen data set. The comparison vectors generated by a string comparison functions will contain the information related to the degree of match of the selected attributes for a pair of records. The result of attribute comparison from one record with the same one in the other can be simply defined as “matched” or “not-matched”.

- Result: At this point each field will have a quality of match value. Fuzzy logic has the positive attribute of being able to develop rules for the determination of a “match” or “non-match” decision for record matching from the quality of match of the individual attributes. In the machine learning approach, once the decision tree has been constructed, it can be easily transformed to decision rules. Based on these matching rules, a matching status can be determined.

4.2 General System Architecture

The matching method developed and the general system architecture is shown in Figure 7. There are two types of knowledge necessary for handling records matching: (1) the importance of the different attributes for deciding a matching and (2) the text formatting differences or transformations that may be relevant to the application domain. The application of this knowledge is very expensive, in terms of the user’s time. Record matching is accomplished in two stages. In the first stage, after repeated calls to third party payer databases, a set of possible matching records is generated.

The similarity between attributes of a pairs of records is defined by first determining a formatting transformation, and then calculating a similarity score of presented–contested pair of records. In the second stage, once the similarity scores for the attributes pairs are determined, the quality of the match is assessed using decision rules. These records are ranked based on the quality of match.

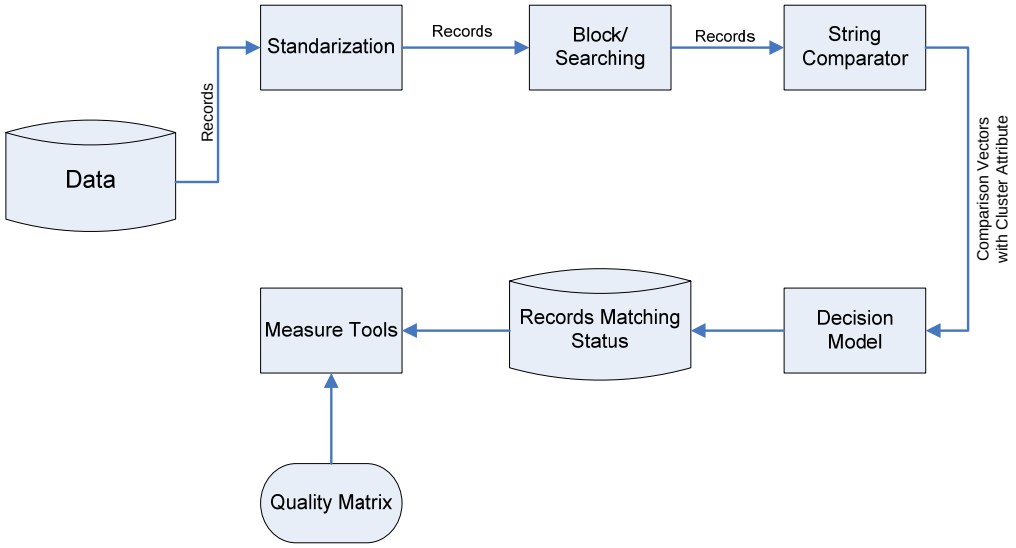


Figure 7 Records Matching Information Flow Diagram

4.3 Attribute Similarity Measurements

For the record matching problem, each attribute is compared individually for presented-contested pairs. How closely each of these attributes of a contested record match their counterparts in the presented record will be measured with string comparison functions. In this research, the attributes, HIC number, last name, first name, gender, date of birth in records are considered. In the literature, there are

several string comparator functions to measure the closeness of two strings for string comparison. They have some pros and cons. For example, the Levenshtein String Distance Statistic [Soukireff and MacKenzie, 2001; Navarro 2001] determines the minimum number of "primitive" errors (insertions, deletions, and substitutions) to convert a presented string into a contested string. Some string comparator functions espoused in the literature are presented in following sections. The evaluation of string comparator functions, for the application presented in this dissertation, is presented in Chapter 5.

4.3.1 The Levenshtein Edit Distance Metric

The Levenshtein edit distance (LED) [Levenshtein 1966]: this method uses edit distance to compare the similarity of two strings. Edit distance, a common measure of textural similarity, determines the minimum number of insertions, deletions, and substitutions of single character required to change one string into another (i.e., make two strings equal)[Gu et al. 2004]. The edit distance is symmetric, it holds $0 \leq d(x, y) \leq \max(|x|, |y|)$ [Navarro 2001]. Where *x, y* represents the number of characters in the two strings & *d(x,y)* is the distance measure

For instance: quickly

qucehkly

A simple character-wise comparison suggests that all letters after the "u" are incorrect. However, the final three ("kly") appear correct, despite misalignment. The minimum string distance is 3. In fact, there are often multiple answers, because more than one

minimum set of transformation may exist for the computed LED. Each transformation is called an "alignment", and represents a possible explanation of the error made [MacKenzie and Soukoreff, 2002]. A dynamic programming algorithm is used to find the optimal edit distance. The time complexity of this algorithm can be an issue for large databases [Gu et al., 2004; Jin et al., 2003].

The Levenshtein edit distance of two strings (s_1, s_2) can be denoted as $LED(s_1, s_2)$. A similarity metric between two strings is constructed, ranging from 0 to 1.0 using a normalized formula:

$$sim(s_1, s_2) = 1 - \frac{LED(s_1, s_2)}{MAXLEN(s_1, s_2)}$$

Where $MAXLEN$ denotes maximum numbers of characters in those two strings of length s_1 and s_2 and where *LED* is the Levenshtein edit distance, which is minimum number of deletions, insertions, and substitutions required to convert the contested string to presented on. From this formula, the similarity value of 1 represents the two compared strings are exactly the same, a perfect match, while value of zero indicates little similarity.

The formula can be used to quantify the closeness of two strings:

$$sim(s_1, s_2) = 1 - \frac{LED(s_1, s_2)}{MAXLEN(s_1, s_2)}$$

Because the maximum difference in this comparison of the two strings is the length of the longest string, the similarity is in scale of [0, 1]. For instance, if a pair of last names is compared, Taylor and Sailor, two substitutions, S → T and i → y, would

make the names identical - hence the LED is 2- therefore the similarity $sim(s_1, S_2)$ between the two strings is:

$$sim(s_1, s_2) = 1 - \frac{2}{6} = 0.667$$

However, a problem with using the Levenshtein String Distance Statistic to measure the closeness of two strings for comparison is that it can not judge which of two contested names is better matched to a presented string when the two generate the same LED but with different alphabetic configurations. For example, “Tyalor” and “Sailor” both generate the same LED when matched to “Taylor”; yet human judgment would select “Tyalor” as the better match. Ideally, the transformation of adjacent characters should count for less than two separate primitive errors.

4.3.2 The Jaro Algorithm

Another algorithm presented in the literature for similarity measures, the Jaro Algorithm, is also evaluated in this research. Introduced by Jaro [Jaro 1985, 1989, 1995], to take into account typical spelling deviations for measurement of closeness of two strings for comparison. Jaro’s string distance metric accounts for insertions, deletions, and transpositions. The value of similarity of two strings presented for comparison is given by the follow formula:

$$sim(s_1, s_2) = \frac{1}{3} * \left(\frac{N_{common}}{L_{s1}} + \frac{N_{common}}{L_{s2}} + 0.5 * \frac{N_{common} - N_{transposition}}{N_{common}}\right)$$

where s_1 and s_2 are the two strings to be compared, with lengths L_{s1} and L_{s2} respectively. N_{common} and $N_{transposition}$ are the numbers of common characters and transpositions.

For the previous example presented in section 4.3.1, "Taylor" – "Tyalor" and "Taylor" – "Sailor", the values of similarity calculated by Jaro's string distance metric are $sim_1 = 0.9444$ and $sim_2 = 0.7778$, respectively, which means that "Tyalor" is the better match with the Jaro's algorithm. This result is identical with human judgment. Using Jaro's algorithm the transposition of two characters causes less or equal of a downweighting of similarity than a substitution For example, consider the two first name pairs: "Martha" – "Marhta", and "Jonathon" – "Jonathan". The values of similarity by Jaro's string distance metric are $sim_1 = 0.9444$ for "Martha" – "Marhta", and $sim_2 = 0.9167$ for "Jonathon" – "Jonathan", respectively. Table 6 below provides more examples of similarity values obtained using Jaro's algorithm for several pairs of last names and first names.

In addition to how well the value of similarity of two strings indicates a match, whether the last name shown on the contested record is rare or not will also be considered. Name rarity can be measured by the relative frequency of a name in a database consisting of all service and eligibility records. Highly relative frequency indicates a common name and low frequency, a rare one. If a presented name matches to a rare contested name, the users have more assurance of overall record match than if the name matches a common name.

Shackeford	Shackelford	.9393
Cunningham	Cunnigham	.9667
Campell	Campbell	.9583
Nichleson	Nichulson	.9259
Massey	Massie	.8889
Galloway	Calloway	.9167
Lampley	Campley	.9048
Abroms	Abrams	.8889
Dixon	Dickson	.7905
Frederick	Fredrick	.9630
Michele	Michelle	.9583
Jesse	Jessie	.9444
Marhta	Martha	.9444
Jeraldine	Geraldine	.9259
Jonathon	Jonathan	.9167
Yvette	Yevett	.8889
Tanya	Tonya	.8667
Julies	Juluis	.8889
Dwayne	Duane	.8222

Table 6 Similarity scores for pairs of last names and first names

4.3.3 **The Jaro-Winkler Method**

The Jaro-Winkler string comparator is a method based on the Jaro algorithm [Jaro 1989], that was extended by Winkler [Poster and Winkler 1997]. The basic algorithm accounts for the number of common characters and the number of transposition of characters to transform one string to the other. The common characters in the two strings have to be located within half length of the shorter string. Winkler [Poster and Winkler 1997] modified the original string comparator introduced by Jaro in the following three ways:

Let l_{common} denote the longest common prefix of string s_1 and s_2. $l = \max(l_{common}, 4)$, Cohen [Cohen et al 2003] defined the Jaro-Winkler string comparator as follows:

$$sim(s_1, s_2) = sim_{Jaro}(s_1, s_2) + \frac{l}{10} * (1 - sim_{Jaro}(s_1, s_2))$$

Where: $sim_{Jaro}(s_1, s_2) = \frac{1}{3} * (\frac{N_{common}}{L_{s1}} + \frac{N_{common}}{L_{s2}} + 0.5 * \frac{N_{common} - N_{transposition}}{N_{common}})$

The Jaro-Winkler string comparator modifies the Jaro algorithm by slightly improving the weight of poorly matching pairs s_1 and s_2 that share a long common prefix. The Jaro and Jaro-Winkler string comparators seem to be intended primarily for short strings. Good measures of similarity are expected when these algorithms are applied to measure the similarity of pairs of last and first name in record matching problem. In this dissertation, the performance of the string comparator functions with real last and first names and human-generated errors, such as typographic errors, such as typographic errors, phonetic errors, key punch errors, random typing errors and the errors due to reversal of last and first name is evaluated and presented in Chapter 5.

4.4 Decision Models

In section 4.3, several string comparator functions, for computing the similarity values of selected attributes between the contested-presented pair of records, were presented. The similarity values define a quantitative measure of the similarity of attribute pairs in the corresponding records. For a decision as to whether a record pair should be classified as a “match”, “non-match”, or “possible match”, the similarity

values of all the attributes in a pair of records have to be combined. In this section, the use of fuzzy logic and machine learning approaches to define the degree of match between a pair of records is examined. The performance of these decision models with different comparison vectors will be evaluated and presented in section 5.2.

4.4.1 Fuzzy Logic Approach

In a manual record matching process, experienced users develop "rule of thumb" to judge how well a contested record matches a presented record. These rules may be conceptualized as aggregate weightings of numerous individual or attribute comparisons between the contested and presented records. Fuzzy logic enables the mapping of similarity values of two corresponding attributes in a contested-presented pair of records to linguistic concepts, such as "matched", "possible matched", and "not-matched". The selected attributes in the records may include last and first name, date of birth etc.

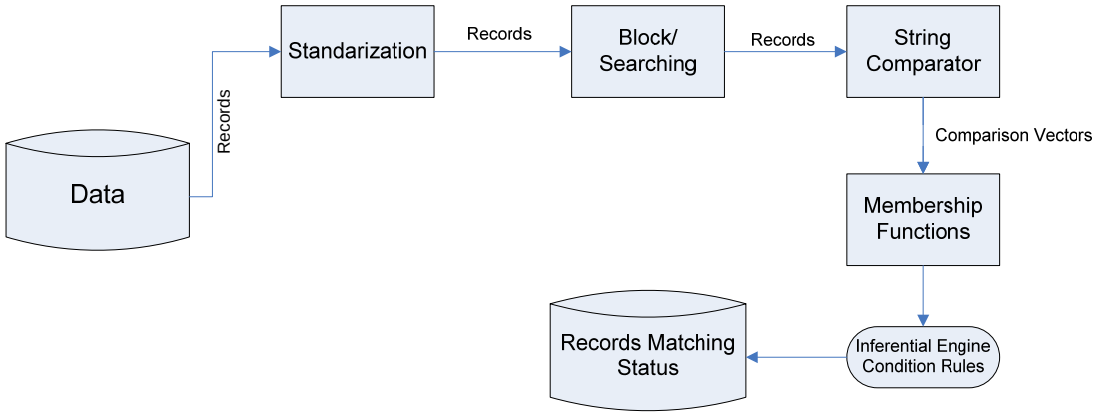


Figure 8 Fuzzy Logic Approach Process Diagram

4.4.1.1 Mapping Quantitative Measure to Linguistic Concepts Using Fuzzy Set Theory

Mapping quantitative measures such as similarities to linguistic concepts such as large similarity, somewhat large similarity, and small similarity so that they may be used in decision rules for defining the degree of match between two records is a two step process

1. Determine a quantitative measure of the comparison or attribute in question and
2. Assign a number between 0 and 1 to represent how strongly the comparison or attribute measure relates to the linguistic concept.

In the previous section, the use of Jaro's distance metric to quantify text similarity between two corresponding attributes is described. What follows is a description of how to measure, such as this are linked, to the linguistic concept.

Consider an example taken from Bezdek [1993], the statement "Lisa is old." If Lisa's age was 75, we might assign the statement the truth value of 0.80. The statement could be translated into set terminology as follows: "Lisa is a member of the set of old people." This statement would be rendered symbolically with fuzzy sets as:

$$m_{OLD}(Lisa) = 0.80$$

where m is the membership function, operating in this case on the fuzzy set of old people, which returns a value between 0.0 and 1.0. At this juncture it is important to point out the distinction between fuzzy systems and probability. Both operate over the

same numeric range, and at first glance both have similar values: 0.0 representing False (or non-membership), and 1.0 representing True (or membership). However, there is a distinction to be made between the two statements: The probabilistic approach yields the natural-language statement, "There is an 80% chance that Lisa is old," while the fuzzy terminology corresponds to "Lisa's degree of membership within the set of old people is 0.80." The semantic difference is significant: the first view supposes that Lisa is or is not old; it is just that we only have an 80% chance of knowing *which* set she is in. By contrast, fuzzy terminology supposes that Jane is "more or less" old, or some other term corresponding to the value of 0.80.

Membership functions links or map the basic quantitative measures onto the linguistic concepts. For example, if $x = sim(s_1, s_2)$ for a given presented-contested last name pair, and "1" represents "large $sim(s_1, s_2)$", then

$$m_1(x)$$

represents how strongly a $sim(s_1, s_2)$ value of x relates to the concept "large $sim(s_1, s_2)$". The membership functions describe the degree of similarity of two strings as the linguistic term. Our concepts of how well one string matches another include:

1. "large $sim(s_1, s_2)$", $m_1(x)$,
2. "somewhat large $sim(s_1, s_2)$", $m_2(x)$,
3. "small $sim(s_1, s_2)$", $m_3(x)$,

Figure 9 below illustrates an example set of membership functions relating similarity measures to the linguistic concepts, “large similarity”, “somewhat large similarity” and “small similarity”

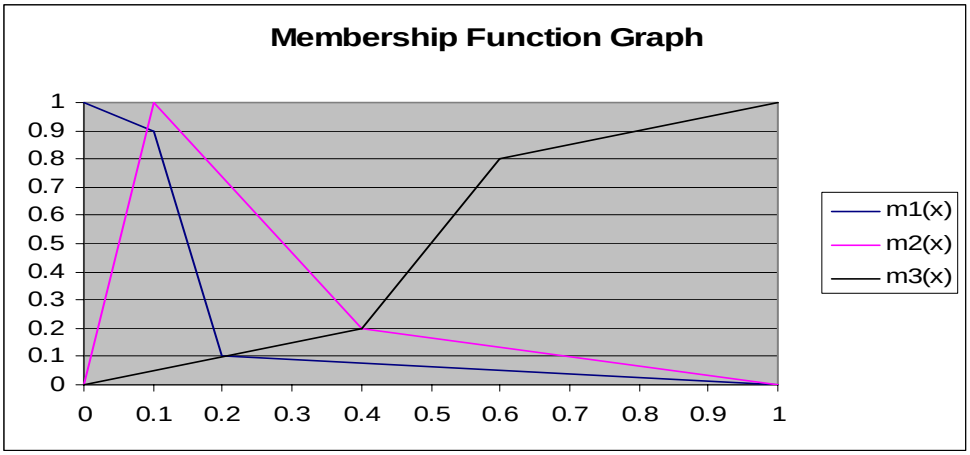


Figure 9 Membership Function for Last Name Pairs

$$m_1(x) = \begin{cases} -x + 1.0 & 0 \le x < 0.1 \\ -8x + 1.7 & 0.1 \le x < 0.2 \\ -(1/8)x + 1/8 & 0.2 \le x \le 1.0 \\ 0 & \textit{elsewhere} \end{cases}$$

$$m_2(x) = \begin{cases} 10x & 0 \le x < 0.1 \\ -(8/3)x + 19/15 & 0.1 \le x < 0.4 \\ -(1/3)x + 1/3 & 0.4 \le x \le 1.0 \\ 0 & \textit{elsewhere} \end{cases}$$

$$m_3(x) = \begin{cases} (1/2)x & 0 \le x < 0.4 \\ 3x - 1 & 0.4 \le x < 0.6 \\ (1/2)x + 1/2 & 0.6 \le x \le 1.0 \\ 0 & \textit{elsewhere} \end{cases}$$

$x =$ *similarity score*

These membership functions allow the interpretation of linguistic terms that go into the premises of the decision rules. The decision rules for matching records allow the results of a number of attribute similarities to be combined, via an inference engine, to yield conclusions representing a "favorable", "somewhat favorable", and "unfavorable" matches. These conclusions are consolidated and "defuzzified" to present a value between 0 and 100 that would represent the strength of the overall match of two records.

4.4.1.2 Rules of Inference

Last name, first name, HIC number, date of birth and gender are used as attributes for records matching. The "degree of match" for these attributes in two compared records are represented as follows:

Last name: 3 ways, large $sim(s_1, s_2)$, somewhat large $sim(s_1, s_2)$, small $sim(s_1, s_2)$

First name: 3 ways, large $sim(s_1, s_2)$, somewhat large $sim(s_1, s_2)$, small $sim(s_1, s_2)$

HIC number: 3 ways, large $sim(s_1, s_2)$, somewhat large $sim(s_1, s_2)$, small $sim(s_1, s_2)$

Last name rarity: 2 ways, rare or common

Date of birth: 2 ways, adequate or in adequate

Gender: 2 ways, matched or mis-matched

Primitive inference rules are structured by using only the "and" operator in the premises. These rules represent all the combinations of membership function. The sample primitive rules for match inference is as shown in Table 7

HIC Number	Last Name *sim*	Last Name Rarity	Date of Birth	Gender	First Name *sim*	Overall Match
Large *sim*	Large *sim*	Rare	Adequate	Matched	Large *sim*	F
Large *sim*	Large *sim*	Rare	Adequate	Matched	Somewhat large *sim*	F
Large *sim*	Large *sim*	Common	Inadequate	Mis-matched	Somewhat large *sim*	SF
Large *sim*	Large *sim*	Common	Inadequate	Mis-matched	Small *sim*	SU
Large *sim*	Somewhat large *sim*	Rare	Inadequate	Matched	Somewhat large *sim*	U
Somewhat large *sim*	Somewhat large *sim*	Rare	Adequate	Mis-matched	Somewhat large *sim*	SF
Somewhat large *sim*	Somewhat large *sim*	Common	Inadequate	Mis-matched	Small *sim*	U
Small *sim*	Large *sim*	Rare	Adequate	Mis-matched	Small *sim*	SU
Small *sim*	Small *sim*	Rare	Inadequate	Matched	Somewhat large *sim*	U

Table 7 the Sample of Primitive Rules for Matching Inference

*F – Favorable, SF- Somewhat Favorable, SU – Somewhat Unfavorable, U – Unfavorable

The fuzzy inference process with rules of this is shown in Figure 10.

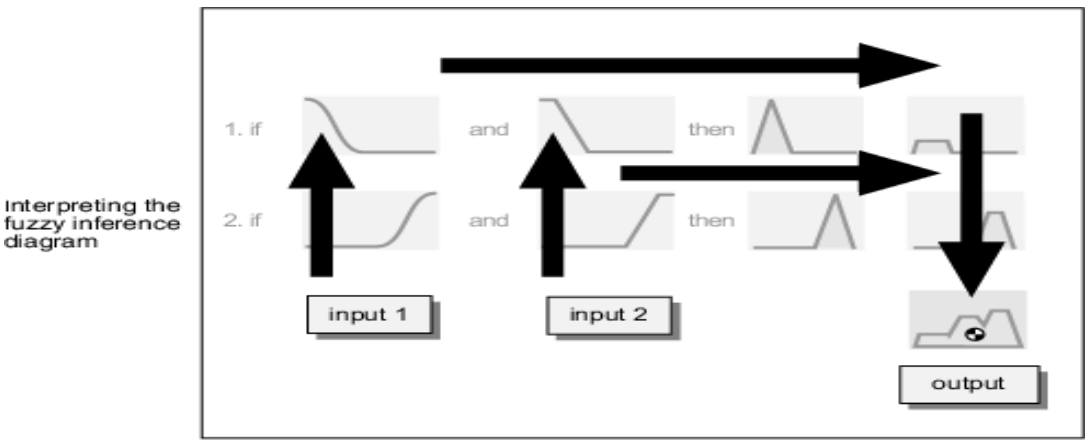


Figure 10 Fuzzy Inference Diagram (Source: www.mathworks.com, accessed on Jan.20, 2007)

The syllogistic rules and inference engine use membership functions of the inputs in the premises of rule to derive the membership function for the conclusion. An inference engine consists of two parts:

1) A rule processor that evaluates the premises of each rule in the rule set and establishes a membership function representing the degree of match according to that rule
2) An aggregation routine that combines the "degree of match" judgments of each rule into a single aggregated membership function that represents the conclusion. This function is then "defuzzied" obtain a numerical priority index, between 0 and 100, that quantifies the quality of match.

Numerous methods are available for defuzzifying the aggregate membership function to obtain a priority index. In this dissertation, the centroitd defuzzifiction approach proposed by Mamdani [Mukaidono, 2001] is applied owing to its intuitive appeal.

4.4.2 **Machine Learning Approach**

This research deals with *record linkage* applications that identify matching records in different data sources that refer to the same entity. In this scenario, the task is to decide whether a given pair of data records from one or more sources (generally databases) refers to the same entity, such as a person. Machine learning can be used to create rules for identifying likely matches even in the presence of spelling mistakes, nicknames, abbreviation, missing words, etc. For example, a record linkage

application should indicate that “Kate Simpson” at “1200 Main Street, Apt 304” in “Louisville, KY” is highly likely to be the same person as “Kate Simson” at

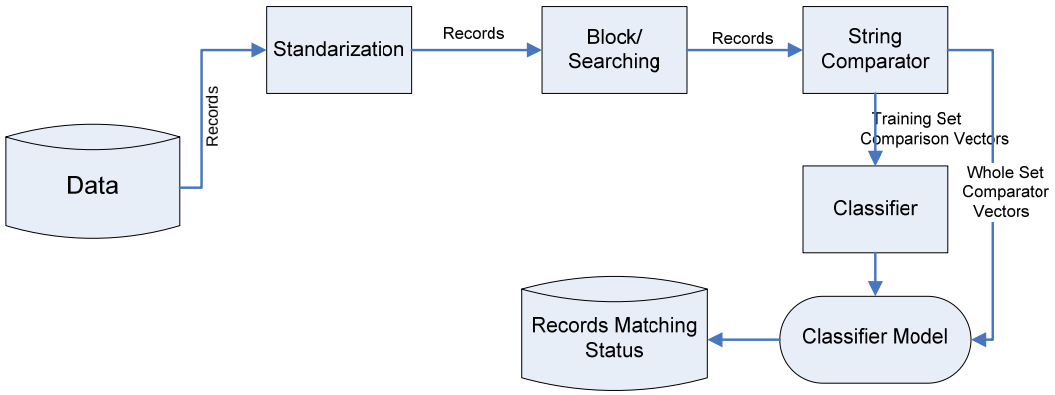


Figure 11 Decision Tree Approach Process Diagram

“1200 Main St.” in “Louisville, KY”. The records matching problem can be viewed as a pattern classification problem. The process of classification is accomplished via a learning that maps a data item into one of several predefined classes. In this research, two classes can be predefined as matched and non-matched. Every classification based on inductive-learning algorithms is given as input a set of samples that consist of vectors of attribute value and a corresponding class. The goal of learning is to create a classification model, which will predict the class for unseen data sets. In other words, classification is the process of assigning a class to an unlabeled record so the model predicts a class of records when the other attributes are given. A particularly efficient method for constructing classifiers from data is to generate a decision tree. The decision tree is the most widely used logic method. The tree is constructed through recursive partitioning of a universe. Recursive partitioning

identifies subgroups within a population that are relatively homogeneous with respect to experiencing an event (in the situation considered here the event is a match). A branching "tree" is created with the trunk (made up of the entire set of observations in an initially linked file). "Branches" emanate from the trunk and larger branches are further split into smaller branches based on a set of variables (a list of available matching variables) that minimize within-group heterogeneity. Repeated partitioning of the branches with different variables results in increasingly homogeneous subgroups of the population.

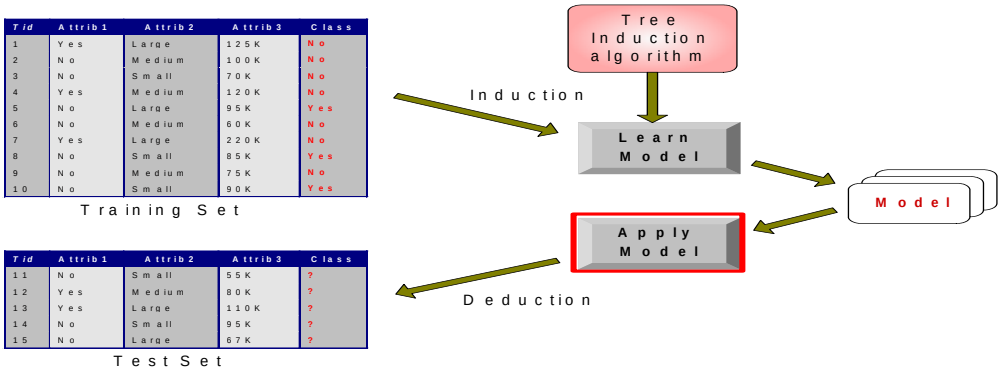


Figure 12 Decision Tree Classification Task

Source: http://www.cse.ohio-state.edu/~srini/674/chap4.ppt (Accessed on 12/31/07)

In this research, a predictive model is constructed through decision tree as shown in Figure 12. The process of decision tree approach is presented in the Figure 11. This approach is appropriate when adequate training data is available to train the model. There are several advantages for using the decision tree algorithm for matching records. The theory is well developed and computer programs to implement decision tree algorithm are widely available. A successful demonstration and evaluation of its use to the record linkage will be a useful contribution to the record matching problem.

4.4.2.1 Classification Algorithms

A well-known decision tree algorithm for generating decision trees based on univariate splits is Quinlan's ID3 with an extended version called C4.5 [Kantardzic 2003]. Greedy search methods, which involve growing and pruning decision tree structures, are typically employed in these algorithms to explore the exponential space of possible models. C4.5 algorithm starts withal the training samples at the root node of the tree. An attribute is selected to partition these samples. For each value of the attribute a branch is created, and the corresponding subset of samples that have the attribute value specified by the branch is moved to the newly created child node. Attribute selection in C4.5 algorithm is based on minimizing an information entropy measure applied to examples at a node. This is described in next section in more detail.

The skeleton of the C4.5 algorithm is based on Hunt's CLS method for constructing a decision tree from a set T of training samples. Let the classes be denoted as $\{C_1, C_2 \cdots, C_k\}$ there are three possibilities for the content of the set T:

a) T contains one or more samples, all belonging to a single class C_j. The decision tree for T is a leaf identifying class C_j.

b) T contains no samples. The decision tree is again a leaf but the class to be associated with the leaf must be determined from information other than T, such as the overall majority class in T. The C4.5 algorithm uses as a criterion the most frequent class at the parent of the given node.

c) T contains samples that belong to a mixture of classes. In this situation, the idea is to refine T into subsets of samples that are heading towards a single class collection of samples. Based on single attribute, an appropriate test that has one or more mutually exclusive outcomes $\{O_1, O_2, \cdots, O_n\}$ is chosen. T is partitioned into subset T_1, T_2, $\cdots$, T_n where T_i contains all the samples in T that have outcome O_i of he chosen test. The decision tree for T consists of a decision node identifying the test and one branch for each possible outcome.

An illustration of the algorithm can be provided by a simple example for the proposed application. Suppose there is a training data set. Each record contains the HIC code, the first and last name, gender, and date of birth of an individual patient. The correct matching status of the compared pair of records is known. For numerical fields such as HIC code, Euclidean distance in an m-dimensional feature space is computed using the formula:

$$d(x_i, x_j) = \left(\sum_{k=1}^{m} (x_{ik} - x_{jk})^2 \right)^{1/2}$$

Where x_i and x_j are two the two HIC number to be compared; and $i \neq j$. m is number of digits in the HIC code.

For character string fields, the well-known Jaro's string distance metric is computed using the formula below:

$$sim(s_1, s_2) = \frac{1}{3} * \left(\frac{N_{common}}{L_{s1}} + \frac{N_{common}}{L_{s2}} + 0.5 * \frac{N_{common} - N_{transposition}}{N_{common}} \right)$$

Where s_1 and s_2 are the two strings to be compared, with lengths l_{s1} and l_{s2} respectively. N_{common} and $N_{transposition}$ are the numbers of common characters and transpositions respectively. So the similarity score of the training set is shown in Table 8, for example:

HIC code	First Name	Last Name	Gender	DOB	Matched
4.51	0.70	0.75	M	0.65	No
1.68	0.90	0.78	M	0.77	Yes
1.32	0.85	0.88	F	0.80	Yes
0.45	0.95	0.90	F	0.91	Yes
2.68	0.70	0.65	F	0.65	No
5.63	0.71	0.75	M	0.75	No
4.45	0.78	0.81	F	0.72	No
4.06	0.65	0.63	M	0.70	No
5.74	0.75	0.65	F	0.70	No
3.28	0.80	0.87	M	0.85	Yes
1.56	0.70	0.80	M	0.80	Yes
5.18	0.70	0.65	F	0.72	No
5.32	0.80	0.72	F	0.65	No
1.18	0.76	0.68	F	0.76	No

Table 8 Similarity Score for an Example Data Set

Suppose there is the set T of training samples into subsets T_1, T_2, …., Tn. If S is any set of samples, let $freg(C_i, S)$ stand for the number of sample in S that belong to class C_i and let $|S|$ denote the number of samples in the set S. The following relation gives the computation of the entropy of the set S (bits are units)

$$Info(S) = -\sum_{i-1}^{k}((freq(C_i, S)/|S|) \cdot \log_2(freg(C_i, S)/|S|))$$

Consider a similar measurement after T has been partitioned in accordance with n outcomes of on attribute test X. the expected information requirement can be found as the weighted sum of entropies over the subsets:

$$Info_x(T) = -\sum_{i=1}^{n}((|T_i|/|T|) \cdot Info(T_i))$$

The quantity

$$Gain(X) = Info(T) - Info_x(T)$$

Measures the information that is gained by partitioning T in accordance with the test X. the gain criterion selects a test X to maximize Gain(X), i.e., this criterion will select an attribute with the highest info-gain.

C4.5 is top-down inducing algorithm. It starts to split the branch from the attribute which is the most important to the classification task based on the information theory concept: entropy. Choosing the attribute is the most complicated step in C4.5 algorithm. This is accomplished by looking at each attribute in turn and all candidate conditions and seeing what sort of split they would achieve. The C4.5 algorithm uses the entropy of the slit to decide on the best attribute. Nine samples belong to class 1, labeled “No”, and five samples to class 2, labeled “Yes”. The entropy is calculated for before splitting as

$$Info(T) = -9/14\log_2(9/14) - 5/14\log_2(5/14) = 0.940 \text{ bits}$$

Then we need to test on each attribute to get the information gain and choose the attribute with highest information gain to start splitting.

In general, C4.5 contains mechanisms for proposing three types of tests:

a) The "standard" test on a discrete attribute, with one outcome and one branch for each possible value of that attribute
b) If attribute Y has continuous numeric values, a binary test with outcomes $Y \leq Z$ and $Y > Z$ could be defined, by comparing its value against a threshold value Z.
c) A more complex test also based on a discrete attribute, in which the possible values are allocated to a variable number of groups with one outcome and branch for each group.

In this example, there are five attributes, HIC code, first name, last name, gender and date of birth. The similarity measures of these attributes are continuous values except for gender the value for which is discrete.

If the test and splitting is based on gender (male or female), a computation will give a result:

$$Info_{Gender}(T) = 6/14(-3/6\log_2(3/6) - 3/6\log_2(3/6)) + 8/14(-6/8\log_2(6/8) - 2/8\log_2(2/8)) = 0.892 \text{ bits}$$

And corresponding gain is

$$Gain(Gender) = 0.940 - 0.892 = 0.048 \text{ bits}$$

In the test and splitting is based on attribute first name which is continuous value, for continuous attribute, the training samples are first sorted on the values of the attribute Y being considered. There are only a finite number of these values. So let us denote

them in sorted order as $\{v_1, v_2, \cdots v_m\}$. There are thus only m-1 possible splits on Y. The C4.5 chooses as the threshold a smaller value v_i for every interval $\{v_i, v_{i+1}\}$. This ensures that the threshold values appearing in either the final decision tree or rules or both actually occur in the database.

To illustrate this threshold-finding process, the possibilities of attribute splitting are analyzed. For example of the attribute first name is {0.65, 0.70, 0.75, 0.78, 0.80, 0.85, 0.90, 0.95, 0.96}, and the set of potential threshold value Z is {0.65, 0.70, 0.75, 0.78, 0.80, 0.85, 0.90, 0.95}. Out of these eight values the optimal Z (with the highest information gain) should be selected. In this case, the optimal Z value is Z = 0.80 and the corresponding process of information-gain computation for the test attribute first name (First Name $\leq$ 0.80 or First Name > 0.80) is as follow:

$$Info_{FirstName} = 9/14(-7/9\log_2(7/9) - 2/9\log_2(2/9))$$

$$= 5/14(-2/5)\log_2(2/5) - 3/5)\log_2(3/5))$$

$$= 0.837 \text{ bits}$$

$$Gain(FirstName) = 0.940 - 0.837 = 0.103 \text{ bits}$$

The C4.5 would do the same test on the other attributes: HIC code, last name, and date of birth to get the corresponding information-gain for each attribute. The slitting starts with the attribute with highest information-gain. After the first splitting, the C4.5 repeats to calculate information-gain for each branch and split until the final leaf nodes in which the subsets of cases in each of the branches belong to the same class (matched or non-matched)

4.4.2.2 Pruning Decision Trees

Pruning of the decision tree is done by replacing a whole subtree by a leaf node [Kantardzic 2003]. The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf. In replacing the subtree with a leaf, the algorithm expects to lower the predicted error rate and increase the quality of a classification model. But computation of error rate is not simple. An error rate based only on a training data set does not provide a suitable estimate. One possibility to estimate the predicted error rate is to use a new, additional set of test samples if available, or to use the cross-validation techniques. This technique divides initially available samples into equal size blocks and, for each block; the tree is constructed from all samples except this block and tested with a given block of samples.

Having grown the largest possible tree based on certain criterions, such as maximization of profit, minimum of classification error rate, the pruning process starts with the largest tree, and eliminate one split as each step. If the original tree has M leaves and if one split is removed at given point, then the tree becomes a subtree of size M-1. If one split is removed at a different point, the tree becomes another subtree with size of M-1.Thus; there can be a number of sub-trees of size of M-1. If two splits are removed at a time, the tree becomes a subtree with size of M-2. There could be more than one subtrees with size of M-2.This process continues until there is a tree with only one leaf. At the end of this process, there will be a sequence of trees of

sizes, M, M-1, M-2, M-3…..1. So the pruning process is to select an optimal subtree based on certain criteria from this sequence.

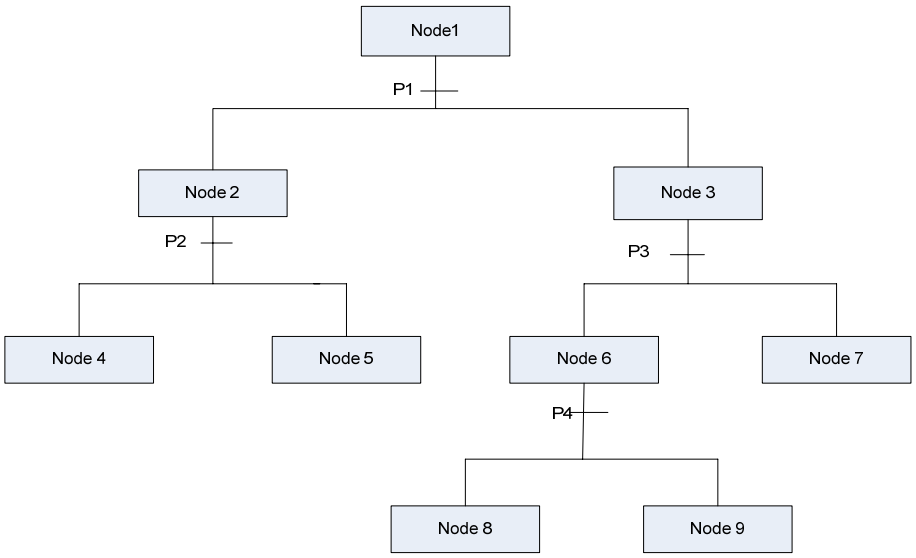


Figure 13 Pruning a Tree

Figure 13 illustrates the process of pruning a large tree as in Figure 13 at every possible point to create an optimal subtree. The tree shown in Figure 13 has nine nodes. The maximal tree has five leaf nodes, Node 4, Node 5, Node 7, Node 8 and Node 9. P1, P2, P3, and P4 are possible points to remove a split to prune the tree to get a subtree. If one split is removed at P4, the tree becomes a subtree of size of 4, which has four leaf nodes, Node 4, Node 5, Node 6 and Node 7. If one spit is removed at P2, the tree becomes a subtree of size of 4 with 4 leaf nodes, Node 2, Node 8, Node 9 and Node 7. So there are two subtrees with size of 4.

If two splits are removed at P3 and P4, the tree becomes a subtree with three leaf nodes, Node 4, Node 5 and Node 3. If two splits are removed at P2 and P4, then the

tree becomes a subtree with three leaf nodes, Node 2, Node 6 and Node 7. So there are two subtrees with size of 3.

If three splits are removed at P2, P3, and P4, the tree becomes a subtree with two leaf nodes, Node 2 and Node 3. There is one subtree with size of 2.

If four splits are removed at P1, P2, P3, and P4, the tree becomes a subtree with only one leaf node, Node1.

There is no further possible pruning in this case. So the sequence consists of one tree with five leaves (the original tree), two trees with four leaves, two trees with three leaves, one tree with two leaves, and one tree with one leaf. The process of selecting an optimal tree in this sequence has two steps 1) Select a best tree from a number of trees of the same size. The final sequence consists of five trees, each with size of 1, 2, 3, 4 and 5. 2) Select an optimal tree within the final sequence based on certain criteria. These criteria could be profit maximization, cost minimization, minimization of classification error rates, minimization of average squared error, or maximization of lift. The criterion, minimization of classification error rates, is applied for this research to prune the largest tree to an optimal decision tree.

After a decision tree is generated, it can be easily transformed into a rule set by converting each path of the tree into a rule as follows: (1) the internal nodes and their output branches are converted into conditions of the antecedent ("if-part") of the rule; (b) the leaf nodes are converted into the consequent ("then-part") of the rule. An

example of decision tree for the patient records matching problem is shown in Figure 14.

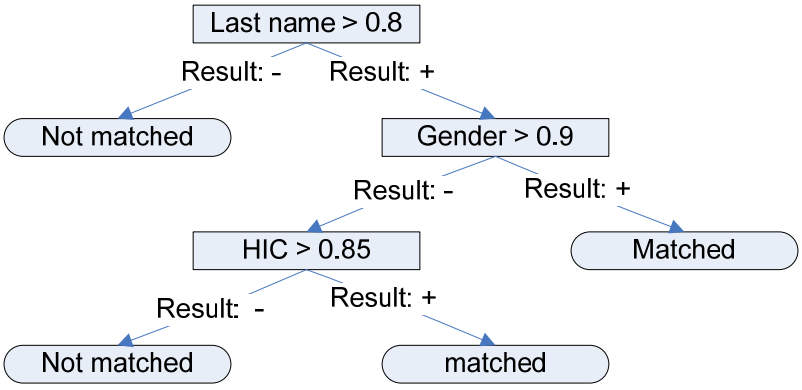


Figure 14 an Example of Decision Tree

4.4.2.3 Applying Inductive Model for Records Matching

Decision tree predictive models are applied to the record matching problem. When pairs of records are brought together for comparison, decisions must be made as to whether these are to be regarded as linked, not linked, or possibly linked, depending upon the various agreements and disagreements of items of identifying information. For the patient record matching task, a possible measurement would be to compare family names on the two records, and assign the value of 1 for those pair where there is an agreement and 0 for those pairs where is a disagreement. So a set of decision rules are needed to classify those records as matched or not-matched. For inducing a model, two labels, "matched" and "not-matched", are assigned to records in a training dataset. This information is given to a decision tree inducer for building a model for classification.

The process of "learning" that develops the decision tree that leads to the matching rules results in the highest possible accuracy for object matching. The classification process determines what attributes statistically significantly influence the outcome of the matching, as well as, the thresholds on the similarity scores for each attribute. Thresholds must be set to decide which linkages should be accepted as true without any human evaluation. If the threshold is set too low, the defined linkage groups may incorrectly join the medical records for different persons. But if the threshold is set too high, there will be undesired duplication of persons in the systems. Several matching rules may be necessary to properly classify the objects for a specific domain application. Examples of matching rules are:

Rule 1: Last name > 0.8 and Gender > 0.9 ➔ Matched

Rule 2: Last name > 0.8 and HIC > 0.85 ➔ matched

In summary, the rules for classifying data records as matched or not matched are obtained through decision tree learning [Hall et al., 1998; Kudoh et al., 2003]; i.e. the above mentioned, decision tree is developed through an inductive learning technique. As illustrated in this section, creating a decision tree is an iterative process, where the attribute with the greatest information gain is chosen at each level. Once a decision tree is created, it is easily translated into rules for classifying records as matched or not matched.

4.4.3 Model Performance Assessment

There are several measures that define the quality of record matching. They include:

1) The number of record pairs linked correctly (true positive, TP).
2) Type I error, the number of incorrectly linked records that do not represent the same entity (false positive, FP)
3) The number of record pair unlinked correctly (true negative, TN).
4) Type II error, the number of linked record pairs that are not identified, which represent the failure to identify true linkages (false negative, FN)

These measures are represented in a "Confusion Matrix", which is defined in the next section.

4.4.3.1 Matrix for Performance Evaluation – "Confusion Matrix"

The confusion matrix is shown in Table 9 below:

	PREDICTED CLASS		
ACTUAL CLASS		Class = Yes	Class = No
	Class = Yes	a (TP)	b (FN)
	Class = No	c (FP)	d (TN)

Table 9 "Confusion Matrix"

One of most widely-used metric for assessing the quality of record matching methodologies is "accuracy". This term is defined below:

$$Accuracy = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

However, there are some limitations when using accuracy to evaluate the performance of the predictive capability of a model. For instance, consider a two-class problem like the patient record matching problem. If the training data set has 10000 records, 9990 records are not-matched or class 0 and 10 records are matched or class 1. If a model was created, which predicted everything to be class 0, then accuracy is 9990/10000 = 99.9%. So, accuracy is misleading in this case because the model does not detect any records that are matched. Hence, the profit or cost of the failure to detect records that were matched may be computed as a measure of performance. This approach is explained in the next section, using a bank loan analogy.

Computing the profit or cost of classification may be described via a profit (cost) matrix, such as that is shown in Table 10.

	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	Class = Yes	Class = No
	Class = Yes	C(Yes|Yes)	C(No|Yes)
	Class = No	C(Yes|No)	C(No|No)

Table 10 Matrix for Computing Profit (Cost) of Classification

Where C(i|j): Cost of misclassification class j as class i

In a bank loan predictive model, the objective of the model is to maximize profit. When the model makes a correct prediction, the bank would profit by selling a loan, while if the model makes a bad prediction, such as when a bank sells a loan to someone who with bad credit history, and then the bank may lose money. For example, consider the profit matrix shown in table 11:

	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	+	-
	+	109	0
	-	-3400	0

Table 11 Profit/Cost Matrix for a Bank Load Predictive Model

Suppose two models M1 and M2, provide results shown in tables 12 and 13 below

	Approval	Denial	Total
Good Loan	126,954 (76.7%)	25,579 (15.5%)	152,533 (92.2%)
Bad Loan	2,159 (1.3%)	10,820 (6.5%)	12,979 (7.8%)
Total	129,113 (78.0%)	36,399 (22.0%)	165,512 (100%)

Table 12 Confusion Matrix for Model M1

For Model M1:

$$Accuracy = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN} = 83.24\%$$

$$\Pr ofit = 6{,}497{,}386$$

	Approval	Denial	Total
Good Loan	142,653 (86.2%)	11,605(7.0%)	154,258 (93.2%)
Bad Loan	1,511 (0.9%)	9,743 (5.9%)	11,254 (6.8%)
Total	144,164 (87.1%)	21348(12.9%)	165,512 (100%)

Table 13 Confusion Matrix for Model M2

For Model M2:

$$Accuracy = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN} = 92.08\%$$

$$\Pr ofit = 10{,}411{,}777$$

From the Tables 12 and 13, it can be observed that using model M2, has improved the good loan approval rate from 76.7% to 86.2%, while bad loan approval rate is decreased from 1.3% to 0.9%. The profit is increased from 6, 497,386 to 10,411,777. Hence, with this measure, the model M2 is better than model M1.

In information retrieval application, metrics such as precision, recall, and F measures are used to assess the quality of the retrieval algorithm. Those measures are applied in this research. These measures are explained in the next section.

4.4.3.2 Precision

The proportion of retrieved and relevant documents to all the documents retrieved:

$$precision = \frac{\left|\{relevantdocuments\} \cap \{retrieveddocuments\}\right|}{\left|\{retrieveddocuments\}\right|}$$

In binary classification, precision is analogous to a positive predictive value. Precision takes all retrieved documents into account. It can also be evaluated at a given cut-off rank, considering only the topmost results returned by the system. This measure is called precision at n or P@n.

Note that the above meaning and usage of "precision" is borrowed from the field of Information Retrieval and may differ from the definition of accuracy and precision used in other fields. Precision measures how much of a result set is on target—i.e., how many of the returned documents are actually relevant. For example, an information retrieval (IR) system that achieves 75% precision means that 75% of all documents that are assessed to be relevant are actually relevant, and 25% of documents are erroneously identified as such. Precision is critical in IR applications such as litigation because a low precision rate means that a large number of irrelevant documents will need to be reviewed. This translates into considerable costs, as well as significant delays in processing time.

4.4.3.3 Recall

The proportion of relevant documents that are retrieved, out of all relevant documents available:

$$recall = \frac{\left|\{relevantdocuments\} \bigcap \{retrieeveddocuments\}\right|}{\left|\{relevantdocument\}\right|}$$

In binary classification, recall is called sensitivity.

It is trivial to achieve recall of 100% by returning all documents in response to any query. Therefore recall alone is not enough but one needs to measure the number of irrelevant document also, for example by computing the precision.

Recall measures how much of a target set has been found—i.e., how many relevant documents have actually been identified as such. For example, an IR system that achieves 80% recall means that 80% of all relevant documents were actually found, and 20% of all relevant documents were not found during the review.

Recall is also critical in IR application in litigation because (1) parties have an obligation to produce documents responsive to discovery requests; and (2) a more complete set of relevant documents enables a more thorough review of evidence to develop an effective case strategy— and reduces the risk of overlooking information that could be critical to your case.

4.4.3.4 F-measure

F-measure is the weighted harmonic mean of precision and recall, the traditional F-measure or balanced F-score is:

$$F = 2 \cdot (precision \cdot recall) / (precision + recall)$$

This is also known as the F_1 measure, because recall and precision are evenly weighted. Two other commonly used F measures are the F_2 measure, which weights

recall twice as much as precision, and the $F_{0.5}$ measure, which weights precision twice as much as recall.

There are two advantages to using these metrics for IR application: i) information retrieval researchers are familiar with them and ii) they do not require a negative case count. The two primary metrics are *precision* and *recall*. Given a subject and a gold standard, precision is the proportion of cases the subject classified as positive that were positive in the gold standard. It is equivalent to positive predictive value. Recall is the proportion of positive cases in the gold standard that were classified as positive by the subject. It is equivalent to sensitivity. The two metrics are often combined as their harmonic mean, known as the *F-measure*, which can be formulated as follows:

$$F_\alpha = (1+\alpha)\cdot(precision \cdot recall)/(\alpha \cdot precision + recall)$$

α allows one to weight either precision or recall more heavily, and they are balanced when α equals 1. In most experiments, there is no particular reason to favor precision or recall, so most researchers use α equal 1.

For record linkage, precision can be defined, in terms of matches, as the number of correctly linked record pairs divided by the total number of linked record pairs. So precision is equivalent to the positive predicted value defined above. Similarly, recall is defined, in terms of matches, as the number of correctly linked record pairs divided by the total number of true match record pairs. As a result, recall is equivalent to sensitivity defined above. Of course, precision and recall can also be defined in terms of non-matches. Alternatively, combined measures of precision and recall can be

defined in terms of overall record pairs correctly classified (matches and non-matches) [Gu et al., 2004].

In record matching problem, Precision, Recall and F-Measure can be computed as follows:

$$\Pr ecision(p) = \frac{a}{a+c} = \frac{TP}{TP+FP}$$

$$\mathrm{Re}call(r) = \frac{a}{a+b} = \frac{TP}{TP+FN}$$

$$F-measure(F) = \frac{2rp}{r+p} = \frac{2a}{2a+b+c} = \frac{2TP}{2TP+FN+FP}$$

Precision is based on the number of records classified in the categories C(Yes|Yes) and C(Yes|No) and

Recall is based on the number of records classified as C(Yes|Yes) and C(No|Yes). While, the F-measure is based on the numbers of records in all categories, except C(No|No).

Precision, recall and F-measure can be applied in this research. Hence the considering the example below:

After searching, a set of candidates are generated and saved in a candidate pool. F-measure can be used to measure the quality of the searching algorithm. In order to use Precision, Recall, and F-measure, it is needed to define these measures: relevant records, and retrieved records. For the given present records, for instance, there are 30

contested records generated and 5 records are related to the present record. The quality of match metrics, “Precision”, “Recall” and “F-measure” are obtained as shown below:

$$precision = \frac{\left|\{relevantdocuments\} \cap \{retrieveddocuments\}\right|}{\left|\{retrieveddocuments\}\right|}$$

$$= \frac{\{5\} \cap \{30\}}{\{30\}} = \frac{1}{6}$$

$$recall = \frac{\left|\{relevantdocuments\} \bigcap \{retrieeveddocuments\}\right|}{\left|\{relevantdocument\}\right|}$$

$$= \frac{\{5\} \cap \{30\}}{\{5\}} = 1$$

$$F = 2 \cdot (precision \cdot recall) / (precision + recall)$$

$$= 2 \cdot (\frac{1}{6} \cdot 1) / (\frac{1}{6} + 1) = \frac{2}{7}$$

A confusion matrix can be summarized using various formulas. The most commonly used formulas [Lu et al., 2004] are presented in Table 14

Measure	Formula	Intuitive Meaning
Precision	TP / (TP + FP)	The percentage of positive predictions that are correct.
Recall / Sensitivity	TP / (TP + FN)	The percentage of positive labeled instances that were predicted as positive.
Specificity	TN / (TN + FP)	The percentage of negative labeled instances that were predicted as negative.
Accuracy	(TP + TN) / (TP + TN + FP + FN)	The percentage of predictions that are correct.

Table 14 The Most Commonly Used Formulas Derived from a Confusion Matrix

A weighted value for accuracy may also be computed as shown below:

$$Weighted\ \ Accuracy = \frac{w_1 a + w_4 d}{w_1 a + w_2 b + w_3 c + w_4 d}$$

The weighted accuracy is the average of the separate accuracy of each class. These quality metrics are used to compare the performance of the decision models that result from using the different comparison vectors generated by the string comparator functions referred to in section 5.2.

4.4.3.5 ROC Curves

These performance metrics derived from the confusion matrix, such as, "Precision, and "Recall" are sensitive to data with class skew [Fawcett & Flach 2005]. Hence, Received Operating Characteristic (ROC), which are not sensitive to data with class skew, have been applied for classification model assessment. ROC curves are two-dimensional graphs that visually depict the performance and performance trade-off of a classification model [Fawcett, 2004; Flach et al. 2003; Flach 2004; Hamel]. We define ROC curves in terms of the confusion matrix, the true positive rate (TPrate) and the false positive rate (FPrate):

$$\text{TPrate} = \frac{TP}{TP + FN}\ ,\ \ \text{FPrate} = \frac{FP}{TN + FP}$$

ROC graphs are constructed by plotting the true positive rate against the false positive rate [Hamel] shown in Figure 15.

Classifiers can be mapped to a ROC graph. In the Figure 15, the top left corner is perfect performance point. Any point in a ROC graph, the closer to the top left corner, the better performance. So point A in Figure 15 is superior to point B. ROC curves characterize the performance of a classifier model as a curve instead of a single point on the ROC graph.

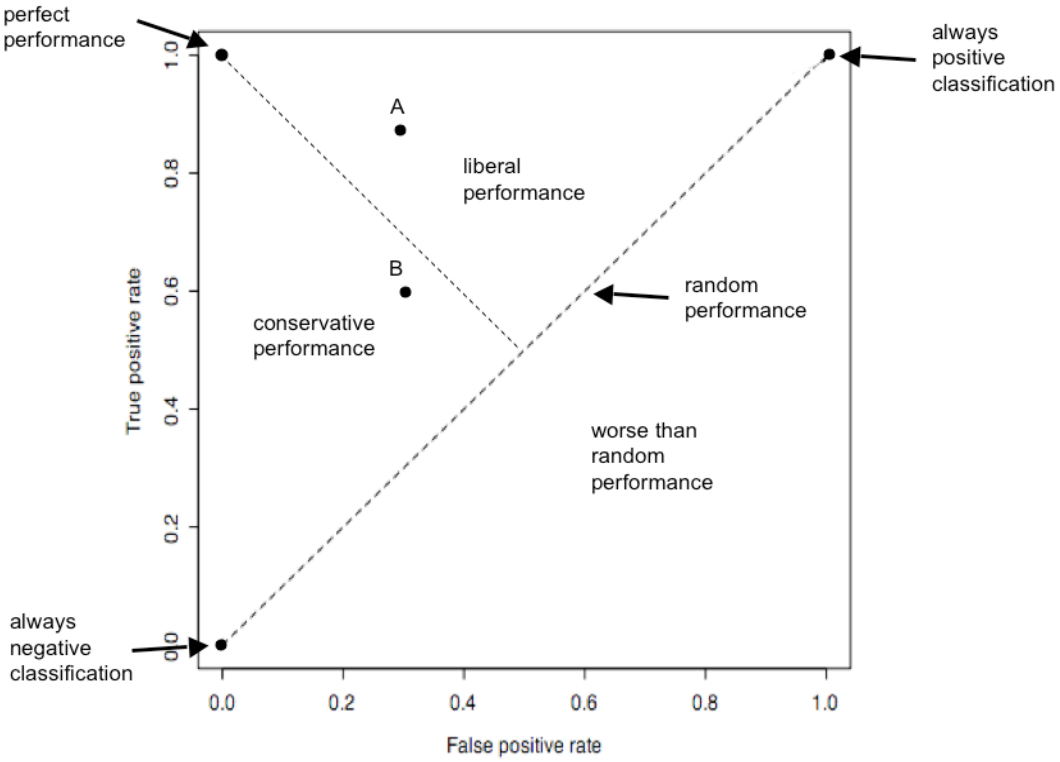


Figure 15 ROC Graph

ROC space is appropriate for measuring the success of subgroup discovery for classification, since subgroups whose TPrate/FPrate tradeoff is close to the diagonal

can be discarded as insignificant. Conversely, significant subgroups are those sufficiently distant from the diagonal. Because in ROC graph, it's easy to see that the diagonal represents random performance.

ROC curves can be one of criteria used for model assessment. If there are more than one model, model assessment and comparison are needed to conduct based on their performance metrics and certain criteria. ROC analysis (Provost and Fawcett, 2001) provides a principled procedure for determining the operating characteristics under which each model is optimal.

5.0 EXPERIMENTAL EVALUATION

Medical record linkage is becoming increasingly important as clinical data is distributed across many independent databases and systems, within and among institutions as separate collections with varying types of identifying information. Because pairs of strings often exhibit typographical variation (e.g., Smith versus Smoth), the record linkage needs effective string comparator functions that deal with typographical variations. While approximate string comparison has been a subject of research in computer science for many years, some of the most effective ideas in the record linkage context were introduced by Jaro [Jaro1989; Winkler 1985, 1990]. Budzinsky [1991] concluded that the Jaro distance metric [Jaro 1989], Jaro-Winkler string comparator, modified by Winkler [1990], and bigrams method worked well after he reviewed about twenty string comparators [Porter and Winkler 1997]. However, there is a paucity of literature describing the actual performance of such comparators in patient record linkage [Grannis et al 2004]. This research is focused on approximate matching problems; the object of approximate matching is to get good enough matching results for a specific domain. In this research, distance metrics are used to measure the similarity of presented-contested pairs of patient records for each attribute. A good algorithm is able to give different similarity scores by considering various possible differences of the attributes of a pair of records in

databases. It should give higher similarity values for the strings that are known to be the same but misspelled; while it gives lower similarity values for the strings that are known to be different. Different string comparison functions that establish agreement or disagreement between corresponding strings are evaluated. The performance of three string distance metrics, the Levenshtein edit distance, the Jaro algorithm, and the Jaro-Winkler method is assessed. The type I and type II error rates, metrics such as precision, recall and F measure by using different comparison vectors with decision models for test datasets will be compared in later sections.

5.1 String Similarity Function Comparison

In order to conduct string similarity function comparisons in patient record linkage, a new tool has been developed. This new tool is coded in Java with a user- friendly interface. A screen shot of this tool is shown in Figure 16 (Program codes in Appendices 1-4). Various string distance functions from an open-source Java toolkit, SecondString [Cohen et al. 2003], are integrated in this new tool. A string similarity function maps a pair of strings str_1 and str_2 to a similarity value in real number sim, where sim equals 1, which means these two strings are exactly the same, while sim equals 0, these two strings are totally different. sim is in a higher value, which indicates greater similarity between a presented-contested pair of strings. Users may input attributes of a contested patient record, such as last name, first name, and date of birth etc, choose a presented data set by browsing a data file, select a specific string distance function by checking a checkbox in the user interface, then compare a

contested string with presented strings by clicking on “Comparison” button. The similarity values would be returned to users.

According to Porter and Winkler [1997] more than 20% of first names and last names in many lists were entered incorrectly.

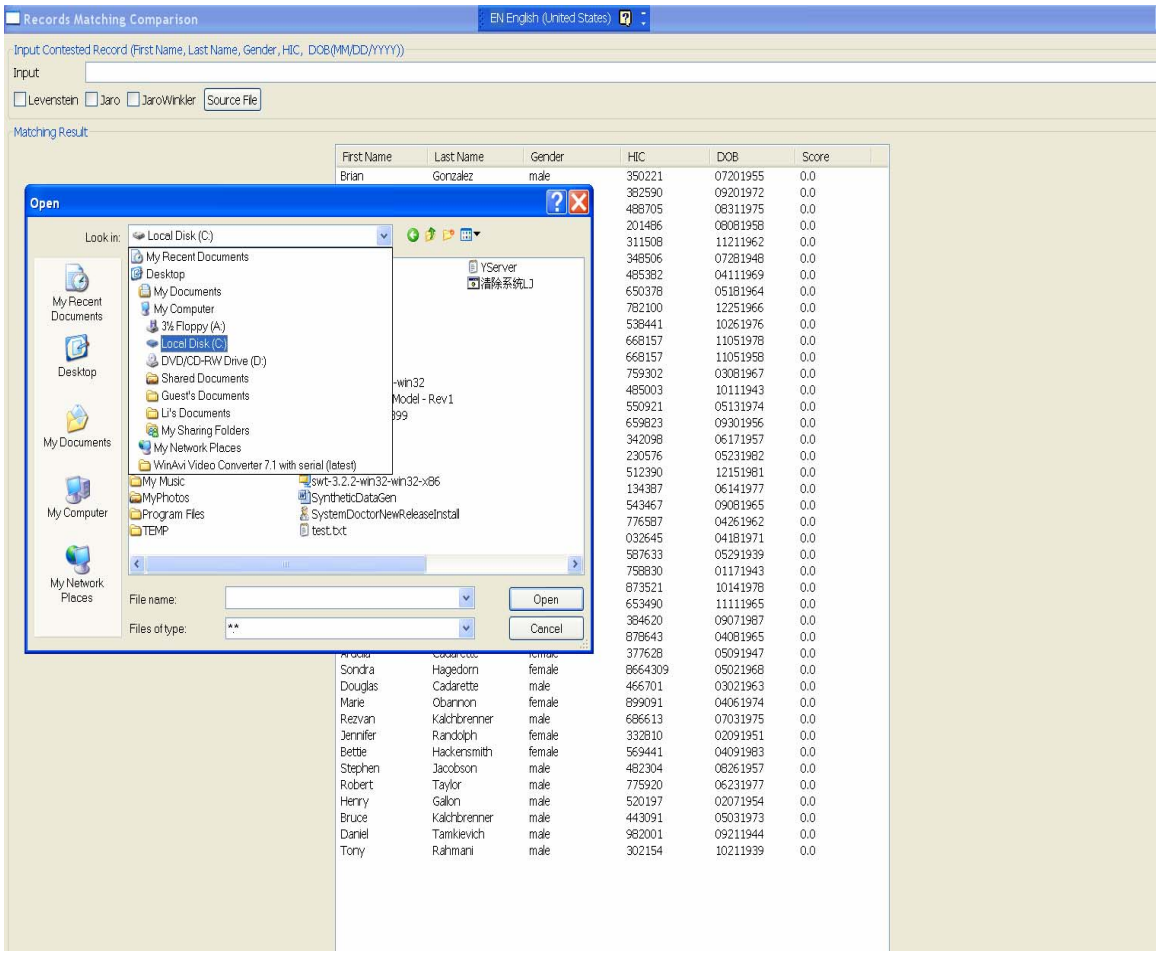


Figure 16 Screenshot of the Tool for String Similarity Function Comparisons

In the Post Enumeration Survey (PES) (Winkler and Thibaudeau 1991), about 20% of last names and 25% of first names have various typographical errors. In other words,

these names would not be matched character-by-character. So if matching is conducted by an exactly matching algorithm, more than 30% of matches are lost.

Real data is “dirty”. As mentioned above, the fields, or attributes, of last name and first name in a record contain typographic errors. The other field, such as gender, in a record, normally only has two value states, and consequently could not impact enough information to identify a unique match. In contrast, last name or first name contains much more information for matching two records, but they may frequently be recorded incorrectly. Furthermore, personal names are often used as *identifiers* to access data or when searching for people; Personal names can have several valid variations; and names are influenced by language and culture. So, personal names play a more important role in approximate record matching. Having randomly picked up last names and first names from the phone directory of Louisville Metro Area, experiments to compare the performance of various string distance functions are conducted by two categories: 1) Matching strings, first name and last name that are known to be the same but misspelled; 2) Non-Matching strings, first name and last name that are known to be different.

5.1.1 String Similarity Function Comparison for Matching Names

There are a variety of input errors for patient records, including phonetic errors, homonym errors, incorrect data entry by typographic errors, random typing errors, and reversal of first name and last name (especially in Asian names). String

similarity functions are proposed to measure the difference between the presented-contested pairs of patient records in order to create a decision model for the approximate matching approach. Different string similarity functions are studied, and their performances in the pairs of first name and last name which are randomly chosen from the phone directory of Louisville Metro Area are verified. The performances for comparing the pairs of last name and first name due to different types of errors in patient data entry, including scanning errors, typographic errors, phonetic and hetergraphic errors, random errors and reversal of first name and last name errors are evaluated

5.1.1.1 Scanning Errors

More and more technologies such as data scanning are replacing traditional manual data entry systems. Data scanning replaces manual data entry by automatically capturing the information from documents and web-based forms and feeding that data directly into the information system. With data-scanning technologies, handwriting and typed documents can be captured; data entry accuracy can be improved, and the data entry time and cost can be reduced.

There are, however, still some errors by using data scanning instead of traditional typing in due to various reasons. Some scanning errors are, for example, letter “l” and number “1”, letter “h” and letter “b”, letter “T” and number “7” etc. The similarity values *sim* for each pair of last names and first names are computed, one name in each pair has one or two scanning errors, by using the three string similarity functions,

summarized in the tables 15 and 16. These pairs of last name and first name are known to be the same but with scanning errors. The similarity value, generated by each string comparison function, is greater, which indicates the pair is closer to each other. From these two tables, it can be clearly seen that the Jaro-Winkler string

Pair No.	Last Name Pairs		Similarity Scores		
			LED	Jaro	J-W
1	Randolph	Rando1ph	.875	.917	.950
2	Galandiuk	Gu1andiuk	.778	.852	.867
3	Hackmiller	blackmiller	.818	.795	.795
4	Rahmani	Ratemani	.750	.869	.895
5	Tamkievich	7amkievicb	.800	.867	.867
6	Madorsky	Mudursky	.750	.833	.850
7	Kalchbrenner	Ka1ohbrenner	.833	.889	.911
8	Gallon	Gal1un	.667	.778	.844

Table 15 Last Name Pair Similarity Score Comparisons Due to Scanning Errors

Pair No.	First Name Pairs		Similarity Scores		
			LED	Jaro	J-W
1	Tony	7ony	.750	.833	.833
2	Charles	Char1es	.857	.905	.943
3	Margie	Murgie	.833	.889	.900
4	Daniel	Danie1	.833	.889	.933
5	Kathy	Kattey	.667	.822	.876
6	Thomas	Thumus	.667	.778	.822
7	Henry	blenry	.667	.822	.822
8	Bruce	Braoe	.600	.733	.787

Table 16 First Name Pair Similarity Score Comparisons Due to Scanning Errors

comparison function gives the highest similarity values and the Jaro algorithm gives greater similarity values than the Levenshtein edit distance metric for each pair of last name and first name except the last name pair number 3 in Table 15.

From the experiments, it can be concluded that basically the Jaro-Winkler mostly outperforms the other two string similarity functions and the Jaro algorithm outperforms the Levenshtein edit distance metric in the case of pairs of names that are to be known as the same but contain scanning errors.

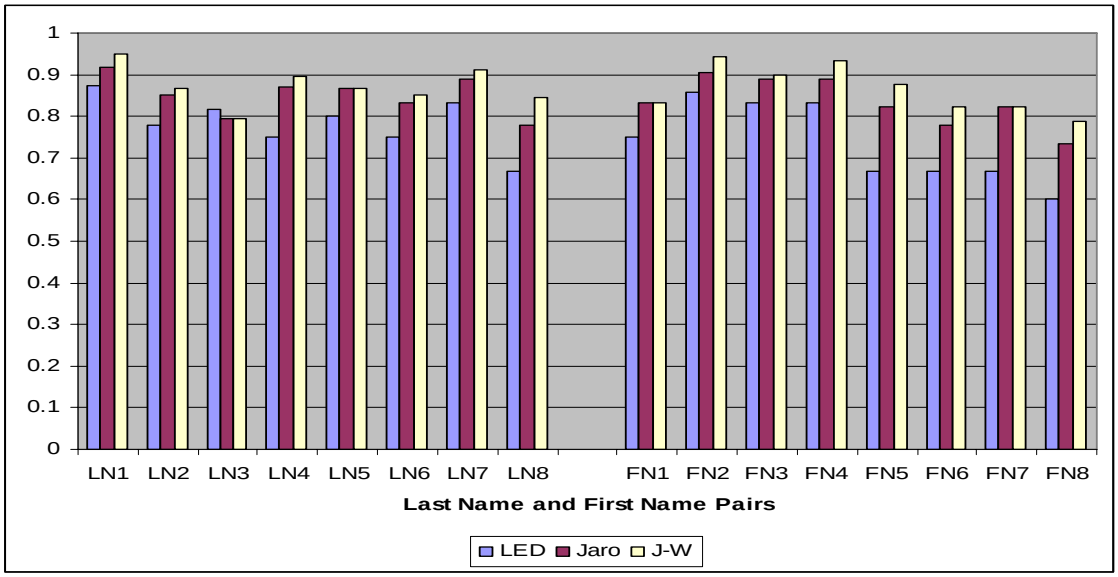


Figure 17 Personal Name Pair Similarity Score Comparisons Due to Scanning Errors

5.1.1.2 Typographic Errors

Patient data in a hospital may come from various sources. Some data come from the electronic input such as Health Care Financing Administration forms, laboratory results, or pharmacy reports. Traditional data entry is generally done on an individual personal computer. Health care information system users type in patient information into the system. Data type-in is a tedious process that is a consequent error prone. Typographic and key punch errors may occur in the data entry process. A

typographical error, typo, or fat-finger is a mistake made during the tying data entry process. The term includes errors due to mechanical failure or slips of the hand or finger, but excludes errors of ignorance. Most typos involve simple duplication, omission, transposition, or substitution of a small number of characters [Wiki1]. The typical typo errors, for examples, are key punch errors by hitting a near key in keyboard, such as “m” and “n”, “b” and “v” etc, transposition errors, such as “Taylor” and “Tyalor”, “Alexander” and “Alxenadra” etc. The last names and first names are

Pair No.	Last Name Pairs		Similarity Scores		
			LED	Jaro	J-W
1	**U**sher	**Y**sher	.800	.867	.867
2	O**b**annon	O**v**annon	.857	.905	.914
3	Na**m**aki	Na**n**aki	.833	.889	.911
4	Fank**h**auser	Fank**g**auser	.900	.933	.960
5	T**ay**lor	T**ya**lor	.667	.944	.950
6	Sm**i**th	Sm**o**t**j**	.600	.733	.786
7	Al**exan**der	Al**xenad**r**e**	.444	.884	.919
8	**J**acobs**o**n	**H**acobs**e**n	.750	.833	.833

Table 17 Last Name Pair Similarity Score Comparisons Due to Key Punch Errors

Pair No.	First Name Pairs		Similarity Scores		
			LED	Jaro	J-W
1	R**u**by	R**y**by	.750	.722	.750
2	Ro**b**ert	Ro**v**ert	.833	.889	.911
3	A**n**gela	A**m**gela	.833	.889	.900
4	Step**h**en	Step**g**en	.857	.905	.943
5	Br**ia**n	Br**ai**n	.600	.933	.947
6	Ca**ro**l	Ca**ti**l	.600	.733	.787
7	A**rd**ella	A**dr**ella	.714	.952	.957
8	Adri**en**n**e**	Adri**on**m**e**	.750	.778	.867

Table 18 First Name Pair Similarity Score Comparisons Due to Key Punch Errors

randomly chosen from the same resource as mentioned in previous section. Various human-generated typo errors are introduced in these name pairs. The same concept

and methods are used to compute the similarity values for each name pair as listed in Tables 17 and 18. From these tables, it can be clearly seen that the Jaro-Winkler string comparison function gives the highest similarity values for each pair of last names and first names and the Jaro algorithm gives greater similarity values than the Levenshtein edit distance metric except the first name pair number 1 in Table 18.

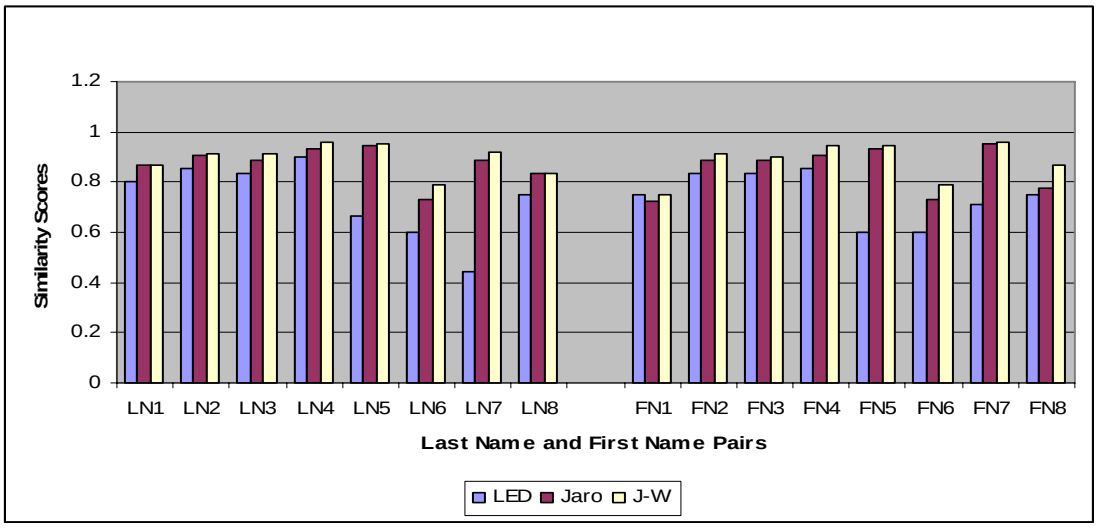


Figure 18 Personal Name Pair Similarity Score Comparisons Due to Key Punch Errors

From the experiments, the conclusion can be drawn on the basis of the evidence that basically the Jaro-Winkler mostly outperforms the other two string similarity functions, and the Jaro algorithm performs better than the Levenshtein edit distance metric in the case of pairs of names that are to be known as the same but contain typographic errors.

5.1.1.3 Phonetic and Heterographic Errors

Patient records in health care information systems come from various sources. Sometimes, hospital admission and discharge offices take patient information over the phone, or information is provided orally, phonetic characteristics of a name can be misheard. For examples, “D” versus “T”, “M” versus “N” and “B” versus “P” etc. In such a case, phonetic errors may occur. Phonetic errors refer to similar sounding spelling. Heterographs that share the same pronunciation are spelled differently [wiki2]. Heterographic examples include *to*, *too*, *two*, and *there*, *their*. There are lots of heterographic examples for names such as Stephen, Stefen and Cathy, Kathy etc.

There is another case. Some patients are originally from other countries, in which their native language is not English. Their names are originally in other languages, such as Spanish, Chinese etc. The pronunciation and spelling are so different from English. When their information is provided orally, the persons who record the information sometimes fail to spell a patient’s name correctly. In this kind of situation, the recorders usually spell it based on sounds, or spell it by default, which they think it should be. In some cases, however, it is not. So phonetic and heterographic errors may occur in such a situation. There are some other circumstances in which various spelling errors based on sounds may occur when information is provided orally. This is not going to be described in detail, because that would be out of scope of this dissertation. Some real last name and first name pairs which contain human-generated phonetic and heterographic errors and their similarity values calculated using

different string distance metrics are listed in Tables 19, 20 and Figure 19 as follow. It can be clearly seen that the Jaro-Winkler string comparison function gives the highest similarity values for each pair of last names and first names and the Jaro algorithm gives greater similarity values than the Levenshtein edit distance metric except the last name pair number 2 and 3 in Table 19.

Pair No.	**Last Name Pairs**		**Similarity Scores**		
			LED	**Jaro**	**J-W**
1	McLe**a**n	McLe**i**n	.833	.889	.933
2	Gon**z**alez	Gon**s**alez	.875	.821	.875
3	Hackm**i**ller	Hackm**e**ller	.900	.896	.938
4	Jank**ow**ski	Jank**au**ski	.778	.852	.911
5	Gallman	Gall**a**man	.875	.911	.946
6	Hageman**n**	Hageman	.875	.958	.975
7	Gabehart	Gabeh**e**art	.889	.963	.978
8	Hackensmith	Hackensmith**y**	.917	.972	.983

Table 19 Last Name Pair Similarity Score Comparisons Due to Phonetic and Heterographic Errors

Pair No.	**First Name Pairs**		**Similarity Scores**		
			LED	**Jaro**	**J-W**
1	Br**i**an	Br**y**an	.800	.867	.893
2	Rick**e**y	Ricky	.833	.944	.967
3	Margie	Mar**g**gie	.833	.889	.911
4	M**a**rty	M**u**rty	.800	.867	.880
5	Jo**h**n	Jon	.750	.917	.933
6	Je**a**nn**ie**	Jenn**y**	.571	.790	.832
7	Teresa	T**h**eresa	.857	.952	.957
8	Bett**ie**	Bett**y**	.667	.822	.893

Table 20 First Name Pair Similarity Score Comparisons Due to Phonetic and Heterographic Errors

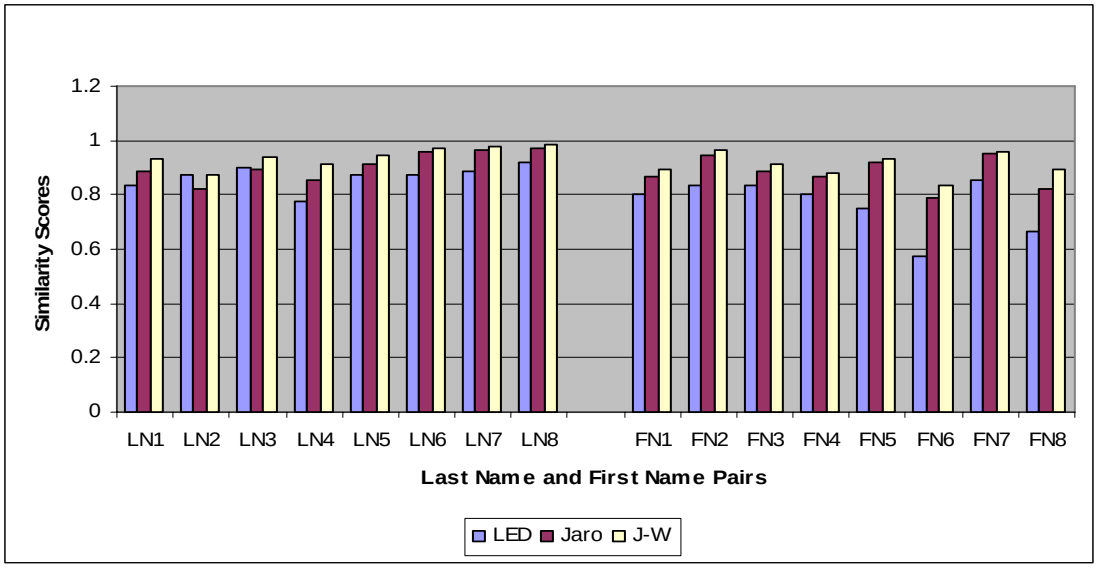


Figure19 Personal Name Pair Similarity Score Comparisons Due to Phonetic and Heterographic Errors

A conclusion can be drawn from the experiments, basically the Jaro-Winkler outperforms the other two string similarity functions, and the Jaro algorithm performs better than the Levenshtein edit distance metric in most of random pairs of names in the case of pairs of names that are to be known as the same but contain phonetic and heterographic errors.

5.1.1.4 Random Errors

Data entry is the stage of entering data from documents into the information system. This is a tedious process that is consequently error prone. Data entry error is usually random. The term random here refers to data entry errors that could be any types of errors which have been described in previous sections. And it also means the errors may occur in any positions of a string. The experiments are conducted in the following steps:

1. Randomly choose the last name and first name from the same resource as mentioned in the previous sections.

2. Use Random Number Generator Pro [RNGP] to generate the position in the strings where a spelling error occurs.

3. Use Random Number Generator Pro to generate the random number between 1 and 26 exclude duplicate numbers for a replacement letter look-up table as shown in Table 21.

4. Generate errors by replacing an original letter in the string with the letter from the letter replacement look-up table

5. Compute the similarity values for each pair of names using different string similarity functions listed in Tables 22 and 23.

Table 21 Random Error Generation Letter Replacement Look-up Table

I	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
II	a	b	c	d	e	f	G	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
III	1	12	9	8	13	11	7	3	16	17	14	24	5	19	18	4	15	25	21	6	23	20	22	10	26	2
IV	a	L	i	h	m	k	G	c	p	d	n	x	e	s	r	d	o	y	u	f	w	t	v	j	z	b

I. Numbers in order
II. Letters in alphabet order
III. Random number generator by RNGP
IV. Letter replacement look-up based on the random number

Pair No.	Last Name Pairs		Similarity Scores		
			LED	Jaro	J-W
1	Rose**n**barger	Rose**s**barger	.909	.939	.964
2	Hor**w**itz	Hor**v**itz	.857	.905	.933
3	G**r**eathouse	G**y**eathouse	.900	.933	.940
4	S**t**ooksberry	S**f**ooksberry	.909	.939	.945
5	**B**a**h**adori	**L**a**c**adori	.750	.833	.833
6	M**c**G**o**hon	M**i**G**r**hon	.714	.743	.769
7	**D**omn**w**achukwu	**H**omn**v**achukwu	.833	.789	.789
8	Q**u**i**s**enberry	Q**w**i**u**enberry	.818	.906	.915

Table 22 Last Name Pair Similarity Score Comparisons Due to Random Errors

Pair No.	First Name Pairs		Similarity Scores		
			LED	Jaro	J-W
1	Cec**e**lia	Cec**m**lia	.857	.905	.933
2	Adol**p**h	Adol**d**h	.833	.889	.933
3	C**y**nthia	C**z**nthia	.857	.905	.914
4	Gord**o**n	Gord**r**n	.833	.889	.933
5	Mar**ga**ret	Mar**ga**ret	1.000	1.000	1.000
6	D**z**e**m**all	D**bee**all	.714	.810	.829
7	**Ro**nnie	**Yr**nnie	.667	.889	.889
8	R**a**ym**o**nd	R**a**ym**r**nd	.857	.905	.943

Table 23 First Name Pair Similarity Score Comparisons Due to Random Errors

From these tables, it can be seen that the similarity values, which are generated by the Jaro-Winkler Method, are the greatest, except the pair number 7 in the last name Table 22. And the Jaro algorithm gives higher similarity values than the Levenshtein edit distance metric, except the pair number 7 in last name Table 22.

From the experiments, as shown in Figure 20, it can be concluded that basically the Jaro-Winkler outperforms the other two string similarity functions in most pairs of

name pairs, and the Jaro algorithm performs better than the Levenshtein edit distance metric in the case of pairs of names that are to be known as the same but contain random errors.

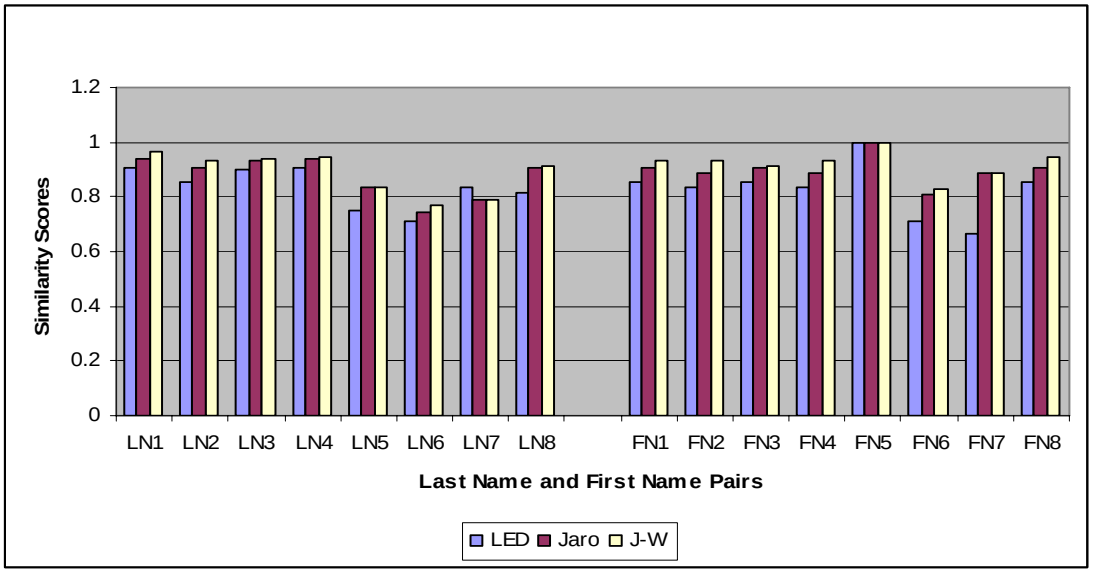


Figure 20 Personal Name Pair Similarity Score Comparisons Due to Random Errors

5.1.1.5 Reversal of First Name and Last Name

From the literature, many of researchers point out that there are personal name variations due to various reasons. Bell and Sethi [2001] concluded that there are a variety of data entry errors. In the medical record databases and National Medical Patient Index, there are common data entry errors which involve reversal of patient's last and first name, especially for patients who are originally from Asia, such as China, Korea etc. Actually, reversal of last and first name not only exists in patient's record databases, but also in the other databases as well. The experiments which have been conducted are described below to evaluate the performance of string similarity

functions in the case of reversal of last and first name. The software tool which has been developed is used to compute each pair of names (last and first name) and their reversals of last and first name, and the similarity values of each pairs are presented in Table 24. The objective of record matching algorithm is to identify those pairs of names which represent the same person. As be described in previous sections, the similarity value equals 1, which indicates the pair is exactly the same (matching character by character); while it equals 0, which means the pair is totally different. So the closer to 1 the similarity value generated by each of the string distance metrics,

Pair No.	Name Pairs		Similarity Scores		
			LED	Jaro	J-W
1	Wakefield Charles	Charles Wakefield	0	.689	.689
2	Usher Alan	Alan Usher	0	0	0
3	Dallenbach Fred	Fred Dallenbach	.286	.700	.700
4	Eckensels Thomas	Thomas Eckensels	.067	.640	.640
5	Obannon Levelle	Levelle Obannon	0	0	0
6	Fankhauser Mary	Mary Fankhauser	.286	.757	.757
7	Gablejic Dzemal	Dzemal Gablejic	0	.728	.728
8	Singerman Lisa	Lisa Singerman	.231	.799	.799
9	Hagedorn Joseph	Joseph Hagedorn	0	.671	.671
10	Rahmani Rezvan	Rezvan Rahmani	.385	0	0
11	Abrams Adrienne	Adrienne Abrams	.143	.728	.755
12	Wurtenberg Oscar	Oscar Wurtenberg	.200	.667	.667
13	Bahadori Farideh	Farideh Bahadori	.200	0	0
14	Lohmeyer William	William Lohmeyer	0	0	0
15	Rosenbarger Earl	Earl Rosenbarger	.333	.803	.803
16	Naftaliyeva Khana	Khana Naftaliyeva	.250	0	0

Table 24 Similarity Scores Comparison Due to Reversal of Last Name and First Name

the better. From Table 24 and Figure 21, it can be seen that the Jaro algorithm and the Jaro-Winkler method give most of pairs higher values than the Levenshtein edit distance function, while they give pair No. 2, 5, 10, 13, and 16 value of 0. On the

other hand, the Levenshtein edit distance function gives pair No. 10, 13 and 16 higher similarity values than the other two string comparator functions.

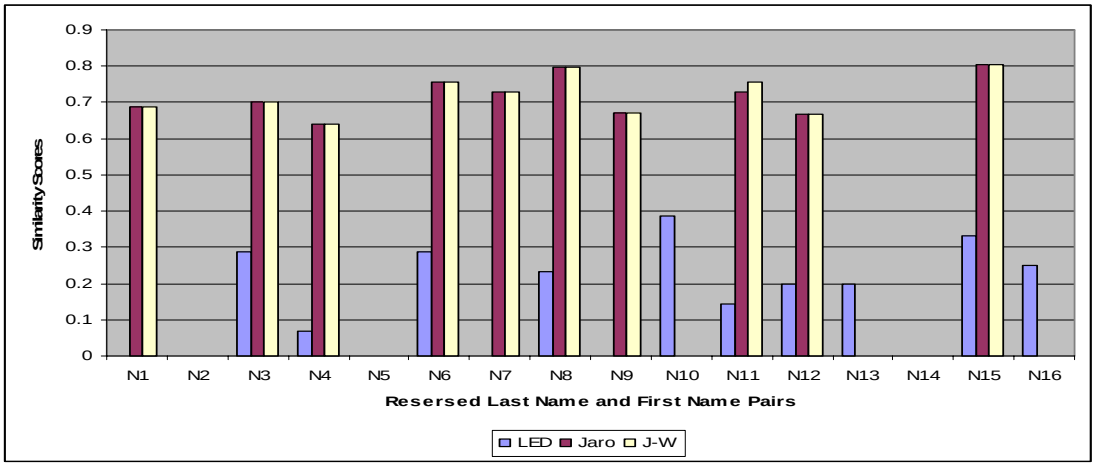


Figure 21 Personal Name Pairs Similarity Score Comparisons
Due to Reversal of Last Name and First Name

So from the experiment, it can be concluded that in the case of swapped last and first name, no string distance metric is definitely superior over the others.

5.1.2 String Similarity Function Comparison for Non-Matching Names

In Section 5.1.1, experiments have been conducted and described to evaluate the performance of the three string similarity functions, the Levenshtein edit distance, the Jaro algorithm, and the Jaro-Winkler method, for last and first name pairs that are known to be the same but misspelled due to various reasons. In this section, an experiment is going to be conducted and presented to compare the performance of these three string comparator functions for pairs of last and first names that are known to be different, non-matching names.

A list of pairs of real last and first name is taken from the same resource as previous experiments. It is wanted to make sure that each pair of last and first name is different. The name pairs are listed in Tables 25 and 26 as follows:

Pair No.	Last Name Pairs		Similarity Scores		
			LED	Jaro	J-W
1	Gaba	Obannon	.142	.429	.429
2	Fankhauser	Madorsky	.200	.458	.458
3	Randolph	Shaughnessy	.091	.438	.438
4	Abrams	Gallaway	.025	.528	.528
5	Domnwachukwu	Tanselle	.083	.306	.306
6	MacLean	Kalchbrenner	.333	.644	.644
7	Eckensels	Cadarette	.111	.556	.556
8	Quisenberry	Hagedorn,	.182	.438	.438

Table 25 Similarity Score Comparisons for Non-Matching Pairs of Last Name

Pair No.	First Name Pairs		Similarity Scores		
			LED	Jaro	J-W
1	Raymond	Jennifer	.000	.423	.423
2	Levelle	Marie	.143	.448	.448
3	Rezvan	Thomas	.167	.444	.444
4	William	Douglas	.143	.524	.524
5	Kathy	Margie	.167	.456	.456
6	Charles	Joan	.143	.464	.464
7	Ardella	Khana	.143	.562	.562
8	Oscar	Sondra	.167	.656	.656

Table 26 Similarity Score Comparisons for Non-Matching Pairs of First Name

The same methodology is used as in previous experiments in Section 5.1.1. In order to compare the performance of these string edit distance metrics, all of them are applied to all pairs of last and first name. The software comparison tool is used to

compute the similarity values for each pair of names using each string comparator function. These similarity values are shown in Tables 25, 26 and Figure 22. From these two tables, it can be seen that the similarity values, generated by the Jaro, and the Jaro-Winkler algorithm, are the same and greater than the values generated by the Levenshtein edit distance metric. In the other words, the degree of similarity between each pairs is greater if the Jaro and the Jaro-Winkler algorithm are applied. As

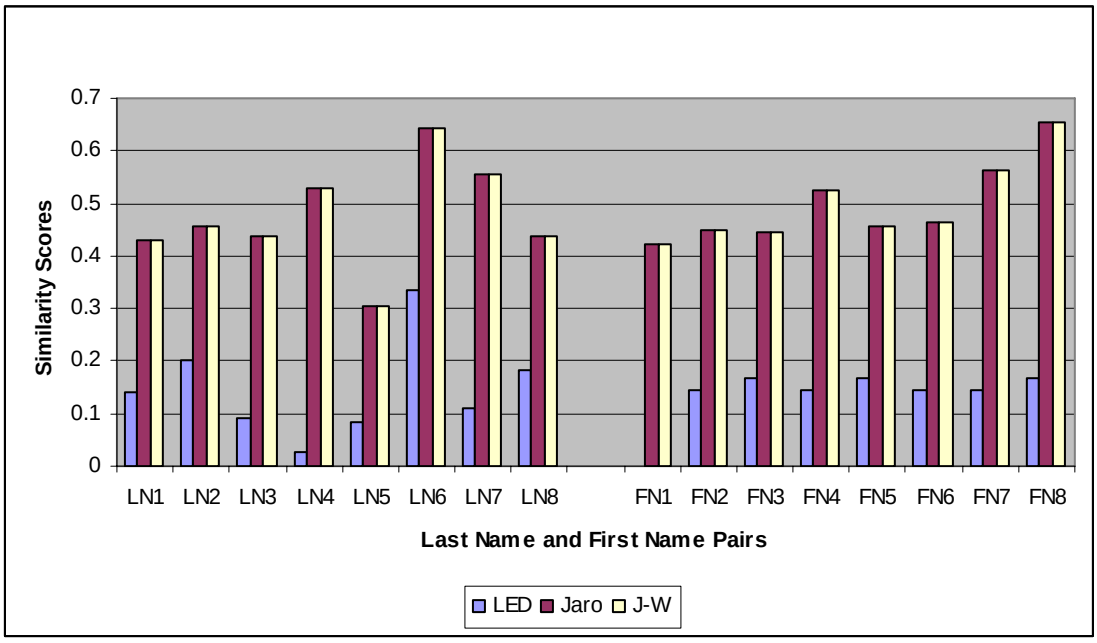


Figure 22 Similarity Score Comparisons for Non-Matching Pairs of Personal Names

mentioned above, these pairs of last and first name are different. So the performance of the Jaro and the Jaro-Winkler functions is not as good as if of the Levenshtein edit distance metric.

It can be concluded, from the experiments which have been conducted, that in the case of pair of names that are different from each other, or say non-matching name

pairs, the Levenshtein edit distance metric outperforms over the other string comparator functions.

5.1.3 Conclusion

Extensive experiments were conducted to compare string similarity functions in two categories, 1) Matching strings, first name and last name that are known to be the same but misspelled; 2) Non-Matching strings, where first name and last name that are known to be different. For the misspelled names, i.e. those in the first category, human-generated errors were introduced based on various errors, such as scanning errors, phonetic and heterographic errors, typographic errors and random errors.

Conclusions can be drawn from the experiments, the Jaro-Winker and the Jaro string similarity functions mostly outperform over the Levenshtein edit distance metric for matching strings that are misspelled, while they are not as good as the Levenshtein edit distance metric for non-matching strings. Hence, overall, no single string similarity function outperforms over the others in all circumstances. If the Jaro or the Jaro-Winkler algorithms are applied, more record pairs appear to be linked correctly (true positive), however, there are more type I errors, the number of record pairs linked incorrectly (false positive). While, if the Levenshtein edit distance metric is applied it appears to identify more record pairs unlinked correctly (true negatives), however, it appears to increase type II errors.

5.2 Decision Model Comparison Using Different Comparison Vectors

In Section 5.1, extensive experiments have been conducted and described to evaluate the performance of the three string comparators using pairs of first and last name, which are typically as identities for matching patient records in multiple databases, with different types of human-generated errors. In this section, decision models constructed by different comparison vectors are compared to evaluate their matching results using type I and type II errors and the other matching quality metrics such as precision, recall, F-measure etc. In order to conduct experiments to compare decision models, datasets [febrl-0.4] are downloaded. Basically, these datasets contain from 1,000 to 10,000 records in each dataset. Each record has attributes such as record number, last name, first name, address, date of birth, phone number etc. Also some data pre-processing programs, attached in Appendices 5-14, have been developed to generate errors in the datasets, and comparison vectors. So that these generated comparison vectors can be used as input data to generate decision models using induction learning algorithm. In the following sections, test data sets with human-generated errors, comparison vectors constructions, decision model generation, prediction quality of models and performance comparison using matching quality metrics are described in details.

5.2.1 Test Data Sets and Comparator Vectors Generation

One of supervised learning algorithms, decision induction learning, is applied in this research to address patient record matching problem. Decision trees attempt to find a

strong relationship between input values and target values in a group of observations that form a data set. When a set of input values is identified as having a strong relationship to a target value, then all of these values are grouped in a bin that becomes a branch on the decision tree. So the decision model can be used to predict the target value from unseen datasets. In the other words, training datasets, which contain attributes such as last and first name, date of birth, gender etc, and records matching status, are needed for supervised learning, and matching rules generation. What is needed is a collection of real test data sets, which can be used as a standard test bed for developing and comparing algorithms. However, due to privacy and confidentiality issues it is unlikely that such a data will ever become publicly available. An alternative is the use of artificially generated data sets instead. They have advantages that the amounts of errors introduced, as well as the matching status of record pairs, are known [Christen and Churches].

Two datasets [febrl-0.4], dataset_A_10000.csv and dataset_C_10000.csv, are downloaded from the internet. Each data set has totally different 10,000 records and identical attributes such as record number, last name, first name, address, date of birth, phone number etc. It has been proposed that last name, first name, gender, date of birth and HIC are as identifiers for patient records matching in this research. There is no an attribute, gender, in these downloaded datasets. Gender is then just randomly assigned as male or female to each of the records in dataset_A_10000.csv and dataset_C_10000.csv.

Then a duplicate data set dataset_A_1_10000.csv is generated. Human-generated errors are introduced in data set dataset_A_1_10000.csv using developed data pre-processing programs (Appendices 5-14). In Section 5.1, extensive experiments have been conducted to evaluate the three string comparators, the Levenshtein edit distance, the Jaro algorithm, and the Jaro-Winkler method, based on pairs of first and last name with various errors: scanning errors, typographic errors, phonetic and heterographic errors, and random errors. The similar results due to these types of errors are obtained. So to simplify, the random errors are introduced in the data set dataset_A_1_10000.csv. The error generation rules are:

1) For each record in the dataset, at least one character of an attribute is replaced by a random character.

2) For each attribute, at most two characters are replaced by a random character.

Now there are two pairs of datasets, dataset_A_10000.csv versus dataset_A_1_10000.csv, and dataset_A_10000.csv versus dataset_C_10000.csv. The first pair of datasets is known to be the same but misspelled in one or more attributes of each records; while the second is known to be different. So the matching status of record pairs is known.

The next step of the experiment is to generate comparison vectors using these three string comparison functions: the Levenshtein edit distance, the Jaro algorithm, and the Jaro-Winkler method for both pairs of datasets using developed data pre-processing programs (Appendices 5-14) and assign a class attribute 1 as matched for records from the first pair datasets, 0 as non-matched for records from the second pair

datasets (Recall the matching status of these two pairs of data sets are known). An example of generated comparison vectors for both matched and not-matched records is listed in Tables 27 and 28:

Pairs	Last Name	First Name	Gender	DOB	HIC
Matched	Horwitz	Cynthia	Female	19750815	778579369
	Horvitz	Cznthia	Femela	19781115	798549266
Not Matched	Abrams	William	Male	19350321	256400811
	Gallaway	Douglas	Malle	19681028	621700571

Table 27 an Example of Matched and Non-Matched records

Pairs	Comparison Vectors		
	LED	**Jaro**	**Jaro-Winkler**
Matched	(.86 .86 .67 .63 .56 1)	(.91 .91 .94 .78 .67 1)	(.93 .91 .96 .84 .70 1)
Not-Matched	(.03 .14 .80 .38 .33 0)	(.53 .52 .93 .68 .72 0)	(.53 .52 .95 .74 .72 0)

Table 28 an Example of Generated Comparison Vectors

As described above, from those two pairs of data sets, dataset_A_10000.csv versus dataset_A_1_10000.csv and dataset_A_10000.csv versus dataset_C_10000.csv, each of the three string comparator functions is applied to generate two comparison vectors with a class attribute 1 or 0. These two comparison vectors are combined to one vector which contains both class 1 as matched and class 0 as not-matched. So there are three comparison vectors, one generated using each of the three string comparator functions. These three comparison vectors are as data sets to construct three predictive models using supervised learning technique in SAS Enterprise Miner 5.3 environment. The decision model generation is described in detail in following sections.

5.2.2 Decision Model Generation Using SAS Enterprise Miner 5.3

In Section 5.2.1, the process of comparison vector generations have been described in detail. So at this point, three comparison vectors which are generated by using the three string comparison functions from the original downloaded data sets are available. Each comparison vector consists of 19090 records, 5 attributes which are similarity measurements of gender, first name, last name, date of birth, and health insurance code (HIC) from original pairs of records, and a class attribute which is the matching status indicator, 1 as matched and 0 as not-matched. The sample of the data set is as in Table 29.

gender	given_name	surname	date_of_birth	hic	matching_status
0.89	0.83	0.93	0.92	0.74	1
1.00	1.00	1.00	0.82	1.00	1
1.00	0.00	0.00	0.00	0.00	0
0.89	1.00	1.00	0.78	0.95	1
0.83	0.87	0.78	0.92	0.81	1
0.78	0.81	1.00	1.00	0.90	1
0.72	0.70	0.47	0.75	0.62	0
1.00	0.47	0.52	0.68	0.00	0
0.67	0.78	0.78	0.87	1.00	1
0.72	0.50	0.50	0.68	0.71	0
1.00	0.46	0.42	0.58	0.62	0
0.72	0.50	0.66	0.68	0.00	0
0.83	0.89	0.87	0.83	0.85	1
1.00	0.52	0.54	0.00	0.63	0
1.00	0.90	1.00	1.00	1.00	1

Table 29 A Sample of Comparison Vector Generated Using Jaro String Comparison Function

All attributes are numeric value (interval variables), except the class attribute, which is a binary variable in the source data sets. SAS Enterprise Miner 5.3 is used for decision tree predictive model generation. According to Ville [2006], the decision tree growing process using SAS Enterprise Miner can be illustrated as in Figure 23

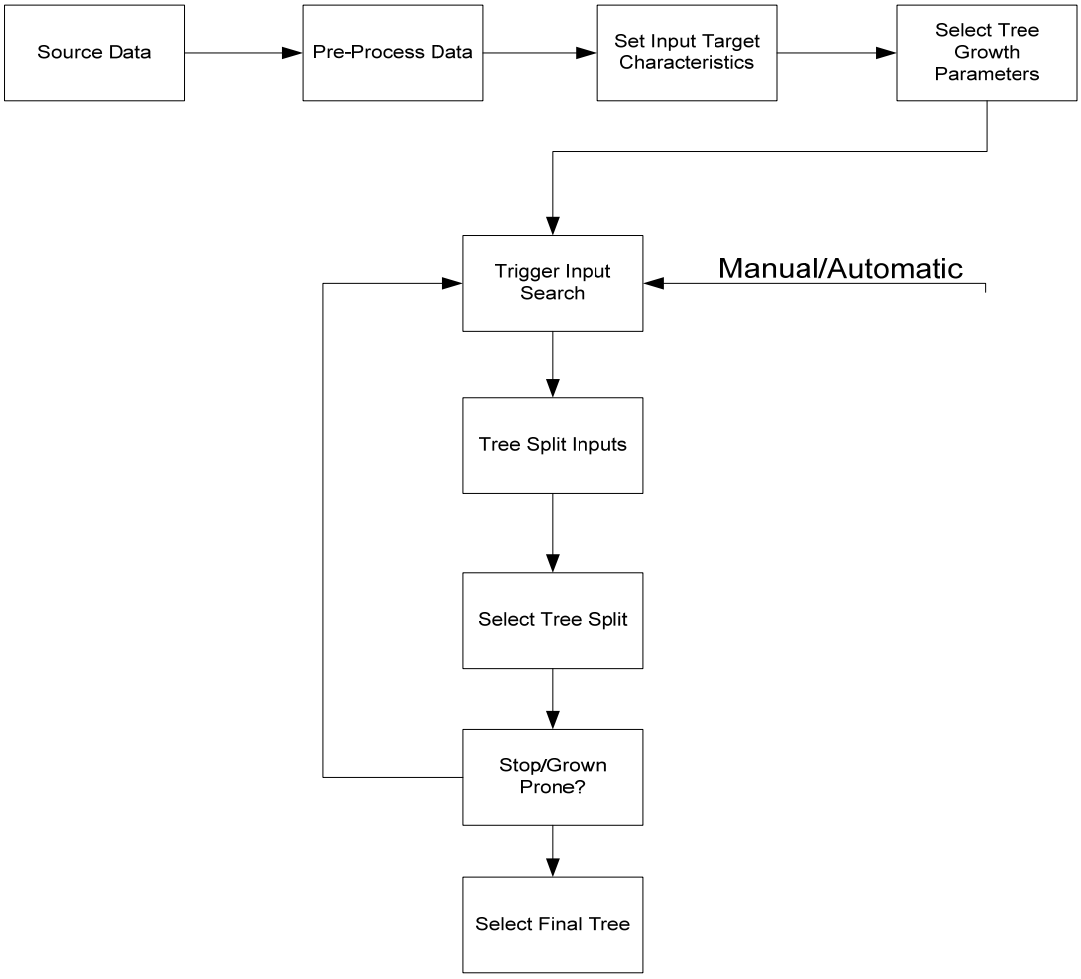


Figure 23 Processes in Decision Tree Generation in SAS Enterprise Miner [Ville]

These steps are performed in sequence, with the development of each layer of branches of the decision tree. The decision tree process, Tree Split Inputs, and Select Tree Splits, is an iterative process.

In order to get started with SAS Enterprise Miner, those source data is needed to read into SAS. There are two steps involved in reading source data into SAS. The first step is to tell SAS where it can both find and write SAS datasets. This is accomplished by

creating user-defined libraries. The second step is to determine the file format in which my own data is stored. This process is accomplished by using Enterprise Guide 4.1. As described in Section 5.2.1, the source data files, the comparison vectors in common-separated values format (CSV format), are generated using the developed programs and stored in my local drive. Some data pre-processing has also been done before SAS Enterprise Miner 5.3 lunched. So the data sets have been defined and introduced into the data mining environment, SAS Enterprise Miner 5.3. Once the data is available in SAS, the attributes of the data source can be displayed by using the StatExplore node in SAS Enterprise Miner. An example of diagnostic summary of the attributes as illustrated in the output as in Table 30. The details are in appendix 28-30.

Variable	ROLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
date_of_birth	INPUT	0.66092	0.35052	19090	0	0	0.78	1
gender	INPUT	0.87552	0.12940	19090	0	0	0.89	1
given_name	INPUT	0.61958	0.34816	19090	0	0	0.70	1
hic	INPUT	0.67934	0.30441	19090	0	0	0.74	1
surname	INPUT	0.61339	0.35394	19090	0	0	0.70	1

Table 30 Interval Variable Summary Statistics for Comparison Vector (Jaro)

In decision tree models, one of the fields of data set serves as the target of analysis. Other fields are defined as inputs that can be used to predict this target of analysis. The matching status is the target, which is binary attribute. The other fields, such as last and first name, date of birth, gender, and HIC, are interval inputs. The decisions are made in order to minimize misclassification error in this research project. So the Decision Tree node is selected, and then under Subtree in the properties panel of the Decision Tree node, the Method property is set to Assessment and the Assessment Measure property is set to Misclassification. The maximum number of branches is set

to 2, because it's a binary tree. In SAS Enterprise Miner, the splitting method used for partitioning the data is determined by the value of the Criterion property in the Splitting Rule section of the properties panel. This Criterion is set to ProbChisq in SAS Enterprise Miner. The Stopping Rules are also set up to control tree growth through the Threshold significance Level, the leaf Size Property and Maximum Depth Property. Decision tree models are created in SAS Enterprise Miner as figure 24

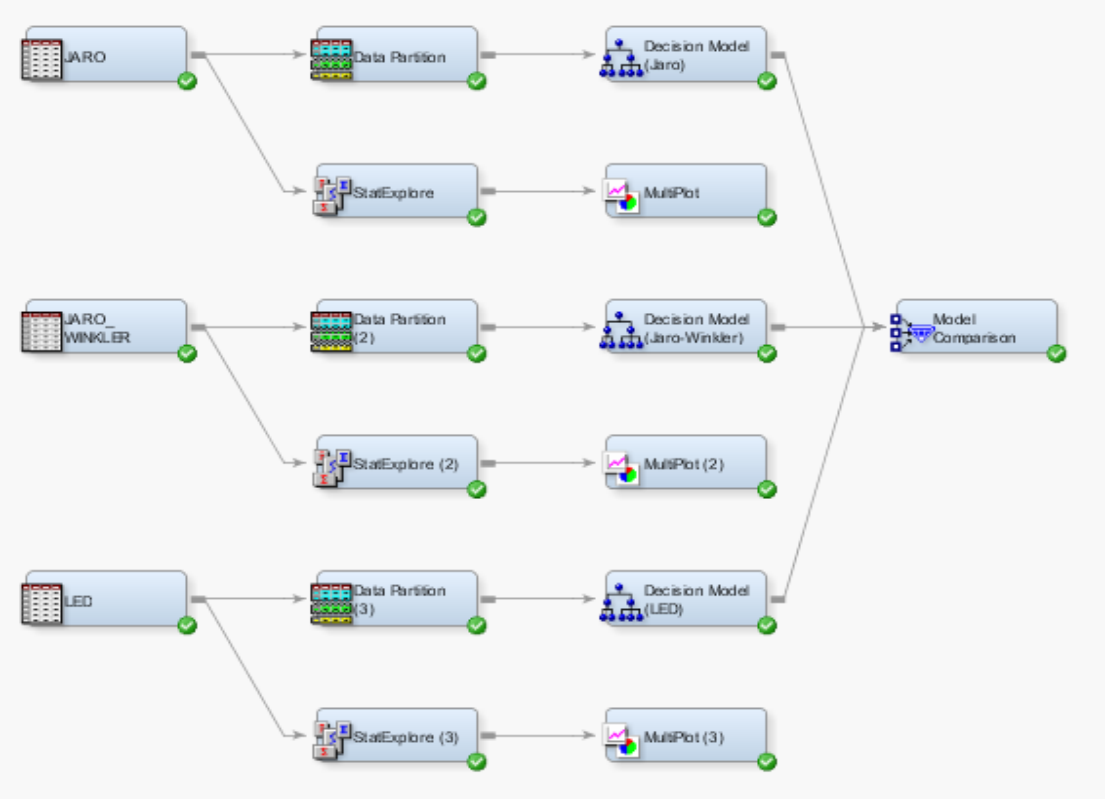


Figure 24: Decision Tree Models and Comparison Created in SAS Enterprise Miner 5.3

5.2.3 Results and Performance Comparison

Matching models are developed using SAS Enterprise Miner 5.3. The models are based on comparison vectors generated from the three string comparison functions.

Data sets are partitioned 55% as a training data set, and 45% as a validation data set for each decision model. The training data set is used for developing the node definitions and posterior probabilities, and the validation data set is used for pruning the tree to find the optimal tree. After running the Decision Tree node in SAS Enterprise Miner, the Results Window yields the optimal tree and the corresponding rules. An example of decision tree is shown in Figure 25. The decision trees of these models are attached in Appendices 15-20.

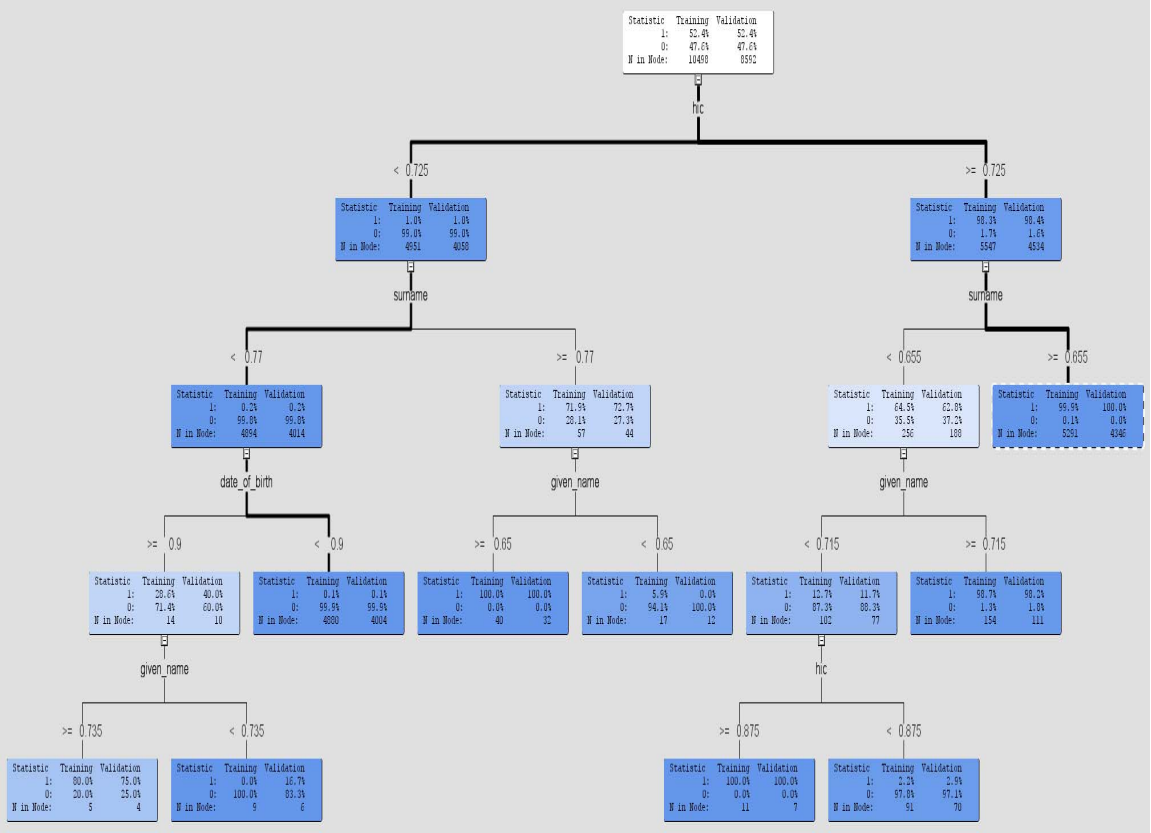


Figure 25 Decision Tree Generated by Comparison Vector Using the Jaro String Metrics

The decision rules for this decision tree are also generated as follows:

IF 0.655 <= surname

```
AND        0.725 <= hic
THEN
  NODE    :       7
  N       :    5291
  1       :   99.9%
  0       :    0.1%

IF  date_of_birth <          0.9
AND surname <         0.77
AND hic <        0.725
THEN
  NODE    :       8
  N       :    4880
  1       :    0.1%
  0       :   99.9%

IF  given_name <         0.65
AND         0.77 <= surname
AND hic <        0.725
THEN
  NODE    :      10
  N       :      17
  1       :    5.9%
  0       :   94.1%

IF          0.65 <= given_name
AND         0.77 <= surname
AND hic <        0.725
THEN
  NODE    :      11
  N       :      40
  1       :  100.0%
  0       :    0.0%

IF         0.715 <= given_name
AND surname <        0.655
AND         0.725 <= hic
THEN
  NODE    :      13
  N       :     154
  1       :   98.7%
  0       :    1.3%

IF  given_name <        0.735
AND          0.9 <= date_of_birth
AND surname <         0.77
```

```
AND hic <        0.725
THEN
  NODE    :      18
  N       :       9
  1       :    0.0%
  0       :  100.0%

IF         0.735 <= given_name
AND          0.9 <= date_of_birth
AND surname <         0.77
AND hic <        0.725
THEN
  NODE    :      19
  N       :       5
  1       :   80.0%
  0       :   20.0%

IF         0.725 <= hic <        0.875
AND given_name <        0.715
AND surname <        0.655
THEN
  NODE    :      20
  N       :      91
  1       :    2.2%
  0       :   97.8%

IF         0.875 <= hic
AND given_name <        0.715
AND surname <        0.655
THEN
  NODE    :      21
  N       :      11
  1       :  100.0%
  0       :    0.0%
```

The result of a decision tree induction model generated by comparison vectors from Levenshtein Edit Distance is shown as in Table 31 below

	PREDICTED MATCHED		
ACTUAL MATCHING		Matched	Not Matched
	Matched	10000	0
	Not Matched	1	9089

Table 31 Results of the LED Model

The result of a decision tree induction model generated by comparison vectors from Jaro String Distance Function is shown as in Table 32 below

	PREDICTED MATCHED		
ACTUAL MATCHING		Matched	Not Matched
	Matched	9987	13
	Not Matched	14	9076

Table 32 Results of the Jaro Model

The result of a decision tree induction model generated by comparison vectors from Jaro-Winkler Method is shown as in Table 33 below

	PREDICTED MATCHED		
ACTUAL MATCHING		Matched	Not Matched
	Matched	9985	15
	Not Matched	22	9068

Table 33 Results of the Jaro-Winkler Model

The quality metrics, accuracy, precision, recall, and F-measure for each model are also computed and summarized in Table 34

	Accuracy	**Precision**	**Recall**	**F-Measure**
LED	99.99%	99.99%	100%	100%
Jaro	99.86%	99.86%	99.87%	99.87%
Jaro-Winkler	99.81%	99.78%	99.85%	99.82%

Table 34 Matching Results Comparison for the Three Models

The random error generation rules are then modified in order to produce more random errors in the dataset. The modified error generation rules are:

1) For each record in the dataset, at least two characters of an attribute are replaced by a random character.

2) For each attribute, at most four characters are replaced by a random character.

By using these new error generation rules, new comparison vectors are generated and the experiment are repeated. The quality metric results of the three models for this dataset are shown in Table 35 below.

	Accuracy	Precision	Recall	F-Measure
LED	99.93%	99.96%	99.91%	99.86%
Jaro	99.54%	99.69%	99.44%	99.56%
Jaro-Winkler	99.34%	99.54%	99.20%	99.37%

Table 35 Matching Results Comparison for the Three Models

From these matching result comparison summary tables, all three string comparison functions perform very well as measures of similarity of pairs of attributes of records. And there are no significant differences between these three string comparison functions.

From the experiments, on the decision models generated using comparison vectors obtained using the three similarity methods, the decision tree model obtained via the Levenshtein Edit Distance Metric performs slightly better than models generated by the other two metrics, in the accuracy, precision recall and F-measure. However, based on the ROC index and other performance metrics such as average squared error, misclassification rate etc., and the comparison statistics for the three models using SAS are summarized in Tables 36 and 37. The SAS Enterprise Miner model comparison selected the Levenshtein Edit Distance Model for the first data set, and the Jaro String Comparison function model for the modified data set. The three model comparisons based on ROC index, and cumulative lift for these two data sets

generated by the SAS are shown in Figures 26 and 27, and Figure 28 and 29, respectively. The output of the models and detailed model comparison reports are attached in Appendices 21-27. Hence, based on these experiments, it is recommended that a variety of string comparison functions be used in the framework presented to identify matching records in a variety of databases.

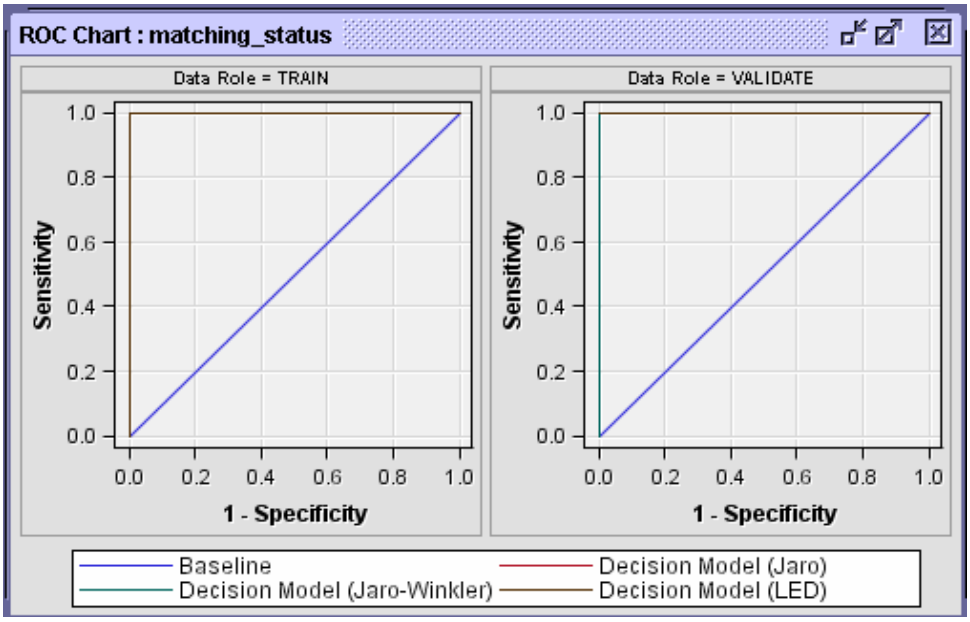


Figure 26 The Three Model Comparison ROC Curves for the First Data Set

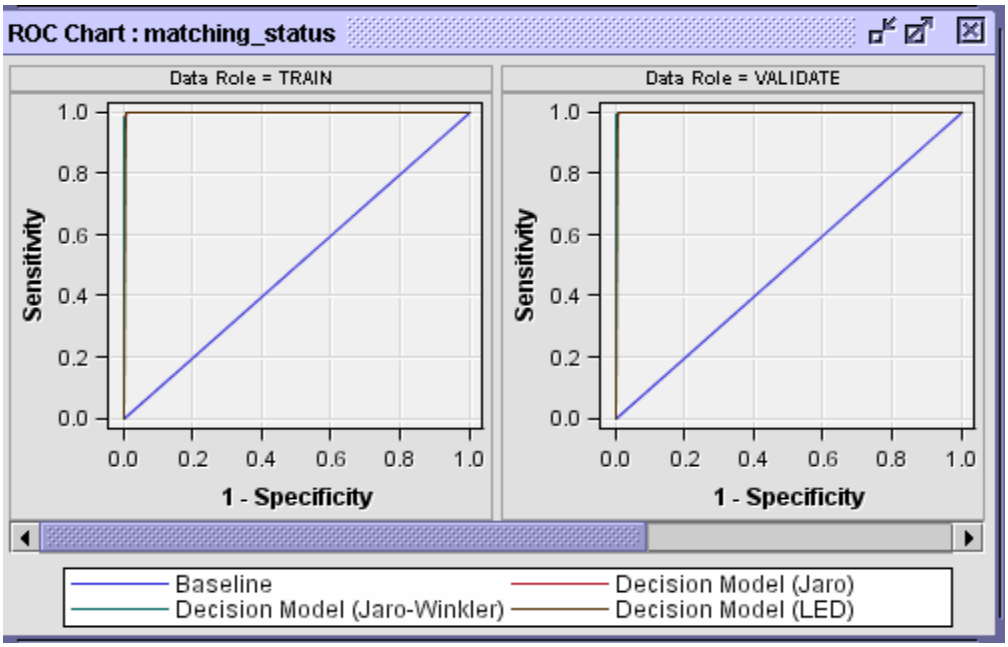


Figure 27 the Three Model Comparison ROC Curves for the Modified Data Set

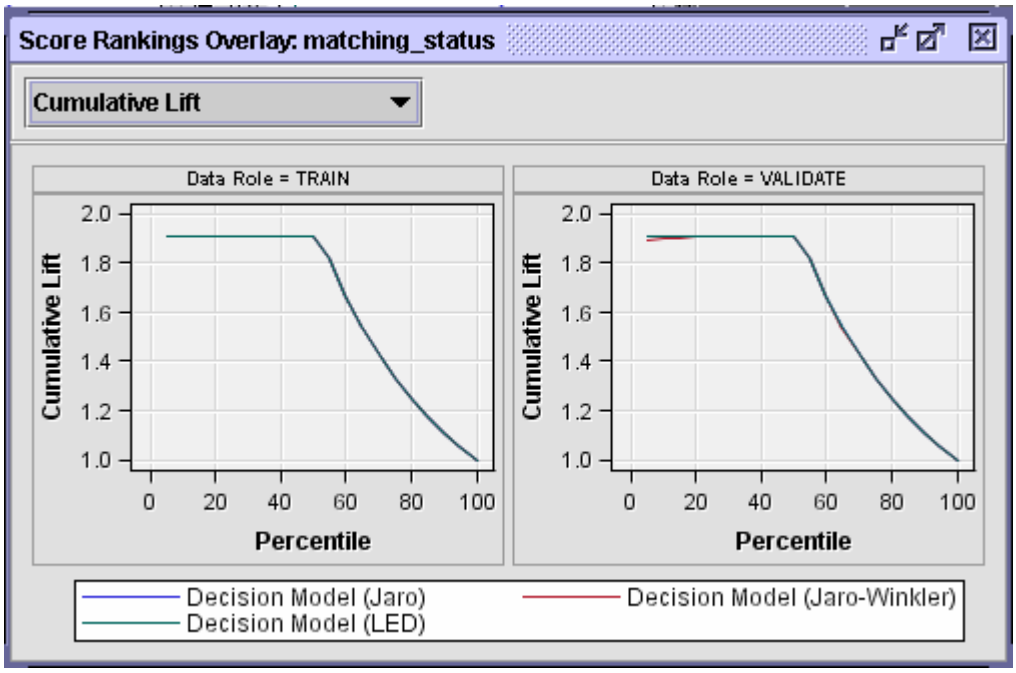


Figure 28 the Three Model Comparison Cumulative Lift for the First Data Set

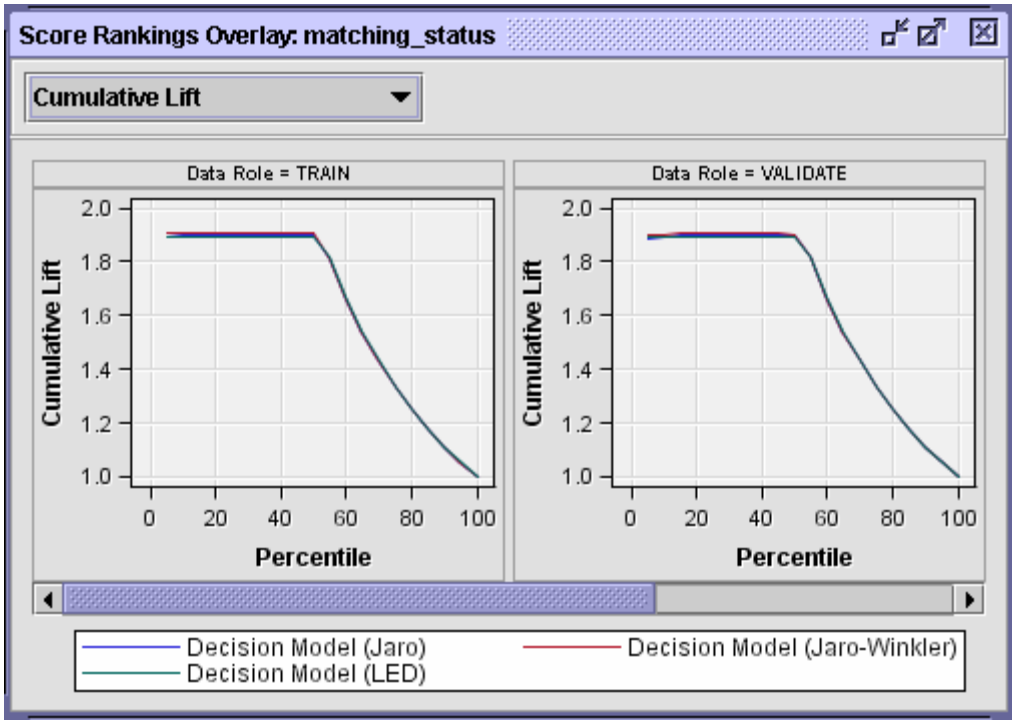


Figure 29 the Three Model Comparison Cumulative Lift for the Modified Data Set

DataRole	Target	STAT	LABEL	Tree3(L)*	Tree2(JW)	Tree(J)
Train	matching_status	KS	Train: Kolmogorov-Smirnov Statistic	0.9998	0.996327	0.996927
Train	matching_status	_APROF_	Train: Average Profit for matching_status	0.999905	0.99819	
Train	matching_status	_ASE_	Train: Average Squared Error	9.52E-05	0.001802	0.001492
Train	matching_status	_AUR_	Train: Roc Index	0.9999	0.998457	0.999006
Train	matching_status	_CAP_	Train: Percent Capture Response	9.543636	9.529147	9.534547
Train	matching_status	_DFT_	Train: Total Degrees of Freedom	10498	10498	10498
Train	matching_status	_DIV_	Train: Divisor for ASE	20996	20996	20996
Train	matching_status	_GAIN_	Train: Gain	90.87273	90.59902	90.70147
Train	matching_status	_GINI_	Train: Gini Coefficient	0.9998	0.996914	0.998013
Train	matching_status	_KS_Bin_	Train: Bin-Based Two-Way Kolmogorov-Smirnov Statistic	0.954173	0.951164	0.952286
Train	matching_status	_LIFT_	Train: Lift	1.908727	1.905829	1.906909
Train	matching_status	_MAX_	Train: Maximum Absolute Error	0.999818	0.998555	0.99918
Train	matching_status	_MISC_	Train: Misclassification Rate	9.53E-05	0.00181	0.001524
Train	matching_status	_NOBS_	Train: Sum of Frequencies	10498	10498	10498
Train	matching_status	_PROF_	Train: Total Profit for matching_status	10497	10479	
Train	matching_status	_RASE_	Train: Root Average Squared Error	0.009759	0.04245	0.038624
Train	matching_status	_RESP_	Train: Percent Response	99.98182	99.83003	99.8866
Train	matching_status	_SSE_	Train: Sum of Squared Errors	1.999636	37.83452	31.32233
Train	matching_status	_SUMW_	Train: Sum of Case Weights Times Freq	20996	20996	20996
Valid	matching_status	VKS	Valid: Kolmogorov-Smirnov Statistic	1	0.995778	0.997445
Valid	matching_status	_VAPROF_	Valid: Average Profit for matching_status	1	0.997905	
Valid	matching_status	_VASE_	Valid: Average Squared Error	1.73E-08	0.002089	0.001247
Valid	matching_status	_VAUR_	Valid: Roc Index	1	0.997548	0.999178
Valid	matching_status	_VCAP_	Valid: Percent Capture Response	9.544546	9.535765	9.540153
Valid	matching_status	_VDIV_	Valid: Divisor for VASE	17184	17184	17184
Valid	matching_status	_VGAIN_	Valid: Gain	90.89091	89.8356	90.80705
Valid	matching_status	_VGINI_	Valid: Gini Coefficient	1	0.995097	0.998356
Valid	matching_status	_VKS_BIN_	Valid: Bin-Based Two-Way Kolmogorov-Smirnov Statistic	0.954455	0.950763	0.95354
Valid	matching_status	_VLIFT_	Valid: Lift	1.908909	1.907153	1.908031
Valid	matching_status	_VMAX_	Valid: Maximum Absolute Error	0.000182	1	1
Valid	matching_status	_VMISC_	Valid: Misclassification Rate	0	0.002095	0.00128
Valid	matching_status	_VNOBS_	Valid: Sum of Frequencies	8592	8592	8592
Valid	matching_status	_VPROF_	Valid: Total Profit for matching_status	8592	8574	
Valid	matching_status	_VRASE_	Valid: Root Average Squared Error	0.000132	0.045709	0.035311
Valid	matching_status	_VRESP_	Valid: Percent Response	100	99.908	99.95398
Valid	matching_status	_VSSE_	Valid: Sum of Squared Errors	0.000298	35.90228	21.42604
Valid	matching_status	_VSUMW_	Valid: Sum of Case Weights Times Freq	17184	17184	17184

Table 36 The Three Model Statistic Comparisons for the First Data Set
*In this case, tree3 (the Levenshtein Edit Distance) was selected

DataRole	Target	STAT	LABEL	Tree2(J)*	Tree(JW)	Tree3(L)
Train	matching_status	KS	Train: Kolmogorov-Smirnov Statistic	0.991053	0.986998	0.990398
Train	matching_status	_APROF_	Train: Average Profit for matching_status	0.995428	0.993427	0.995428
Train	matching_status	_ASE_	Train: Average Squared Error	0.004084	0.006026	0.004533
Train	matching_status	_AUR_	Train: Roc Index	0.997986	0.997427	0.995199
Train	matching_status	_CAP_	Train: Percent Capture Response	9.539391	9.531532	9.462773
Train	matching_status	_DFT_	Train: Total Degrees of Freedom	10498	10498	10498
Train	matching_status	_DIV_	Train: Divisor for ASE	20996	20996	20996
Train	matching_status	_GAIN_	Train: Gain	90.79295	90.63987	89.25545
Train	matching_status	_GINI_	Train: Gini Coefficient	0.995973	0.994854	0.990398
Train	matching_status	_KS_Bin_	Train: Bin-Based Two-Way Kolmogorov-Smirnov Statistic	0.952579	0.94978	0.945009
Train	matching_status	_LIFT_	Train: Lift	1.907878	1.906306	1.892555
Train	matching_status	_MAX_	Train: Maximum Absolute Error	0.999373	0.99855	0.991347
Train	matching_status	_MISC_	Train: Misclassification Rate	0.004572	0.006573	0.004572
Train	matching_status	_NOBS_	Train: Sum of Frequencies	10498	10498	10498
Train	matching_status	_PROF_	Train: Total Profit for matching_status	10450	10429	10450
Train	matching_status	_RASE_	Train: Root Average Squared Error	0.063908	0.077625	0.067326
Train	matching_status	_RESP_	Train: Percent Response	99.93734	99.85501	99.13467
Train	matching_status	_SSE_	Train: Sum of Squared Errors	85.75335	126.5144	95.16928
Train	matching_status	_SUMW_	Train: Sum of Case Weights Times Freq	20996	20996	20996
Valid	matching_status	VKS	Valid: Kolmogorov-Smirnov Statistic	0.98829	0.987379	0.986556
Valid	matching_status	_VAPROF_	Valid: Average Profit for matching_status	0.994181	0.993715	0.993599
Valid	matching_status	_VASE_	Valid: Average Squared Error	0.005396	0.006075	0.00633
Valid	matching_status	_VAUR_	Valid: Roc Index	0.998302	0.997827	0.993278
Valid	matching_status	_VCAP_	Valid: Percent Capture Response	9.539665	9.539717	9.429324
Valid	matching_status	_VDIV_	Valid: Divisor for VASE	17184	17184	17184
Valid	matching_status	_VGAIN_	Valid: Gain	90.79739	90.13141	88.58648
Valid	matching_status	_VGINI_	Valid: Gini Coefficient	0.996604	0.995653	0.986556
Valid	matching_status	_VKS_BIN_	Valid: Bin-Based Two-Way Kolmogorov-Smirnov Statistic	0.952614	0.949777	0.945099
Valid	matching_status	_VLIFT_	Valid: Lift	1.907933	1.907943	1.885865
Valid	matching_status	_VMAX_	Valid: Maximum Absolute Error	0.999373	1	0.991347
Valid	matching_status	_VMISC_	Valid: Misclassification Rate	0.005819	0.006285	0.006401
Valid	matching_status	_VNOBS_	Valid: Sum of Frequencies	8592	8592	8592
Valid	matching_status	_VPROF_	Valid: Total Profit for matching_status	8542	8538	8537
Valid	matching_status	_VRASE_	Valid: Root Average Squared Error	0.073456	0.077944	0.079563
Valid	matching_status	_VRESP_	Valid: Percent Response	99.94886	99.94941	98.7928
Valid	matching_status	_VSSE_	Valid: Sum of Squared Errors	92.72032	104.3973	108.7786
Valid	matching_status	_VSUMW_	Valid: Sum of Case Weights Times Freq	17184	17184	17184

Table 37 The Three Model Statistic Comparisons for the modified Data Set
*In this case, tree2 (The Jaro String Comparison Function) was selected

6.0 CONTRIBUTION

The record linkage problem is one of the classic problems faced when integrating information from multiple information resources. Efforts to consolidate patient data coming from heterogeneous databases with different identification schemas have a long history. Automated methods for linking patient records have been described as early as the 1970's [Sachs et al. 2000]. Though, there have been numerous efforts on resolving this problem in the record linkage problem, recently, there has been an increasing emphasis on resolving this problem in health care information systems. The lack of efficient tools for patient records matching still exists in health care providers. This was the primary motivation of this research. This research effort provides an approach to resolving this problem using a hybridization of several technologies to address the patient record matching issue in multiple databases. Methodologies and techniques from other fields, such as information retrieval, text correction, and data mining, are integrated in this approach to address the patient records matching problem. This problem is more relevant today with increasing medical costs and veterans returning from wars desiring adequate medical benefits. The framework presented could be utilized to address several issues in records matching such as:

- Define the quality of a match in third party payer databases by using quality metrics, such as Precision, Recall, F-measure etc, which are commonly used in Information Retrieval.
- Support the user in efficiently making correct matches
- Data cleaning in local databases and data warehouse.

Also, via this research

- A machine leaning approach for the patient record matching problem is introduced and decision tree induction models are applied.
- Extensive experiments have been conducted to access performances of the decision models developed for records matching and results have been very good.

The framework and methodology proposed can be used for similar problems in different domains and other applications such as multi-source integration, information retrieval.

7.0 CONCLUSION

A number of major issues, algorithms, and methodologies in record matching were reviewed. Significant research efforts were expended to develop efficient solution methods for the record matching problem. A major application focus was the patient record matching in third party payer databases. The studies of the measure of string differentiation that accounts the number of deletions, insertions, substitutions, and transpositions to transform one string into the other for string comparison was a very important step for this research. The literature offers numerous solution methods for quantifying the differences between two strings. However, selecting the best for patient record matching problem in the context of an integrated decision tree models has not been done. Even though the Levenshtein String Distance Statistic has been applied for measuring the percentage of errors in evaluations of text entry methods, this measure does not appear to reflect a human's process for defining the quality of a match, especially for last name, and first name. Extensive experiments were conducted to compare string similarity functions for pairs of last and first names in two categories, 1) Matching strings, first name and last name that are known to be the same but misspelled; 2) Non-Matching strings, first name and last name that are known to be different. For the misspelled name, human-generated errors were introduced based on various errors, such as scanning errors, phonetic and

heterographic errors, typographic errors and random errors. Overall, no string similarity function outperforms over the others in all circumstances. All three comparison vectors generated by these three string comparison functions were utilized to construct decision tree models for records linkage. The performance of resulting decision models were evaluated through extensive experiments and found to perform very well. The matching quality metrics, such as precision, recall, F-measure, and accuracy, are over 99%, ROC index are over 99.50% and mismatching rates are less than 0.18% for each model generated based on different data sets. It is therefore recommended that all three string comparison functions be used in the framework defined to construct the decision models for records matching. The quantitative matching results of this research, however, are not compared to other approaches for records matching problem. Since a standard data set is needed to perform these comparisons. In practice, there is no such a standard data set available. There is a paucity of literature describing the actual performance of such comparators in patient record linkage [Grannis et al 2004].

As an alternative to the decision tree approach, a "fuzzy logic" approach was also presented. Here the rules of matching records must be obtained from "expert" users. Fuzzy logic can be used to translate similarity measures to linguistic concept that can be utilized with expert rules and an inference system to define the quality of match. The ranking system is automated and it essentially provides a prioritized order for the user making the matching decisions with any additional information. It would improve the efficiency and quality of matching process.

8.0 FUTURE WORK

Using simple weights to assess a value for the "quality of match" of each attribute may yield efficiencies in processing and could be an alternative to ranking via fuzzy set theory. When matching candidates have been generated, they output the entire candidate matches along with each of their corresponding set of attribute similarity scores can be generated. The total object similarity score might be calculated as a weighted sum of the attributes similarity scores.

Each attribute also could be assigned a uniqueness weight that is a heuristic measure of the importance of that attribute. This is to reflect the idea that it is more likely for the records match to be correct if there is a match between unique attributes values they are rare. The uniqueness weight of an attribute is measured by the total number of unique attribute values contained in the attribute set divided by the total number of values for that attribute set.

Some type of unsupervised learning algorithms, such as clustering learning, might be applied with decision tree learning for matching rules generation. The approach, proposed and evaluated in this research, only a decision tree classifier is applied since it is assumed at a training data set is available for supervised learning. With available

training data sets the matching status is known. In practice, however, sometimes there may not be a training dataset available. In other words, there is no matching status element in the comparison vectors, even though it can be sure that the comparison

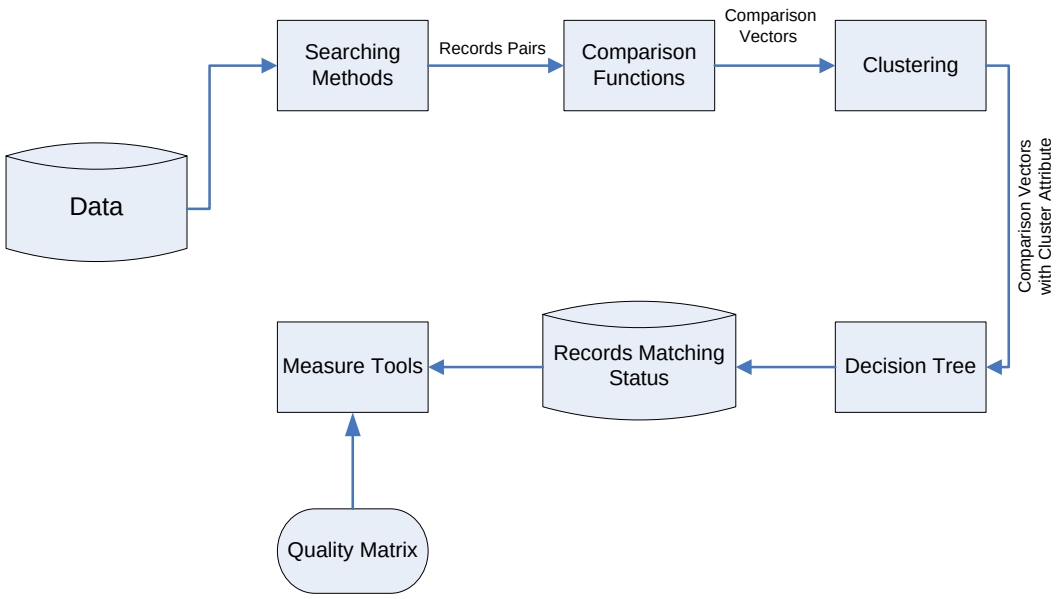


Figure 30 Machine Learning Approach Process Diagram

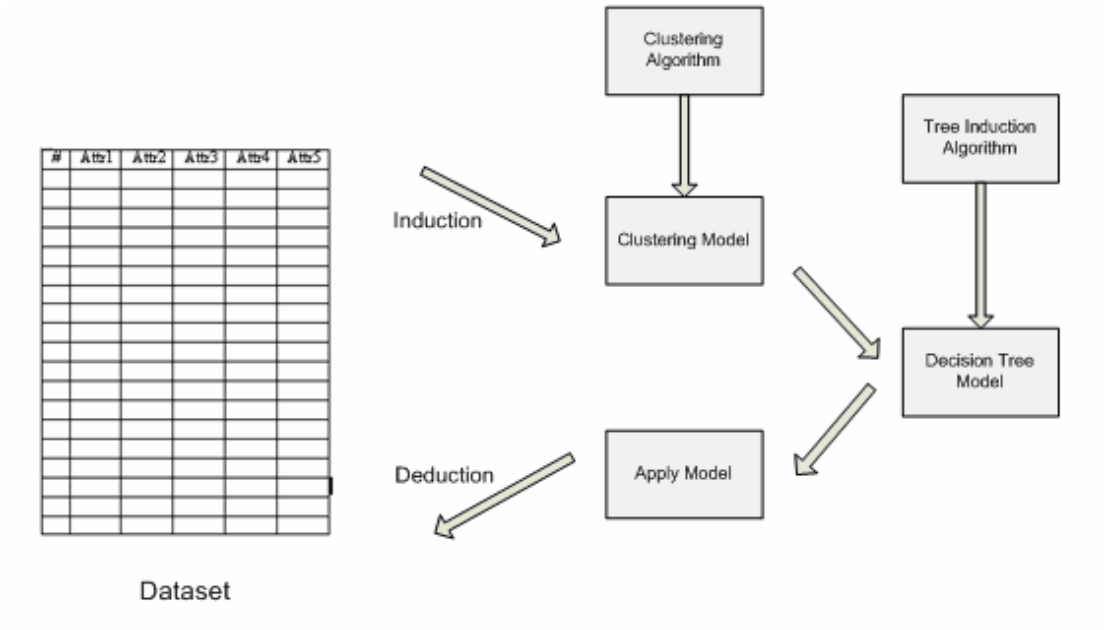


Figure 31 Clustering and Decision Tree Learning

vectors have three possible clusters, matched, not-matched, and possibly matched. So, an unsupervised learning algorithm, such as clustering learning, might be applied to identify clusters of comparison vectors so that these clusters can be mapped to the appropriate matching status. These cluster values can be used in comparison vectors, which can be an input to a decision tree inducer to generate a decision model that can predict the matching status of an unseen dataset. The process of including an unsupervised learning algorithm in the framework is illustrated in Figures 30 and 31.

Other induction learning algorithm, such as, artificial neural network based models could also be considered in the future research.

There are some limitations in this research. The records matching process includes two major steps, searching and matching. A framework and methodology for records matching has been demonstrated in this research, different string comparators have been applied for measuring the similarity of elements in record pairs, matching results and quality matrix have been compared using a decision tree model. However, searching performances and computation costs have not been compared in this research. This technique, however, becomes problematic if a value in blocking variable is recorded wrongly, and the corresponding record is inserted into a different block. So different blocking key or combinations must be tried to compare the matching quality matrix and computation costs.

In this research study, the HIC number, last name, first name, date of birth, and gender are used as the attributes for matching records. Increasing the set of input attributes such as address, place of birth might improve the quality of matching. However, increasing the set of attributes might make the model more complex.

The other limitation is that extensive experimental evaluation to validate the proposed models and their matching quality were made using synthetic datasets with generated errors because access to real patient records and third party payer databases was difficult.

REFERENCES

Adelhard, K., Eckel, R., Holzel, D., and Tretter, W., (1995), A prototype of a computerized patient record, *Computer Methods and Programs in Biomedicine* 48 (1995) 115 -119

Allen, V., Arocha, J., and Patel, V., (1998), Evaluating evidence against diagnostic hypotheses in clinical decision making by students, residents and physicians, *International Journal of Medical Informatics* 51 (1998) 91-105

Amir, A., Lewenstein, M., and Porat, E., (2004), Faster algorithms for string matching with *k* mismatches, *Journal of Algorithms* 50 (2004) 257- 275

Ball, M. J., (2003), Hospital information systems: perspectives on problems and prospects, 1979 and 2002, *International Journal of Medical Informatics* 69 (2003) 83-89

Bell, G. B., and Sethi A., (2001), Examining the benefits and pitfalls of a distributed collection of medical records Matching records in a National Medical Patient Index *Communication of the ACM* September 2001/Vol.44, No. 9

Bell, J. Finding the missing pieces for payment, (2006), *Healthcare Financial Management* March 2006

Bezdek, J.C. 1993 Editorial: Fuzzy Models – What are they, and why? *IEEE Transactions on Fuzzy Systems,* Vol. 1, No.1, February 1993 – Edited by P.D.

Bortolan, G., Bressan, M., and Golferini, F., (2000), Gender and Age Influences in QT-Dispersion, *Computers in Cardiology* 2000; 27:359-362

Budzinsky, C. D. (1991), "Automated Spelling Correction," Statistics Canada.
Fellegi, I. P., and Sunter, A. B. (1969), "A Theory for Record Linkage," *Journal of the American Statistical Association*, 64, 1183-1210

Choi, Y. B., Krause, J. S., Seo, H., and Capitan, E., (2006), Telemedicine in the USA: Standardization through information management and technical applications, *IEEE Communication Magazine* April 2006

Buche, P., Dervin, C, Haemmerle, O., Thomopoulos, R., (2005), Fuzzy querying of incomplete, imprecise, and heterogeneously structured data in the relational model using ontologies and rules, *IEEE Transaction on Fuzzy Systems*, Vol. 13, No. 3, June 2005

Chen, Y, Che, D, Aberer, K., (2002), On the efficient evaluation of relaxed queries in biological database, (2002), *CIKM'02, November 4-9, 2002*, McLean, Virginia, USA

Christen, Peter and Churches, Tim A Probabilistic Deduplication, Record Linkage and Geocoding System, http://datamining.anu.edu.au/linkage.html (Accessed on November 10, 2007)

Ciccarese, P., Caffi, E., Quaglini, S. and Stefanelli, M., (2005), Architectures and tools for innovative health information systems: The Guide Project, *International Journal of Medical Informatics* (2005) 74, 553-562

Cohen, W.W., and Ravikumar, P. (2003) Secondstring: An open-source java toolkit of approximate string-matching techniques. Project page, http://secondstring.sourceforge.net (Accessed on June. 21, 2007)

Cohen, W. W. Ravikumar, P. and Fienberg, S.E. (2003) A Comparison of String Distance Metrics for Name-Matching Tasks *American Associate for Artificial Intelligence (www.aaai.org)*

Dhillon, H. and Forducey, P.G., (2006), Implementation and Evaluation of Information Technology in Telemedicine, *Proceedings of the 39the Hawaii International Conference on system Science* 2006

Dwivedi, A. Bali, R.K., James, A.E., Naguib, R.N.G., and Johnston, D., (2002), *Proceedings of the 2002 IEEE Canadian Conference on Electrical & Computer Engineering* 0-7803-751 4-9/02/$17.00 0 2002 IEEE

Elmagarmid, A. K., Horowitz, B., Karabatis, G., Umar, A., (1996), Issue in Multisystem Integration for Achieving Data Reconciliation and Aspects of Solution, *Technical report*, Bellcore Research, 1996

Fawcett, T. (2004) ROC graphs: Notes and practical considerations for researchers. *Machine Learning, 31*

Fawcett, T., & Flach, P.A. (2005) A response to Webb and Ting's on the application of ROC analysis to predict classification performance under varying class distributions. *Machine Learning, 58(1), 33-38*

Flach, P., (2004) Tutorial at ICML 2004: The Many Faces of ROC Analysis in Machine Learning. Unpublished Manuscript

Flach, P., Blockeel, H., Ferri, C., Hernandez-Orallo, J., & Struyf, J. (2003) Decision support for data mining: Introduction to ROC analysis and its applications. *Data mining and decision support: Aspects of integration and collaboration (pp.81-90)*

Febrl-0.4: http://datamining.anu.edu.au/linkage.html (Accessed on Nov.10, 2007)

Fellegi, L. and Sunter, A., A theory for Record Linkage. *Journal of the American Statistical Society,* 64:1183-1210, 1969

Flores-Fonseca, M. and García-Valdez, J. Design and Implementation of an Inference Engine for Fuzzy Systems,

Field M. J., (1996), Telemedicine, a Guide to Accessing Telecommunication in Health Care. *Washington DC: National Academy Press*. 1996

García-Valdez, M., and Flores-Fonseca, J., Design and Implementation of an Inference Engine for Fuzzy Systems

Georgakopoulos, D., Karabatis, G., Gantimahapatruni, S., (1997), Specification and management of interdependent data in operational systems and data warehouse *Distributed and Parallel Database* 5 (2) (1997) 121-166

Gill, L. E., (1997), OX-LINK: The Oxford Medical Record Linkage System *Record Linkage Techniques* 1997

Ginneken, A. M., (2002) the Computerized Patient Record: Balancing Effort and Benefit, *International Journal of Medical Informatics* 65 (2002) 97 – 119

Grannis et al (2004) Real World Performance of Approximate String Comparators for use in Patient Matching: *MEDINFO 2004 Amsterdam: IOS Press @ 2004 IMIA*

Gravano, L., Ipeirotis, P. G., Jagadish, H. V., Koudas, N., Muthukrishnan S., Srivastava, D., Approximate string joins in a database (almost) for free *Proceedings of the 27th International Conference on Very Large Database*, pp: 491-500

Gu, L., Baxter, R., Vickers, D., and Rainsford, C., (2004) Record linkage: Current Practice and Future Directions, *CSIRO Mathematics and Information Science*

Hall, L. O., Chawla, N., and Bowyer, (1998), K. W., Decision tree learning on very large data sets, *0-7803-4778-1/98 1998 IEEE*

Hamel, L., Model Assessment with ROC Curves http://homepage.cs.uri.edu/faculty/hamel/pubs/hamel-roc.pdf (Accessed on Jan. 28, 2008)

Hernandez, M. A., Stolfo, S. J.,(1995), The Merge/Purge Problem for Large Database *Proceedings of ACM SIGMOD/PODS,* 1995, San Jose, CA.

Hudson, D. L. and Cohen, M. E., (2000), Using the Internet to Assist Clinical Decision Making, *0-7803-6449-x/00 @2000 IEEE*

Ingenerf, J., (1999), Telemedicine and terminology: Different needs of context information, *IEEE Transactions and Information Technology in Biomedicine*, Vol. 3, No. 2, June 1999

Jaro, M. A, (1985), Advances in Record Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. *Journal of the American Statistical Society, 84(406): 414-420, 1985*

Jaro, M. A. (1989) "Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida," *Journal of the American Statistical Association*, 89, 414-420

Jaro, M. A. (1995) "Probabilistic Linkage of Large Public Health Data File" *Statistics in Medicine* 14:491-498

Jin, L., Li, C., and Mehrotra, S., (2003) Efficient Record linkage in Large Data Sets *0-7695-189/03 @20003 IEEE*

Jin, L. Li, C. and Mehrotra, S., (2003) Efficient Record Linkage in Large Data Sets. I*nternational Conference on Database Systems for Advanced Applications (DASFAS)*, Tokyo, Japan, March 2003

Johnson K., Svirbely J., Sriram MG, Smith J., Kantor G., and Rodriguez J., Automated Medical Algorithms: Issues for Medical Errors.

Jones, N. B., Wang, J. T., Sehmi, A. S., and Bono, D. P., (1995) Knowledge-Based Systems and Neural Networks for Clinical Decision Making, *Control Eng. Practice* Vol. 3. No. 7 pp 967-975 1995

Kaihara, S., (1998), Realization of the Computerized Patient Record; Relevance and Unsolved Problems *International Journal of Medical Informatics* 49 (1998) 1-8

Kantradzic, M., Data Mining Concepts, Models, Methods, and Algorithms, a John Willey & Sons, Inc., Publication, 2003

Koncar, M. and Gvozdanovic, D., (2006), Primary healthcare information system -- The Cornerstone for the next generation healthcare sector in Republic of Croatia *International Journal of Medical Informatics*, Volume 75, Issues 3-4, March-April 2006, Pages 306-314

Kraft, D. H., Che, J., and Mikulcic, A. (2000), Combining fuzzy clustering and fuzzy inferencing in information retrieval, *0-7803-5877-5/00/ 2000 IEEE*

Kudoh, Y., Haraguchi, M., and Okubo, Y. (2003), Data abstractions for decision tree induction, *Theoretical Computer Science,* 292 (2003) 378-416

Kukich, K (1992a), Techniques for automatically correcting words in text, *ACM Computing Surveys*, Vol. 24(4)

Kukich, K (1992b), Spelling Correction for the Tele-communications Network for the Deaf, *Communications of the ACM*, Vol. 35(5)

Kyedar J.C., (2003), Special Issue: Success stories in Telemedicine: Some empirical evidence, *Telemedicine J*., vol9, no. 1 2003

Levenshtein VI., Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady* 1966; 10(8):707-710.

Lloyd-Thomas, H., Wright, J. H., and Jones, G. J. F., (1995), an Integrated Grammer/Bigram Language Model Using Path Scores, *0-7803-2431-5/95 1995 IEEE*

Lorence, D. and Churchill, R., (2005), Clinical Knowledge Management Using Computerized Patient Record Systems: Is the Current Infrastructure Adequate? *IEEE Transactions on Information Technology in Biomedicine*, Vol. 9, No2, June 2005

Lu, Z. et al., (2004), Predicting Subcellular Localization of Proteins Using Machine-Learned Classifiers, Bioinformatics, Volume 20, Issue 4, March 2004, pp. 547 - 556.

MacKenzie, I. S., and Soukoreff, R. W, (2002), a Character-lever Error Analysis Technique for Evaluating Text Entry Methods, *NordiCHI,* October 19-23, 2002

Martinez, A., Vilarroel, V., Seoane, J., and Pozo, F.D., (2004), Rural Telemedicine for Primary Healthcare in Developing Countries, *IEEE Technology and Society Magazine* summer 2004

Manzoul A. M. and Rao, B. V. (1988) Multi-input Fuzzy Inference Engine on a Systolic Array, ACM 1988 0-89791-271-3/88/0006/0958

Mohamed, A. (1992), Application of Artificial Intelligence for Clinical Decision Making and Reasoning, *0-7803-0720-8/92 @ 1992 IEEE*

Mukaidono, M. (2001) Fuzzy Logic for Beginners, Singapore: World Scientific, p61

Navarro, G., (2001), A Guided Tour to Approximate String Matching, *ACM Computing Surveys*, Vol. 33, No.1 March 2001, pp. 31-88

Navarro, G., Baeza-Yates, R., Arcoverde J., (2003), Matchsimile: A flexible Approximate Matching Tools for Searching Proper Names, *Journal of the American Society for Information Science and Technology*, vol. 54(1) 2003

Newcombe, H. B., Kennedy, J. M., Axford, S. J., and James, A. P., Automatic linkage of vital records, *Science* 130(3381): 954-959

Pfeifer, U., Peersch, T., and Fuhr, N., (1996) Retrieval Effective of Proper Name Search Methods, *Information Processing and Management* 32, 6 (1996), 667-669.

Pose, M. Czaja, S. J., The Usability of Information Technology within Emergency Care Setting, *19th International Conference on Computers and Industrial Engineering* Vol. 31, No. ½, pp.455-458

Poster E. H. and Winkler W. E., (1997), Approximate string comparison and its effect on an advanced record linkage system, In proceedings of an International Workshop and Exposition-*Record Linkage Techniques*, Arlington, VA, USA, 1997

Provost, F. and Fawcett, T. Robust Classification for Imprecise Environments. *Machine Learning* 42(3): 203-231, 2001.

Quantin, C., Bouzelat, H., Allaert, F.A.A., Benhamiche, A. M., Faivre, J., and Dusserre, L., (1998), How to Ensure Data Security of an Epidemiological Follow-up: Quality Assessment of an Anonymous Record Linkage Procedure. *International Journal of Medical Informatics* 49 (1998) 117-122

Quantin, C., Binquet, C.; Bourquard, K.; Pattisina, R.; Gouyon-Cornet, B.; Ferdynus, C.; Gouyon, J.-B.; Allaert, F.A., (2004), Which are the best identifiers for record linkage? *Medical Informatics and the Internet in Medicine*, v 29, n 3-4, Sept.-Dec. 2004, 221-7

RNGP, Random Number Generator Pro is a Windows based application designed to generate random numbers. http://www.segobit.com/rng.htm (Accessed and downloaded on Nov. 10, 2007)

Sachs, P., Gall, W., Marksteiner, A., and Dorda, W., (2000), Unambiguous identification of hospital patients Case study at the university departments of the General Hospital, Vienna, *International Journal of Medical Informatics* 57(200) 165-179

Saini, S., A (2004) Fuzzy-Based Approach for the Prediction of Quality Attributes, *Transactions on engineering, computing and technology*, vol. 3, DECEMBER 2004 ISSN 1305-5313

Sakamoto, and Norihiro (1998), Availability of software services for a hospital information system, *International Journal of Medical Informatics*, 1998, 49(1): 89 – 96

Schmidtke, D. W., Pishko, M. V., Quinn, C. P., and Heller, A, (1996), Statistics for critical clinical decision making based on readings of pairs of implanted sensors, *Anal. Chem.* 1996, 68, 2845-2849

Schneider, M., Bunke, H. Kandel, A., (2001), Using fuzzy logic to match strings in *documents, International Journal of Intelligent Systems*, Vol. 16, 609-619 (2001)

Shine, K., (1996), Impact of Information Technology on Medicine Technology *in Society*, Vol. 18, No2, pp. 117 -126, 1996

Siler, M. and Buckley, J., (2005) Fuzzy Expert Systems and Fuzzy Reasoning, *John Wiley & Sons, Inc.,* Hoboken, New Jersey, 2005.

Smith, M.E. and Newcombe, H.B. (1975), Methods for computer linkage of hospital admissions-separation records into cumulative health histories, *Methods Information Medical,* 14 (1975) 118-125

Soukoreff, R.W. and MacKenzie, I.S. (2001), Measuring errors in text entry teaks: An application of the Levenshtein string distance statistic. *Companion Proceedings of the ACM Conference on Human Factors in Computer Systems* – CHI2001, New York: ACM, pp319-320

Stamford, P., Bickford, T. Hsiao, H., and Mattern, W., (1999), the Significance of Telemedicine in a Rural Emergency Department, *IEEE Engineering in Medicine and Biology* 1999

Takizawa, M, Sone, S., Hanamura, K, and Asakura, K., (2001), Telemedicine system using computed tomography van of high-speed telecommunication vehicle *IEEE Transactions on Information Technology in Biomedicine,* Vol. 5, No. 1, March 2001

Tishelman, C., Lundgren, E., Skald A., Tornberg, S., and Larsson, B. W., (2002), Quality of care from a patient perspective in population-based cervical cancer screening *Taylor & Francis 2002* ISSN 0284-186X

Turgey, J. P. and Connelly, D. P., (1996), the Relationship between Nursing and Medical Cultures: Implication for the design and implementation of a clinician's workshop *Proceedings of the 17th Annual Symposium on Computer Application in Medical Care*, 1996, 233-237

Ukkonen, E., (1992), Approximate string-matching with q-grams and maximal matches, *Journal of Computer Science* 1992, pp.191-211

Verykios, V. S., Elmagarmid, A. K., and Houstis, E. N., (2000), Automating the approximate record-matching process, *Information Science* 126 (2000) 83-98

Verykios, V. S., Moustakides, G. V., and Elfeky, M. G., (2003), A Bayesian decision model for cost optimal record matching, *The VLDB Journal* (2003) 12:28-40 / Digital Object Identifier (DOI)

Ville, de B. (2006), Decision Trees for Business Intelligence and Data Mining using SAS Enterprise Miner, *SAS Press Series*, SAS Institute Inc., Cary, NC, USA

wiki1 wikipedia.org http://en.wikipedia.org/wiki/Typo (Accessed on Nov. 24, 2007)

wiki2 wilipedia.org http://en.wikipedia.org/wiki/Homonym (Accessed on Nov. 24, 2007)

Wei, M., Sung, A. H., Cather, M. E., (2006), Improving database quality through eliminating duplicate records, *Data Science Journal*, Vol. 5, 19 October, 2006

Weiner, M., Stump, T.E., Callahan, C. M., Levis, J. N., and McDonald, C. J., A Practical Method of Linking Data from Medicare Claims and a Comprehensive Electronic Medical Records System, *International Journal of Medical Informatics* (2003) 71, 57-69

Winkler, W. E., (1985), Preprocessing of Lists and String Comparison *Record Linkage Techniques* 1985, edited by W. Alvey and B. kilss, U. S. Internal Revenue Service, Publication 1299 (2-86), 181-187

Winkler, W. E., and Thibaudeau, Y., (1990), An application of the Fellegi-Sunter Model of Record Linkage to the 1990 U.S. Decennial Census (1990)

Winkler, W. E. (1990), String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage (1990)

Winkler, W. E., (1995), Matching and Record Linkage, *Business Survey Methods* New York: John Wiley and Sons, Inc., 1995, 335-384

Wright, D. D., (1997), Telemedicine and developing countries: A report of Study Group 2 of the ITU Development Sector, *J. Telemedicine and Telecare,* London, vol. 4, Suppl. 2, pp.1-85, 1997

Wootten, R., (1997), the possible use of telemedicine in developing countries *J. Telemedicine and Telecare,* vol. 3 pp 23-26, 1997

Zach, S., (1996), Telemedicine overview and summary, *0-7803-3330-6 1996-IEEE*

Jaro, M. A. 1989 "Advances in record linking methodology as applied to the 1985 census of Tampa Florida". *Journal of the American Statistical Society* 64:1183-1210

Zadeh, L.A. (1965) Fuzzy Sets, *Information and Control*, Vol. 8, pp. 328-352, 1965

APPENDICES

Appendix 1: Application Code 1: MainWindow

```
package dissertation.ie;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import org.eclipse.jface.action.MenuManager;
import org.eclipse.jface.action.StatusLineManager;
import org.eclipse.jface.action.ToolBarManager;
import org.eclipse.jface.viewers.IStructuredContentProvider;
import org.eclipse.jface.viewers.ITableLabelProvider;
import org.eclipse.jface.viewers.LabelProvider;
import org.eclipse.jface.viewers.TableViewer;
import org.eclipse.jface.viewers.Viewer;
import org.eclipse.jface.window.ApplicationWindow;
import org.eclipse.swt.SWT;
import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.graphics.Image;
import org.eclipse.swt.graphics.Point;
import org.eclipse.swt.layout.GridData;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.widgets.Button;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Control;
import org.eclipse.swt.widgets.Display;
import org.eclipse.swt.widgets.FileDialog;
import org.eclipse.swt.widgets.Group;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.swt.widgets.Table;
import org.eclipse.swt.widgets.TableColumn;
import org.eclipse.swt.widgets.Text;

import com.wcohen.ss.*;
```

```
/**
 * @author DW
 *
 */
public class MainWindow extends ApplicationWindow {
	private TableViewer tableViewer;
	private Group inputTestGroup;
	private Table table;
	private Text inputText;
	private List<MyTable> sourceData = new ArrayList();
//	private  MyTableData myTableDatas = new MyTableData();

	/** my domain data * */
	private MyTableData myTableData = new MyTableData();

	/** the file I want to compare * */
	private String file;

	class ContentProvider implements IStructuredContentProvider {
		public Object[] getElements(Object inputElement) {
			MyTableData tableData = (MyTableData) inputElement;
			return tableData.getRows().toArray();
		}

		public void dispose() {
		}

		public void inputChanged(Viewer viewer, Object oldInput,
			Object newInput) {
		}
	}

	class TableLabelProvider extends LabelProvider implements
			ITableLabelProvider {
		public String getColumnText(Object element, int
columnIndex)
			{
			String result = "";
			MyTable row = (MyTable) element;
			switch (columnIndex) {
			case 0:
				result = row.getFirstName();
				break;
			case 1:
				result = row.getLastName();
				break;
			case 2:
				result = row.getGender();
				break;
			case 3:
				result = row.getHic();
				break;
			case 4:
				result = row.getDob();
				break;
			case 5:
```

```
                        result = new
Double(row.getScore()).toString();
                        break;
                  /** for other columns, goes on here * */
                  default:
                        break;
                  }
                  return result;
            }

            public Image getColumnImage(Object element, int
columnIndex)
                  {
                  return null;
            }
      }

/*
 * Create the application window
 */
public MainWindow() {
      super(null);
      createActions();
      addToolBar(SWT.FLAT | SWT.WRAP);
      addMenuBar();
      addStatusLine();
}

/**
 * Create contents of the application window
 *
 * @param parent
 */
@Override
protected Control createContents(Composite parent) {
      Composite container = new Composite(parent, SWT.NONE);
      final GridLayout gridLayout = new GridLayout();
      gridLayout.makeColumnsEqualWidth = true;
      container.setLayout(gridLayout);

      inputTestGroup = new Group(container, SWT.NONE);
      inputTestGroup.setText("Input Contested Record (First Name,
            Last Name, Gender, HIC, DOB(MM/DD/YYYY))");
      final GridLayout gridLayout_1 = new GridLayout();
      gridLayout_1.numColumns = 5;
      inputTestGroup.setLayout(gridLayout_1);
      final GridData gd_inputTestGroup = new GridData(SWT.FILL,
            SWT.FILL,true, false);
      inputTestGroup.setLayoutData(gd_inputTestGroup);

      final Label inputLabel = new Label(inputTestGroup,
            SWT.NONE);
      inputLabel.setLayoutData(new GridData());
      inputLabel.setText("Input");

      inputText = new Text(inputTestGroup, SWT.BORDER);
      final GridData gd_inputText = new GridData(SWT.FILL,
```

```
      SWT.CENTER, true,false, 4, 1);
inputText.setLayoutData(gd_inputText);

final Button method1Button = new Button(inputTestGroup,
      SWT.CHECK);
method1Button.setText("Levenstein");

method1Button.setSelection(false);

final Button method2Button = new Button(inputTestGroup,
      SWT.CHECK);
method2Button.setText("Jaro");
method2Button.setSelection(false);

final Button method3Button = new Button(inputTestGroup,
      SWT.CHECK);
method3Button.setText("JaroWinkler");
method3Button.setSelection(false);
final Button inputFileButton = new Button(inputTestGroup,
      SWT.NONE);
inputFileButton.addSelectionListener(new SelectionAdapter()
{
      public void widgetSelected(SelectionEvent e) {
      System.out.println("Print anything");
      FileDialog dlg = new FileDialog(getShell(),
                  SWT.OPEN);
      file = dlg.open();
      if (file == null) {
          System.err.println("ERROR, file not selected");
      }else {
          System.out.println("File selected");
          readFile();

      }
}

private void readFile() {
      myTableData.clean();
      if (file != null) {
          File myFile = new File(file);
          try {
               BufferedReader br = new
               BufferedReader(new FileReader(myFile));
               try {
                   String eachLine;
                   While ((eachLine=br.readLine()) != null ) {
                        System.out.println("debug,
                        print each line read:" + eachLine);

                        eachLine.trim();//trim  space
                        String[] lineItems =
                          eachLine.split(",");
                        MyTable oneItem = new MyTable();
                        oneItem.setFirstName
                          (lineItems[0].trim());
                        oneItem.setLastName
                          (lineItems[1].trim());
```

```
                                oneItem.setGender(lineItems[2].trim());
                                oneItem.setHic(lineItems[3].trim());
                                oneItem.setDob(lineItems[4].trim());
                                //......


                                myTableData.addMyTableValue(oneItem);



                            }
                        } catch (IOException e1) {
                            // TODO Auto-generated catch block
                            e1.printStackTrace();
                        }

                    } catch (FileNotFoundException e1) {
                        // TODO Auto-generated catch block
                        e1.printStackTrace();
                    } catch (IOException e1) {
                        // TODO Auto-generated catch block
                        e1.printStackTrace();
                    }
                    System.out.println("transfer data to table viewer");
                    tableViewer.setInput(myTableData);
                }
            }

});
inputFileButton.setText("Source File");

final Button letsGoButton = new Button(inputTestGroup,SWT.NONE);
letsGoButton.addSelectionListener(new SelectionAdapter() {
    public void widgetSelected(SelectionEvent e) {
        System.out.println("match pressed");




       /*****************************************
        * IMPORTANT, write all my code here read file
        * line by line, call the functions to compare
        * store the object in MyTable put it into
        * MyTableData. refresh the data in the table
        */




      String inputString = inputText.getText();
      System.out.println("Input string is: " + inputString);
      String inputStringTrim = inputString.trim();
      String[] inputStringItem = inputStringTrim.split(",");
      for(int i =0; i < inputStringItem.length; ++i){
           inputStringItem[i]= inputStringItem[i].trim();
```

```
        }
        if (method1Button.getSelection()) {
            System.out.println("method 1 is checked");
            Levenstein lev = new Levenstein();
            System.out.println("A new round");
            computeScoreLev(inputStringItem, lev);
        }

        if (method2Button.getSelection()) {
            Jaro jaro = new Jaro();
            System.out.println("method 2 is checked");
            computeScoreJaro(inputStringItem, jaro);
        }

        if (method3Button.getSelection()) {
            JaroWinkler jaroWinkler = new JaroWinkler();
            System.out.println("method 3 is checked");
            computeScoreJaroWinkler(inputStringItem,jaroWinkler);
        }


}

private void computeScoreJaro(String[]inputStringItem, Jaro jaro) {
   List<MyTable> myList = myTableData.getRows();
   List<MyTable> scoreList = new ArrayList<MyTable>();
   for (MyTable data : myList) {
        //for (int i =0; i< inputStringItem.length; ++i){
             double scoreFirstName = jaro.score(data.getFirstName(),
                 inputStringItem[0]);
             double scoreLastName = jaro.score(data.getLastName(),
                 inputStringItem[1]);
             double scoreGender = jaro.score(data.getGender(),
                 inputStringItem[2]);
             double scoreHic = jaro.score(data.getHic(),
                 inputStringItem[3]);
             double scoreDob = jaro.score(data.getDob(),
                 inputStringItem[4]);
             double overallScore =(scoreFirstName+scoreDob+
                 scoreLastName + scoreGender+scoreHic)/5; //..todo
             MyTable recordAttrScore = new
             MyTable(String.valueOf((scoreFirstName)),
                  String.valueOf(scoreLastName),
                  String.valueOf(scoreGender),
                  String.valueOf(scoreHic),
                  String.valueOf(scoreDob), overallScore);
                  scoreList.add(recordAttrScore);
                  data.setScore(overallScore);
              //}

    }
   //tableViewer.setInput(myTableData);
   List<MyTable> myNewList = new ArrayList();
   for(int i =0; i < myList.size(); ++i) {
       myNewList.add(myList.get(i));
       myNewList.add(scoreList.get(i));
   }
```

```
   Collections.sort(myNewList);
   myTableData.setRows(myNewList);
   tableViewer.setInput(myTableData);
// tableViewer.setInput(myNewList);


}

private void computeScoreLev(String[] inputStringItem, Levenstein
lev)
    {
    List<MyTable> myList = myTableData.getRows();
    List<MyTable> scoreList = new ArrayList<MyTable>();
    for (MyTable data : myList) {
//       for (int i =0; i< inputStringItem.length; ++i){
           double scoreFirstName =
                simLev(lev.score(data.getFirstName(),
                inputStringItem[0].trim()),
                maxLength(data.getFirstName().length(),
                inputStringItem[0].trim().length()));
           double scoreLastName =
                simLev(lev.score(data.getLastName(),
                nputStringItem[1].trim()),
                maxLength(data.getLastName().length(),
                inputStringItem[1].trim().length()));

           System.out.println("input:"+
                inputStringItem[1].trim()+"end");
           double scoreGender =
                simLev(lev.score(data.getGender(),
                inputStringItem[2].trim()),
                maxLength(data.getGender().length(),
                inputStringItem[2].trim().length()));
           double scoreHic = simLev(lev.score(data.getHic(),
                inputStringItem[3].trim()),
                maxLength(data.getHic().length(),
                inputStringItem[3].trim().length()));
           double scoreDob = simLev(lev.score(data.getDob(),
                inputStringItem[4].trim()),
                maxLength(data.getDob().length(),
                inputStringItem[4].trim().length()));

           System.out.println(scoreFirstName +","+
                scoreLastName+","+scoreGender+",
                "+scoreHic+scoreDob);
           double overallScore =(scoreFirstName+scoreDob+
                scoreLastName+ scoreGender+scoreHic)/(double)5;
                //..todo
           MyTable recordAttrScore = new
                MyTable(String.valueOf((scoreFirstName)),
                String.valueOf(scoreLastName),
                String.valueOf(scoreGender),
                String.valueOf(scoreHic),
                String.valueOf(scoreDob), overallScore);
                scoreList.add(recordAttrScore);
                data.setScore(overallScore);
//          }
```

```
        }
        List<MyTable> myNewList = new ArrayList();
        for(int i =0; i < myList.size(); ++i) {
              myNewList.add(myList.get(i));
              myNewList.add(scoreList.get(i));
        }
        Collections.sort(myNewList);
        myTableData.setRows(myNewList);
        tableViewer.setInput(myTableData);
//      tableViewer.setInput(myNewList);
      }

      private int maxLength(int i, int j) {
         if (i > j) return i;
         else return j;
         }

      private double simLev(double score, int length){
         return 1-Math.abs(score)/(double)length;
      }

      private void computeScoreJaroWinkler(String[]
                inputStringItem, JaroWinkler jaroWinkler) {
           List<MyTable> myList = myTableData.getRows();
           List<MyTable> scoreList = new ArrayList<MyTable>();

           for (MyTable data : myList) {
                //for (int i =0; i< inputStringItem.length; ++i){
                     double scoreFirstName =
                         jaroWinkler.score(data.getFirstName(),
                         inputStringItem[0].trim());
                     double scoreLastName =
                         jaroWinkler.score(data.getLastName(),
                         inputStringItem[1].trim());
                     double scoreGender =
                         jaroWinkler.score(data.getGender(),
                         inputStringItem[2].trim());
                     double scoreHic =
                         jaroWinkler.score(data.getHic(),
                         inputStringItem[3].trim());
                     double scoreDob =
                         jaroWinkler.score(data.getDob(),
                         inputStringItem[4].trim());

                     System.out.println(scoreFirstName +","+
                         scoreLastName +","+scoreGender+",
                         "+scoreHic+scoreDob);
                     double overallScore
                         =(scoreFirstName+scoreDob+scoreLastName+
                         scoreGender+scoreHic)/5; //..todo
                     MyTable recordAttrScore = new
                          MyTable(String.valueOf((scoreFirstName)),
                          String.valueOf(scoreLastName),
                          String.valueOf(scoreGender),
                          String.valueOf(scoreHic),
                          String.valueOf(scoreDob), overallScore);
```

```
                    scoreList.add(recordAttrScore);
                    data.setScore(overallScore);
                //}

            }
            List<MyTable> myNewList = new ArrayList();
            for(int i =0; i < myList.size(); ++i) {
                myNewList.add(myList.get(i));
                myNewList.add(scoreList.get(i));
            }
            Collections.sort(myNewList);
            myTableData.setRows(myNewList);
            tableViewer.setInput(myTableData);
        }
    });

    letsGoButton.setLayoutData(new GridData(SWT.RIGHT, SWT.CENTER,
        false,false));
    letsGoButton.setText("Matching");

    final Group testResultGroup = new Group(container, SWT.NONE);
    testResultGroup.setText("Matching Result");
    final GridData gd_testResultGroup = new GridData(SWT.FILL,
        SWT.FILL, true, true);
    gd_testResultGroup.widthHint = 649;
    testResultGroup.setLayoutData(gd_testResultGroup);
    testResultGroup.setLayout(new GridLayout());

    tableViewer = new TableViewer(testResultGroup, SWT.BORDER);
    tableViewer.setContentProvider(new ContentProvider());
    tableViewer.setLabelProvider(new TableLabelProvider());
    table = tableViewer.getTable();
    table.setHeaderVisible(true);
    table.setLayoutData(new GridData(SWT.CENTER, SWT.FILL, true,
           true));

    final TableColumn newColumnTableColumn_1 = new
        TableColumn(table, SWT.NONE);
    newColumnTableColumn_1.setWidth(121);
    newColumnTableColumn_1.setText("First Name");

    final TableColumn newColumnTableColumn_2 = new
        TableColumn(table, SWT.NONE);
    newColumnTableColumn_2.setWidth(126);
    newColumnTableColumn_2.setText("Last Name");

    final TableColumn newColumnTableColumn_3 = new
        TableColumn(table, SWT.NONE);
    newColumnTableColumn_3.setWidth(128);
    newColumnTableColumn_3.setText("Gender");

    final TableColumn resultTableColumn = new TableColumn(table,
            SWT.NONE);
    resultTableColumn.setWidth(117);
    resultTableColumn.setText("HIC");

   final TableColumn newColumnTableColumn = new TableColumn(table,
```

```
            SWT.NONE);
        newColumnTableColumn.setWidth(112);
        newColumnTableColumn.setText("DOB");

        final TableColumn newColumnTableColumn_4 = new
            TableColumn(table, SWT.NONE);
        newColumnTableColumn_4.setWidth(100);
        newColumnTableColumn_4.setText("Score");


//      MyTable test1 = new MyTable();
//      test1.setDob("123");
//      test1.setFirstName("ffist");
//      test1.setGender("testG");
//      test1.setHic("hic");
//      test1.setLastName("lastname");
//      test1.setScore(0.0D);
//      myTableData.addMyTableValue(test1);
        tableViewer.setInput(myTableData);
              //
        return container;
        }

        /**
         * Create the actions
         */
        private void createActions() {
              // create the actions
        }

        /**
         * Create the menu manager
         *
         * @return the menu manager
         */
        @Override
        protected MenuManager createMenuManager() {
              MenuManager menuManager = new MenuManager("menu");
              return menuManager;
        }

        /**
         * Create the toolbar manager
         *
         * @return the toolbar manager
         */
        @Override
        protected ToolBarManager createToolBarManager(int style) {
              ToolBarManager toolBarManager = new
                  ToolBarManager(style);
              return toolBarManager;
        }

        /**
         * Create the status line manager
         *
         * @return the status line manager
```

```
	 */
	@Override
	protected StatusLineManager createStatusLineManager() {
		StatusLineManager statusLineManager = new
		   StatusLineManager();
		statusLineManager.setMessage(null, "");
		return statusLineManager;
	}

	/**
	 * Launch the application
	 *
	 * @param args
	 */
	public static void main(String args[]) {
		try {
			MainWindow window = new MainWindow();
			window.setBlockOnOpen(true);
			window.open();
			Display.getCurrent().dispose();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	/**
	 * Configure the shell
	 *
	 * @param newShell
	 */
	@Override
	protected void configureShell(Shell newShell) {
		super.configureShell(newShell);
		newShell.setText("Records Matching Comparison");
	}

	/**
	 * Return the initial size of the window
	 */
	@Override
	protected Point getInitialSize() {
		return new Point(590, 518);
	}

}
```

Appendix 2: Application Code 2 MyTable

```
**
 * @author DW
 *
 * This is my model
 */
public class MyTable implements Comparable{
   private String firstName;
   private String lastName;
   private String dob;
   private String gender;
   private String hic;
   private double score;

   public MyTable(){};

   public MyTable(String fn, String ln, String dob, String gen,
String hic, double score) {
         this.firstName = fn;
         this.lastName = ln;
         this.dob = dob;
         this.gender = gen;
         this.hic = hic;
         this.score = score;
   }
   /* other field goes on here */

   /** the score of my comparison result **/

public String getFirstName() {
      return firstName;
}

public void setFirstName(String firstName) {
      this.firstName = firstName;
}

public String getLastName() {
      return lastName;
}

public void setLastName(String lastName) {
      this.lastName = lastName;
}

public String getDob() {
      return dob;
}

public void setDob(String dob) {
      this.dob = dob;
}

public String getGender() {
```

```
	return gender;
}

public void setGender(String gender) {
	this.gender = gender;
}

public String getHic() {
	return hic;
}

public void setHic(String hic) {
	this.hic = hic;
}

public double getScore() {
	return score;
}

public void setScore(double score) {
	this.score = score;
}

public int compareTo(Object arg0) {
	MyTable table = (MyTable) arg0;
	if (this.score < table.score) return 1;
	else if (this.score == table.score) return 0;
	else return -1;
}


}
```

Appendix 3: Application Code 3 MyTableData

```
package dissertation.ie;

import java.util.ArrayList;
import java.util.List;

public class MyTableData {
	// just for initialized pupose, I need to assign the data
private List<MyTable> rows = new ArrayList<MyTable>();

public void addMyTableValue(MyTable lineItem) {
	rows.add(lineItem);
}
public List<MyTable> getRows() {
	return rows;
}

public void setRows(List<MyTable> rows) {
	this.rows = rows;
}
public void clean(){
	rows = new ArrayList<MyTable>();
}
}
```

Appendix 4: Application Code 4 StringScore

```
package dissertation.ie;

public interface StringScore {
	double score(String str1, String str2);
}
```

Appendix 5 Data Pre-Processing Code1 MainProcess

```
package data.process;

import java.io.BufferedReader;


public class Processor {
	String[] genders = new String[]{"Male", "Female"};

	public void processRawFile(String inputFilePath,String
		outputFilePath){
		File inFile = new File(inputFilePath);
		File outFile = new File(outputFilePath);
		BufferedReader bfReader = null;
		BufferedWriter bfWriter = null;
		try {
			bfReader= new BufferedReader(new
FileReader(inFile));
			bfWriter = new BufferedWriter(new
			    FileWriter(outFile));
			String line = bfReader.readLine();
			while (line != null) {
				String[] attrs = line.split(",");
				String givenName = attrs[1].trim();
				String surName = attrs[2].trim();
				String dob = attrs[9].trim();
				String social = attrs[12].trim();
				String gender = randomGender();
				String outputLine = gender + "," +
				  givenName+","+surName+","+dob+","+social;
				  bfWriter.write(outputLine);
				bfWriter.newLine();
				line = bfReader.readLine();
			}
			bfWriter.flush();

		} catch (FileNotFoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} finally{
			try {
				bfReader.close();
				bfWriter.close();
			} catch (IOException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		}

	}
```

```
public void generateRandomError(String inputFilePath,String
        outputFilePath){
      File inFile = new File(inputFilePath);
      File outFile = new File(outputFilePath);
      BufferedReader bfReader = null;
      BufferedWriter bfWriter = null;
      try {
            bfReader= new BufferedReader(new
               FileReader(inFile));
            bfWriter = new BufferedWriter(new
               FileWriter(outFile));
            String line = bfReader.readLine();
            while (line != null) {
                  String[] attrs = line.split(",");
                  String gender = attrs[0];
                  String givenName = attrs[1].trim();
                  String surName = attrs[2].trim();
                  String dob = attrs[3].trim();
                  String social = attrs[4].trim();
                  String[] attrs2 = {gender, givenName,
                     surName, dob, social};
                  randomError(attrs2);
                  String outputLine = attrs2[0]+ "," +
                     attrs2[1]+","+attrs2[2]+","+attrs2[3]+
                     ","+attrs2[4];
                  bfWriter.write(outputLine);
                  bfWriter.newLine();
                  line = bfReader.readLine();
            }
            bfWriter.flush();

      } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
      } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
      } finally{
            try {
                  bfReader.close();
                  bfWriter.close();
            } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
      }

}
public void processComparsion(String inputFilePath1, String
      inputFilePath2, String outputFilePath
         StringComparator stringComparator,
         Wrapper wrapper) {
      File inFile1 = new File(inputFilePath1);
      File inFile2 = new File(inputFilePath2);
      File outFile = new File(outputFilePath);
      BufferedReader bfReader1 = null;
      BufferedReader bfReader2 = null;
```

```
        BufferedWriter bfWriter = null;
        try {
              bfReader1= new BufferedReader(new
                   FileReader(inFile1));
              bfReader2= new BufferedReader(new
                   FileReader(inFile2));
              bfWriter = new BufferedWriter(new
                   FileWriter(outFile));
              String line1 = bfReader1.readLine();
              String line2 = bfReader2.readLine();
              while (line1 != null && line2 != null) {
                    double[] compareResultAttrs=lineComparsion(
                       line1, line2, stringComparator);
                    String outputLine =
                       simpleFormat(compareResultAttrs);
                    outputLine = wrapper.wrap(outputLine);
                    bfWriter.write(outputLine);
                    bfWriter.newLine();
                    line1 = bfReader1.readLine();
                    line2 = bfReader2.readLine();


              }
              bfWriter.flush();

        } catch (FileNotFoundException e) {
              // TODO Auto-generated catch block
              e.printStackTrace();
        } catch (IOException e) {
              // TODO Auto-generated catch block
              e.printStackTrace();
        } finally{
              try {
                    bfReader1.close();
                    bfReader2.close();
                    bfWriter.close();
              } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              }
        }
}
private String simpleFormat(double[] compareResultAttrs) {
        String[] double2Strings = new
                  String[compareResultAttrs.length];
        for(int i = 0; i < compareResultAttrs.length; ++i) {
              NumberFormat nformat = new DecimalFormat("0.00");
              double2Strings[i] =
                  nformat.format(compareResultAttrs[i]);
        }
        return simpleFormat(double2Strings);
}

private double[] lineComparsion(String line1, String line2,
              StringComparator stringComparator) {
        String[] attrs1 = line1.split(",");
        String[] attrs2 = line2.split(",");
```

```
            double[] result = new double[attrs1.length+1];
            for(int idx = 0; idx < attrs1.length; ++idx) {
                  result[idx]= stringComparsion(attrs1[idx],
                  attrs2[idx], stringComparator);
            }
            result[result.length-1] =
                  stringComparator.combineScore(result,
                  result.length-1);
            return result;
      }

      private double stringComparsion(String string, String string2,
                  StringComparator stringComparator) {
            return stringComparator.score(string, string2);
      }

      private String simpleFormat(String[] compareResultAttrs) {
                  String result = "";
            for(int i = 0; i < compareResultAttrs.length-1; ++i) {
                result += compareResultAttrs[i]+",";
            }
           result  +=compareResultAttrs[compareResultAttrs.length-1];
            return result;
      }

      private void randomError(String[] attr) {
            int numOfError = new Random().nextInt(5)+1;
            List<Integer> attrAvailable2Choose = new
                     ArrayList<Integer>();
            int attIndex = 5;
            for (int i = 0; i < attIndex; ++i) {
                 attrAvailable2Choose.add(i);
            }
//          int[] trackAttrError = new int[attIndex];
            while (numOfError > 0) {
                numOfError--;
                int attrIndexInAvailableList= new
                     Random().nextInt(attrAvailable2Choose.size());
                int attrToChoose = attrAvailable2Choose.get
                     (attrIndexInAvailableList);
                attrAvailable2Choose.remove(new
                     Integer(attrToChoose));
                attr[attrToChoose] = randomErrorForAttr
                    (attr[attrToChoose], attrToChoose);

            }
      }

      private String randomErrorForAttr(String string, int
               attrToChoose) {
            String result = null;
            switch (attrToChoose) {
            case 0: // gender
                  result = randomErrorString(string);
                  break;
            case 1: // given_name
                  result = randomErrorString(string);
```

```
                    break;
              case 2:
                    result = randomErrorString(string);
                    break;
              case 3:
                    result = randomErrorDigit(string);
                    break;
              case 4:
                    result = randomErrorDigit(string);
                    break;
              default:
                    break;
              }
              return result;
        }

        private String randomErrorDigit(String string) {
              if (string.length()==0) return string;
              int numOfError= new Random().nextInt(2)+1;
              char[] stringChar = string.toCharArray();
              while (numOfError >0) {
                   numOfError--;
              int attrIndex = new Random().nextInt(string.length());
                   stringChar[attrIndex] = (char) ('0'+new
                      Random().nextInt(10));
              }
              return new String(stringChar);
        }

        private String randomErrorString(String string) {
              if (string.length()==0) return string;
              int numOfError= new Random().nextInt(2)+1;
              char[] stringChar = string.toCharArray();
              while (numOfError >0) {
                   numOfError--;
                    int attrIndex = new
                      Random().nextInt(string.length());
                   stringChar[attrIndex] = (char) ('a'+new
                      Random().nextInt(26));
              }
              return new String(stringChar);
        }

        private String randomGender() {
              Random rand = new Random();
              int i = rand.nextInt(2);
              return genders[i];
        }

        public static void main(String[] args) {
              String strs = "098721";
              char[] strschar = strs.toCharArray();
              strschar[0] = (char)('0'+20);
              System.out.println(new String(strschar));
        }
}
```

Appendix 6 Data Pre-Processing Code2 MainProcess

```
package data.process;

import java.io.File;

/**
 * This is the main process of the data process.
 * The work class for our data processing job is class "Processor".
 * The "MainProcess" is a class organizer and call methods in
 * "Processor". For further design, see Processor.
 */
public class MainProcess {
	String directory = "D:\\dissertation\\code";

	/**
	 *
	 */
	public void preProcess() {
			extractAttrs();
			generateRandom();

	}

	/**
	 * This step generates random errors for the data. inFileName
	 * represents the input file. That is, the data which is
	 * extracted through method extractAttrs(). For each record in
	 * this file, several random errors are generated. The
	 * specific strategy of "random error" is shown on the method
	 * "generateRandomError(...).
	 *
	 */
	public void generateRandom() {
		String inFileName = "dataset.a.csv";
		String outFileName = "dataset.a.errored.csv";
		new Processor().generateRandomError(
			directory+File.separator+inFileName,
			directory+File.separator+outFileName);
	}

	/**
	 * The dataset_*_10000.csv is the original downloaded data
	 * set However, this data set contains many attributes which
	 * are not needed for this research. More specifically, the
	 * following attribute: first name, last name, gender, hic,
	 * age. Thus, these attributes are extracted from the original
	 * data, and written to a file.

	private void extractAttrs() {
		String inFileName = "dataset_C_10000.csv";
		String outFileName = "dataset.c.csv";

		new Processor().processRawFile(directory+
			File.separator+inFileName,
```

```
                directory+File.separator+outFileName);
        }

        /**
         * This is just a method delegator that delegates methods
         *        to the invocation of methods in "Processor".
         * @param inFileName1 the file to be compared,R1,
         *        with attributes (A1, A2,...)
         * @param inFileName2 the file to be compared,R2,
         *        with attributes (A1, A2,...)
         *        this is the one with random generated error.
         * @param outFileName comparison results are written to
         *        this file. Each record correspond to a record
         *        in the input file. The attributes are (A1,A2,...),
         *        here A1 is the score results of comparing the
         *        fields of (R1.A1, R2.A1). Other attributes are
         *        compared in the same way. A final combined score
         *        is also computed.
         * @param stringComparator As several different string
         *        comparison methods are provided in our approach,
         *        this parameter is required.
         * @param wrapper
         */

    public void processDelegator(String inFileName1, String inFileName2,
             String outFileName, StringComparator stringComparator,
              Wrapper wrapper) {
        String inPath1 = directory+File.separator + inFileName1;
        String inPath2 = directory+File.separator + inFileName2;
        String outPath = directory+File.separator+outFileName;
        Processor processor = new Processor();
        processor.processComparsion(inPath1, inPath2, outPath,
           stringComparator, wrapper);
        }

    // The following are different string comparison method.


    public void lev(String inFileName1, String inFileName2, String
              outFile, Wrapper wrapper) {
        StringComparator stringComparator = new LevComparator();
        processDelegator(inFileName1, inFileName2, outFile,
              stringComparator, wrapper);
        }

    public void jaro(String inFileName1, String inFileName2, String
              outFile, Wrapper wrapper) {
        StringComparator stringComparator = new JaroComparator();
        processDelegator(inFileName1, inFileName2, outFile,
              stringComparator, wrapper);
        }


    public void jaroWinkler(String inFileName1, String inFileName2,
               String outFile, Wrapper wrapper) {
        StringComparator stringComparator = new
               JaroWinklerComparator();
```

```
        processDelegator(inFileName1, inFileName2, outFile,
                stringComparator, wrapper);
        }
/**
 * compare dataset.a.csv with dataset.c.csv.
 * It is known that these two files are totally diiferent
 * data, they should not match, thus FalseWrapper is used.
 */

public void compareAandC() {
        String inFileName1 = "dataset.a.csv";
        String inFileName2 = "dataset.c.csv";
        Wrapper wrapper = new FalseWrapper();
        lev(inFileName1, inFileName2, "c.lev.csv", wrapper);
        jaro(inFileName1, inFileName2,"c.jaro.csv", wrapper);
        jaroWinkler(inFileName1, inFileName2, "c.jaroWinkler.csv",
             wrapper);
        }

/**
 * compare dataset.a.csv with dataset.a.errored.csv.
 * It is known that these two files contain same records
 * (the second file is generated from the first through
 * random error generator), they should match, thus
 * TrueWrapper is used.
 */

public void compareAandAError() {
        String inFileName1 = "dataset.a.csv";
        String inFileName2 = "dataset.a.errored.csv";
        Wrapper wrapper = new TrueWrapper();
        lev(inFileName1, inFileName2, "a.lev.csv", wrapper);
        jaro(inFileName1, inFileName2,"a.jaro.csv", wrapper);
        jaroWinkler(inFileName1, inFileName2, "a.jaroWinkler.csv",
             wrapper);
        }
public static void main(String[] args) {
        MainProcess mainProcess = new MainProcess();
//      new MainProcess().preProcess();
//      mainProcess.generateRandom();

        mainProcess.compareAandC();
        mainProcess.compareAandAError();

        }
}
```

Appendix 7 Data Pre-Processing Code 3 Processor

```
package data.process;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

/**
 * This class contains the main method of processing the data,
 * including 1)preprocess the raw data file such that a file
 * containing only attributes which are needed in this research.
 * 2)Generate random error, 3) compare two files (tables) with
 * same schema, generate score for each field.
 *
 */
public class Processor {
	String[] genders = new String[] { "Male", "Female" };

	private int minNumOfErrorEachLine = 2;

	private int maxNumOfErrorEachLine = 5;

	private int minErrorEachAttr = 1;

	/**
	 * extract useful attributes from this raw table.
	 * They are: gender, givenName, surName, DOB, hic.
	 *
	 * @param inputFilePath
	 * @param outputFilePath
	 */

public void processRawFile(String inputFilePath, String
outputFilePath) {
	File inFile = new File(inputFilePath);
	File outFile = new File(outputFilePath);
	BufferedReader bfReader = null;
	BufferedWriter bfWriter = null;
	try {
		bfReader = new BufferedReader(new FileReader(inFile));
		bfWriter = new BufferedWriter(new FileWriter(outFile));
		String line = bfReader.readLine();
		while (line != null) {
			String[] attrs = line.split(",");
			String givenName = attrs[1].trim();
```

```
                    String surName = attrs[2].trim();
                    String dob = attrs[9].trim();
                    String social = attrs[12].trim();
                    String gender = randomGender();
                    String outputLine = gender + "," + givenName + ","
                       + surname + "," + dob + "," + hic;
                    bfWriter.write(outputLine);
                    bfWriter.newLine();
                    line = bfReader.readLine();
               }
               bfWriter.flush();

          } catch (FileNotFoundException e) {
               // TODO Auto-generated catch block
               e.printStackTrace();
          } catch (IOException e) {
               // TODO Auto-generated catch block
               e.printStackTrace();
          } finally {
               try {
                    bfReader.close();
                    bfWriter.close();
               } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
               }
          }

}

/**
 * Read data into memory, for each line (record) of the file,
 * generate several random errors. The number of errors for
 * each record is randomly generated.
 *
 * @param inputFilePath
 * @param outputFilePath
 */

public void generateRandomError(String inputFilePath, String
        outputFilePath) {
     File inFile = new File(inputFilePath);
     File outFile = new File(outputFilePath);
     BufferedReader bfReader = null;
     BufferedWriter bfWriter = null;
     try {
         bfReader = new BufferedReader(new FileReader(inFile));
         bfWriter = new BufferedWriter(new FileWriter(outFile));
         String line = bfReader.readLine();
         while (line != null) {
            String[] attrs = line.split(",");
            String gender = attrs[0];
            String givenName = attrs[1].trim();
            String surName = attrs[2].trim();
            String dob = attrs[3].trim();
            String social = attrs[4].trim();
            String[] attrs2 = { gender, givenName, surName, dob,
```

```
                hic };
            randomError(attrs2);
            String outputLine = attrs2[0] + "," + attrs2[1] + ","
                + attrs2[2] + "," + attrs2[3] + "," + attrs2[4];
            bfWriter.write(outputLine);
            bfWriter.newLine();
            line = bfReader.readLine();
        }
        bfWriter.flush();

} catch (FileNotFoundException e) {
        / TODO Auto-generated catch block
        e.printStackTrace();
} catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
} finally {
     try {
         bfReader.close();
         bfWriter.close();
     } catch (IOException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
         }
    }

}


/**
 * @param inputFilePath1
 *      the file to be compared,R1, with attributes (A1, A2,...)
 * @param inputFilePath2
 *      the file to be compared,R2, with attributes (A1, A2,...)
 *      this is the one with random generated error.
 * @param outputFilePath
 *      comparison results are written to this file. Each record
 *      correspond to a record in the input file. The attributes
 *      are (A1,A2,...), here A1 is the score results of comparing
 *      the fields of (R1.A1, R2.A1). Other attributes are compared
 *      in the same way. A final combined score is also computed.
 * @param stringComparator
 * @param wrapper
 */

public void processComparsion(String inputFilePath1, String
          inputFilePath2, String outputFilePath, StringComparator
          stringComparator, Wrapper wrapper) {
        File inFile1 = new File(inputFilePath1);
        File inFile2 = new File(inputFilePath2);
        File outFile = new File(outputFilePath);
        BufferedReader bfReader1 = null;
        BufferedReader bfReader2 = null;
        BufferedWriter bfWriter = null;
        try {
             bfReader1 = new BufferedReader(new FileReader(inFile1));
             bfReader2 = new BufferedReader(new FileReader(inFile2));
```

```
            bfWriter = new BufferedWriter(new FileWriter(outFile));
            String line1 = bfReader1.readLine();
            String line2 = bfReader2.readLine();
           while (line1 != null && line2 != null) {
              // String[] compareResultAttrs=
              // lineComparsion(line1,line2, stringComparator);
              Double[] compareResultAttrs = lineComparsion(line1,
                 line2, stringComparator);
              String outputLine = simpleFormat(compareResultAttrs);
              outputLine = wrapper.wrap(outputLine);
              bfWriter.write(outputLine);
              bfWriter.newLine();
              line1 = bfReader1.readLine();
              line2 = bfReader2.readLine();

            }
            bfWriter.flush();


      } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
      } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
      } finally {
            try {
                bfReader1.close();
                bfReader2.close();
                bfWriter.close();
            } catch (IOException e) {
                 // TODO Auto-generated catch block
                 e.printStackTrace();
            }
      }
}


private String simpleFormat(Double[] compareResultAttrs) {
      String[] double2Strings = new
            String[compareResultAttrs.length];
      for (int i = 0; i < compareResultAttrs.length; ++i) {
            NumberFormat nformat = new DecimalFormat("0.00");
            double2Strings[i] = nformat.format
              (compareResultAttrs[i]);
            }
            return simpleFormat(double2Strings);
      }



/**
 *
 * compare two records based on the string comparing methods:
 * stringComparator. it could be jaroComparator,
 * jaroWinklerComparator, or levComparator.
 *
```

```
 * @param  line1
 * @param  line2
 * @param  stringComparator
 * @return the score for each attribute is computed and returned.
 */

private Double[] lineComparsion(String line1, String line2,
        StringComparator stringComparator) {
      String[] attrs1 = line1.split(",");
      String[] attrs2 = line2.split(",");
      Double[] result = new Double[attrs1.length + 1];
      for (int idx = 0; idx < attrs1.length; ++idx) {
            result[idx] = stringComparsion(attrs1[idx], attrs2[idx],
                  stringComparator);

      }
      result[result.length - 1] =
          stringComparator.combineScore(result, result.length - 1);
      return result;
}


private double stringComparsion(String string, String string2,
      StringComparator stringComparator) {
      return stringComparator.score(string, string2);
      }

private String simpleFormat(String[] compareResultAttrs) {
      String result = "";
      for (int i = 0; i < compareResultAttrs.length - 1; ++i) {
            result += compareResultAttrs[i] + ",";
      }
      result += compareResultAttrs[compareResultAttrs.length - 1];
      return result;
}


/**
 * For each records of the input file, denoted by attr some
 * random errors are generated to them. The number of errors in
 * each record numOfErrorEachLine, is between
 * [minNumOfErrorEachLine, maxNumOfErrorEachLine. This number is
 * uniformly randomly generated. Notice that not all fields are
 * required to have randomly generated errors, thus after having
 * numOfErrorEachLine being generated, the fields for generating
 * errors are uniformly randomly chosen.
 *
 * Then for each of these chosen fields, errors are generated for
 * them.
 *
 * @param attr
 */

private void randomError(String[] attr) {
   int numOfErrorEachLine = new
Random().nextInt(maxNumOfErrorEachLine
              minNumOfErrorEachLine + 1)
```

```
                + minNumOfErrorEachLine;
    List<Integer> attrAvailable2Choose = new ArrayList<Integer>();
    int attIndex = 5;
    for (int i = 0; i < attIndex; ++i) {
       attrAvailable2Choose.add(i);
   }
   // int[] trackAttrError = new int[attIndex];
   while (numOfErrorEachLine > 0) {
       numOfErrorEachLine--;
       int attrIndexInAvailableList = new Random()
             .nextInt(attrAvailable2Choose.size());
       int attrToChoose = attrAvailable2Choose
             .get(attrIndexInAvailableList);
       attrAvailable2Choose.remove(new Integer(attrToChoose));
             attr[attrToChoose] = randomErrorForAttr
              (attr[attrToChoose], attrToChoose);
    }
}


private String randomErrorForAttr(String string, int attrToChoose) {
      String result = null;
      switch (attrToChoose) {
      case 0: // gender
            result = randomErrorString(string, 'a', 26);
            break;
      case 1: // given_name
            result = randomErrorString(string, 'a', 26);
            break;
      case 2:
            result = randomErrorString(string, 'a', 26);
            break;
      case 3:
      result = randomErrorString(string, '0', 10);
            break;
      case 4:
            result = randomErrorString(string, '0', 10);
            break;
      default:
            break;
      }
      return result;
      }


private String randomErrorDigit(String string) {
      if (string.length() == 0)
            return string;
      int numOfError = new Random().nextInt(2) + 1;
      char[] stringChar = string.toCharArray();
      while (numOfError > 0) {
            numOfError--;
            int attrIndex = new Random().nextInt(string.length());
            stringChar[attrIndex] = (char) ('0' + new
                Random().nextInt(10));
      }
      return new String(stringChar);
```

```
}


private String randomErrorString(String string, char startC, int
range)
      {
      if (string.length() == 0)
            return string;
      minErrorEachAttr = 1;
      int maxErrorEachAttr = ceiling(string);
      int numOfError = new Random().nextInt(maxErrorEachAttr
                  - minErrorEachAttr + 1)+ minErrorEachAttr;
      char[] stringChar = string.toCharArray();
      while (numOfError > 0) {
            numOfError--;
            int attrIndex = new Random().nextInt(string.length());
            stringChar[attrIndex] = (char) (startC + new Random()
                        .nextInt(range));
      }
      return new String(stringChar);
  }


private int ceiling(String string) {
      int maxErrorEachAttr = string.length() / 2;
      if (string.length() % 2 == 1) maxErrorEachAttr++;
      return maxErrorEachAttr;
  }


private String randomGender() {
      Random rand = new Random();
      int i = rand.nextInt(2);
      return genders[i];
  }

public static void main(String[] args) {
      String strs = "098721";
      char[] strschar = strs.toCharArray();
      strschar[0] = (char) ('0' + 20);
      System.out.println(new String(strschar));

      NumberFormat nformat = new DecimalFormat("0.00");
      String nullStr = nformat.format(null);
      System.out.println("nullstr: " + nullStr); // exception
   }
}
```

Appendix 8: Data Pre-Processing Code 4 StringComparator

```
package data.process;

/**
 * An interface to be implemented based on different comparison
 * mechanism.compare score of two strings.
 */
public interface StringComparator {
	public double score(String str1, String str2);
	public double combineScore(Double[] scores);
	public double combineScore(Double[] result, int length);
}
```

Appendix 9: Data Pre-Processing Code 5 Wrapper

```
package data.process;

public interface Wrapper {

	String wrap(String outputLine);

}
```

Appendix 10: Data Pre-Processing Code 6 TrueWrapper

```
package data.process;

public class TrueWrapper implements Wrapper {

	@Override
	public String wrap(String outputLine) {
		return outputLine+"," + "1";
	}

}
```

Appendix 11: Data Pre-Processing Code 7 FalseWrapper

```
package data.process;

public class FalseWrapper implements Wrapper{

	@Override
	public String wrap(String outputLine) {
		return outputLine+"," + "0";
	}

}
```

Appendix 12: Data Pre-Processing Code 8 JaroComparator

```
package data.process;

import com.wcohen.ss.Jaro;


/**
 * @author DW
 *
 */
public class JaroComparator implements StringComparator {

	/* (non-Javadoc)
	 * @see data.process.StringComparator#combineScore(double[])
	 */
	@Override
	public double combineScore(Double[] scores) {
		// TODO Auto-generated method stub
		return 0;
	}

	/* (non-Javadoc)
	 * @see data.process.StringComparator#combineScore(double[],
int)
	 */
	@Override
	public double combineScore(Double[] result, int length) {
		double overall = 0;
		for(int i = 0; i < length; ++i) { // note, length not
			// necessary tobe result.length()
			overall+=result[i];
		}
		return overall/length;
	}

	/* (non-Javadoc)
	 * @see data.process.StringComparator#score
	 * (java.lang.String, java.lang.String)
	 */
	@Override
	public double score(String str1, String str2) {
		Jaro jaro = new Jaro();
		double scoree = jaro.score(str1, str2);
		return scoree;
	}

}
```

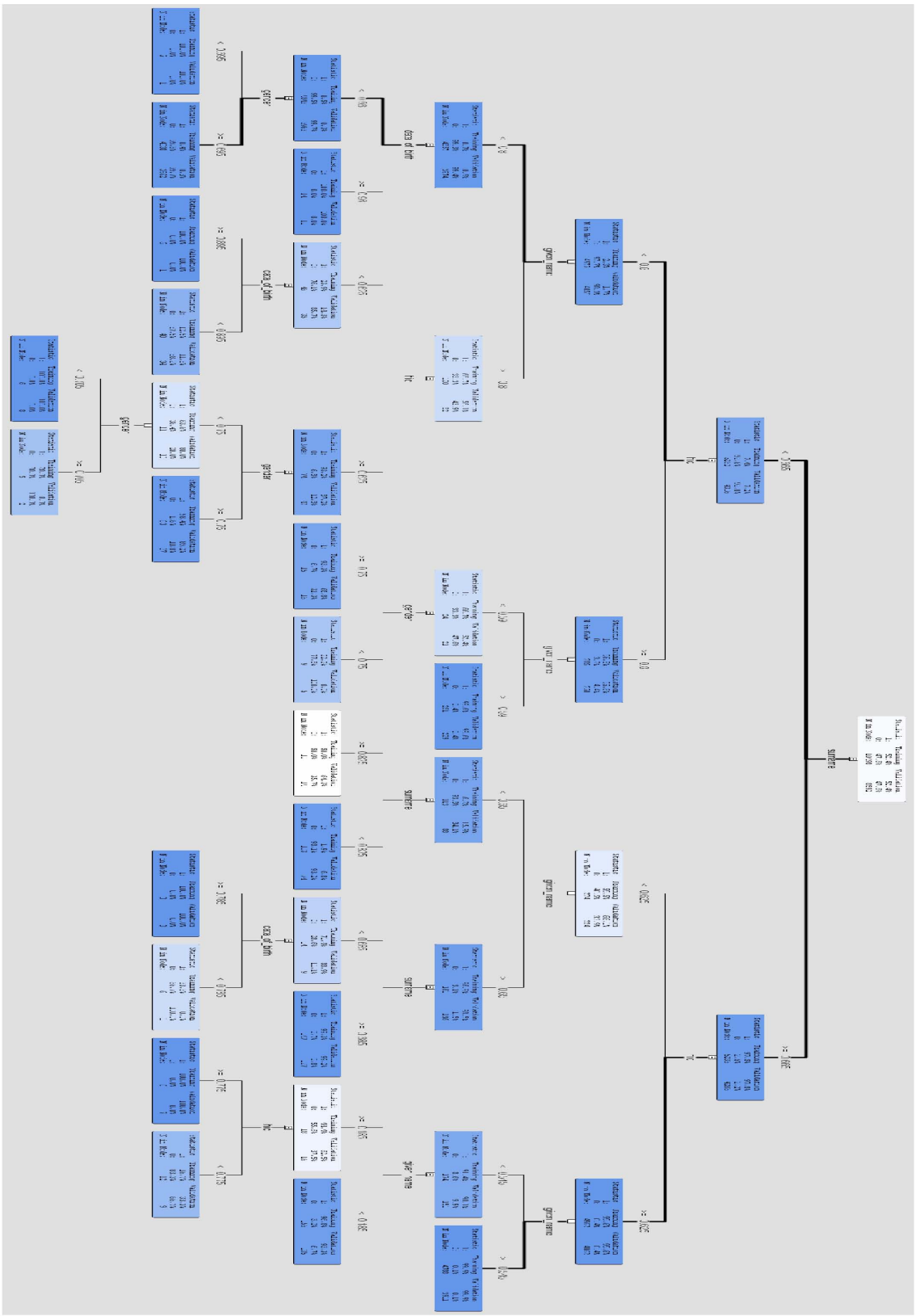
Appendix 13: Data Pre-Processing Code 9 JaroWinklerComparator

```
package data.process;

import com.wcohen.ss.Jaro;
import com.wcohen.ss.JaroWinkler;


/**
 * @author DW
 *
 */
public class JaroWinklerComparator implements StringComparator {

	/* (non-Javadoc)
	 * @see data.process.StringComparator#combineScore(double[])
	 */
	@Override
	public double combineScore(Double[] scores) {
		// TODO Auto-generated method stub
		return 0;
	}

	/* (non-Javadoc)
	 * @see data.process.StringComparator#combineScore(double[],
	 * int)
	 */
	@Override
	public double combineScore(Double[] result, int length) {
		double overall = 0;
		for(int i = 0; i < length; ++i) { // note, length not
			//necessary tobe result.length()
			overall+=result[i];
		}
		return overall/length;
	}

	/* (non-Javadoc)
	 * @see data.process.StringComparator#score(java.lang.String,
	 * java.lang.String)
	 */
	@Override
	public double score(String str1, String str2) {
		JaroWinkler jaroWinkler = new JaroWinkler();
		double scoree = jaroWinkler.score(str1, str2);
		return scoree;
	}

}
```

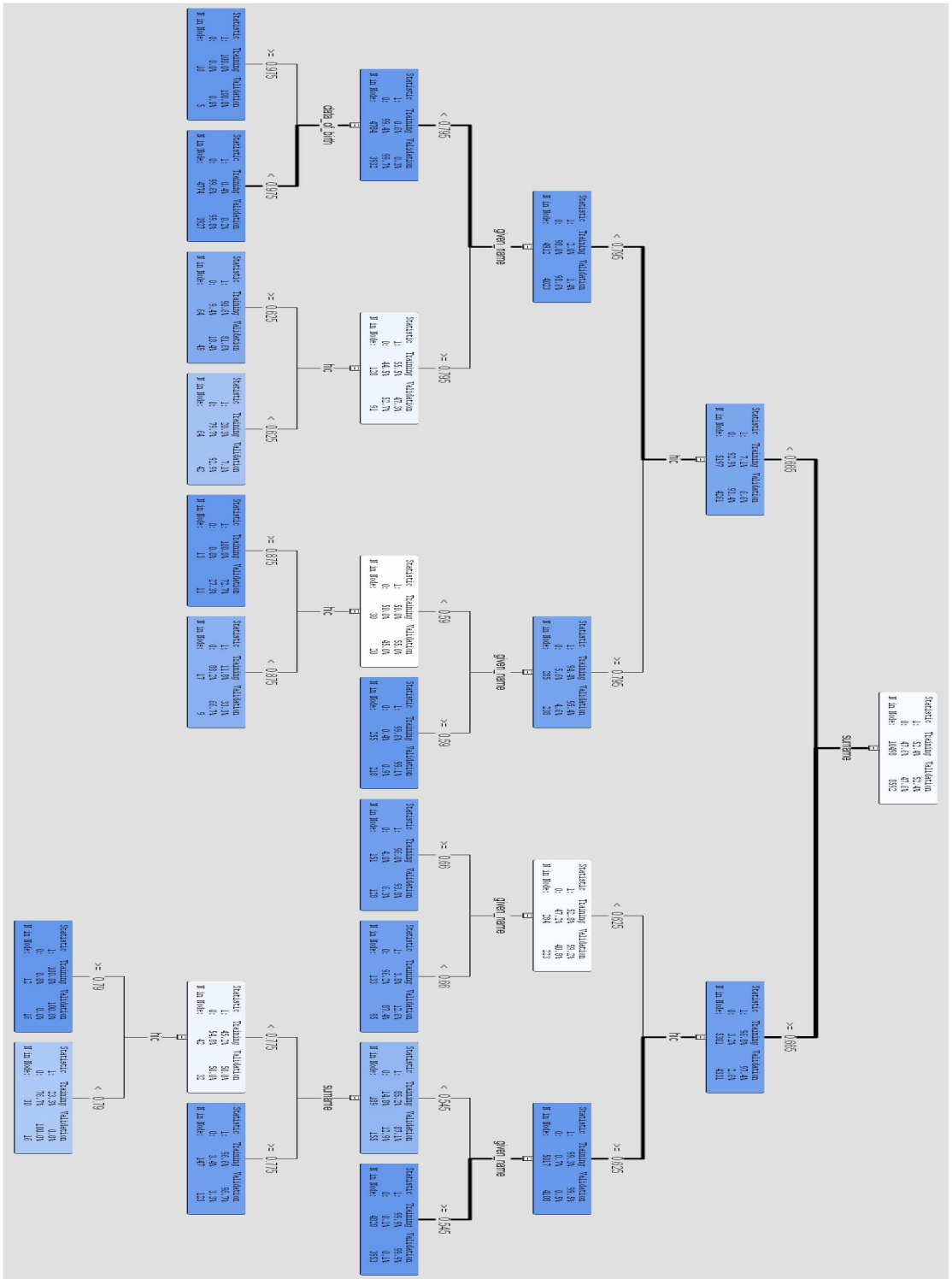
Appendix 14: Data Pre-Processing Code 10 LevenshteinComparator

```
package data.process;

import com.wcohen.ss.Levenstein;

/**
 * @author DW
 *
 */
public class LevComparator implements StringComparator {

	/* (non-Javadoc)
	 * @see data.process.StringComparator#combineScore(double[])
	 */
	@Override
	public double combineScore(Double[] scores) {
		// TODO Auto-generated method stub
		return 0;
	}

	/* (non-Javadoc)
	 * @see data.process.StringComparator#combineScore(double[],
	 * int)
	 */
	@Override
	public double combineScore(Double[] result, int length) {
		double overall = 0;
		for(int i = 0; i < length; ++i) { // note, length not
necessary tobe result.length()
			overall+=result[i];
		}
		return overall/length;
	}

	/* (non-Javadoc)
	 * @see data.process.StringComparator#score(java.lang.String,
	 * java.lang.String)
	 */
	@Override
	public double score(String str1, String str2) {
		Levenstein lev = new Levenstein();
		double scoree = lev.score(str1, str2);
		int length =
str1.length()>str2.length()?str1.length():str2.length();
		return 1-Math.abs(scoree)/(double)length;
	}

}
```

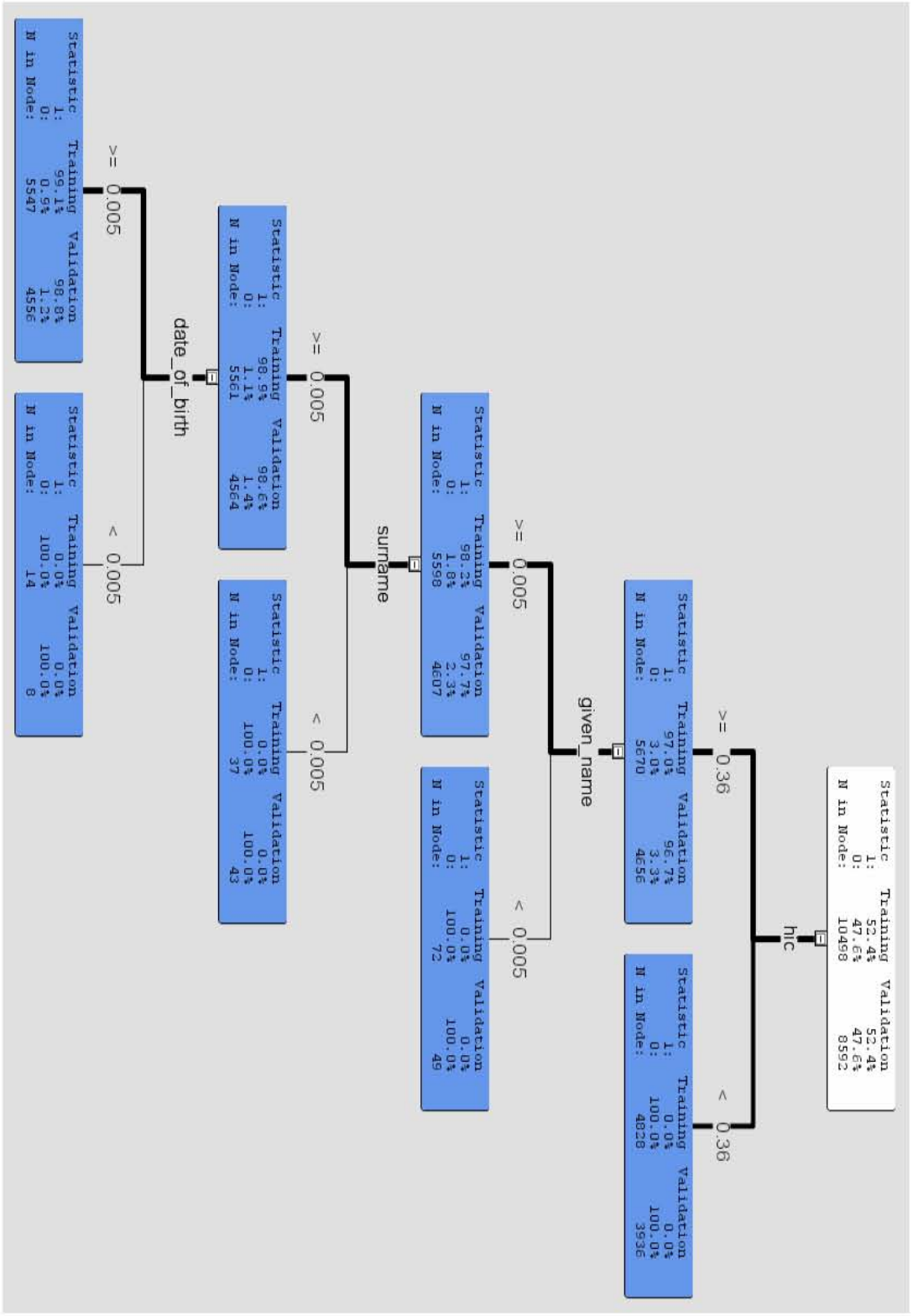
Appendix 15: Decision Tree Model Generated By Using the Jaro String Comparison Function with the Modified Data Set



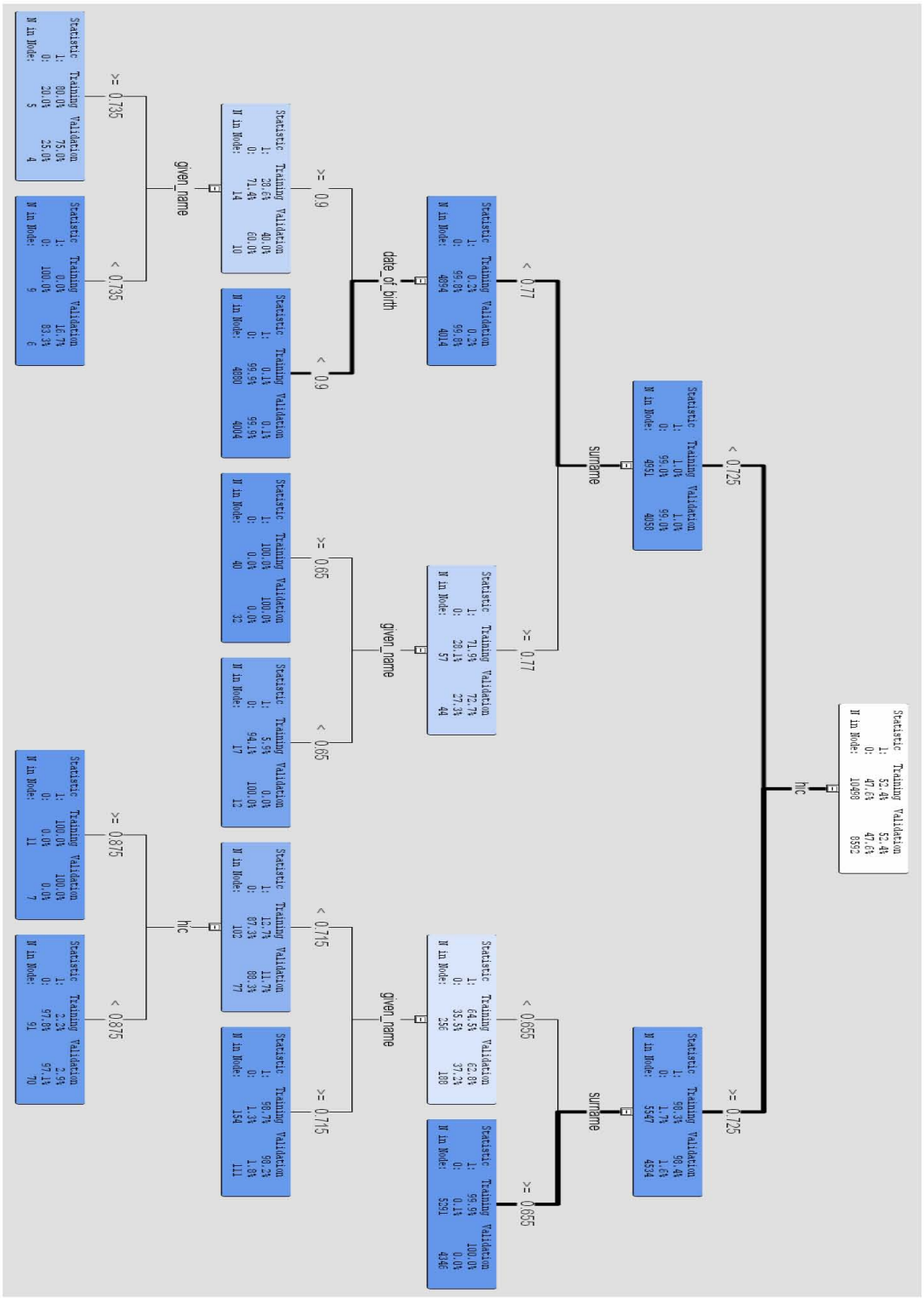
Appendix 16: Decision Tree Model Generated by Using the Jaro-Winkler Function with the Modified Data Set



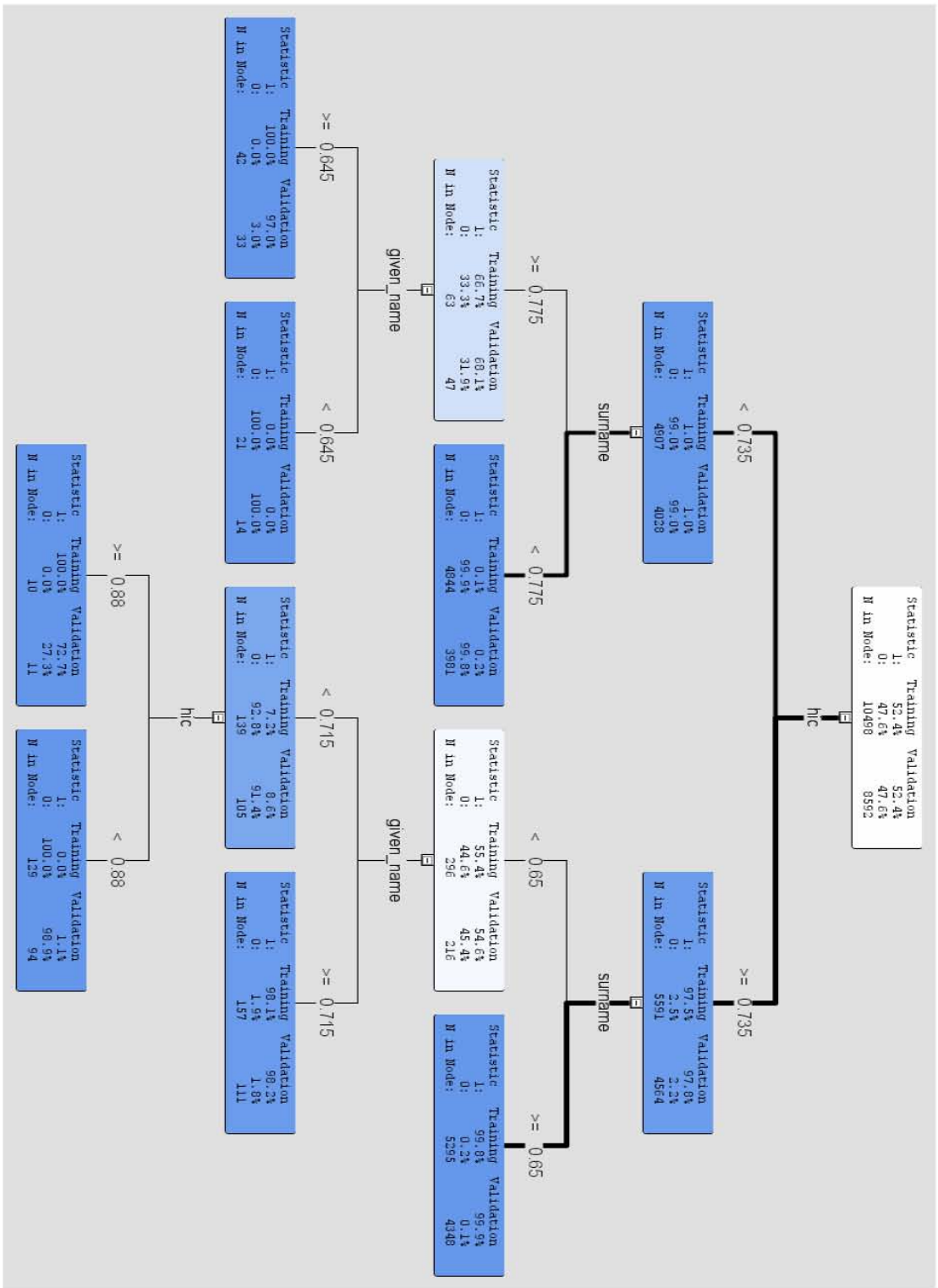
Appendix 17: Decision Tree Model Generated by Using the Levenshtein Edit Distance Function with the Modified Data Set



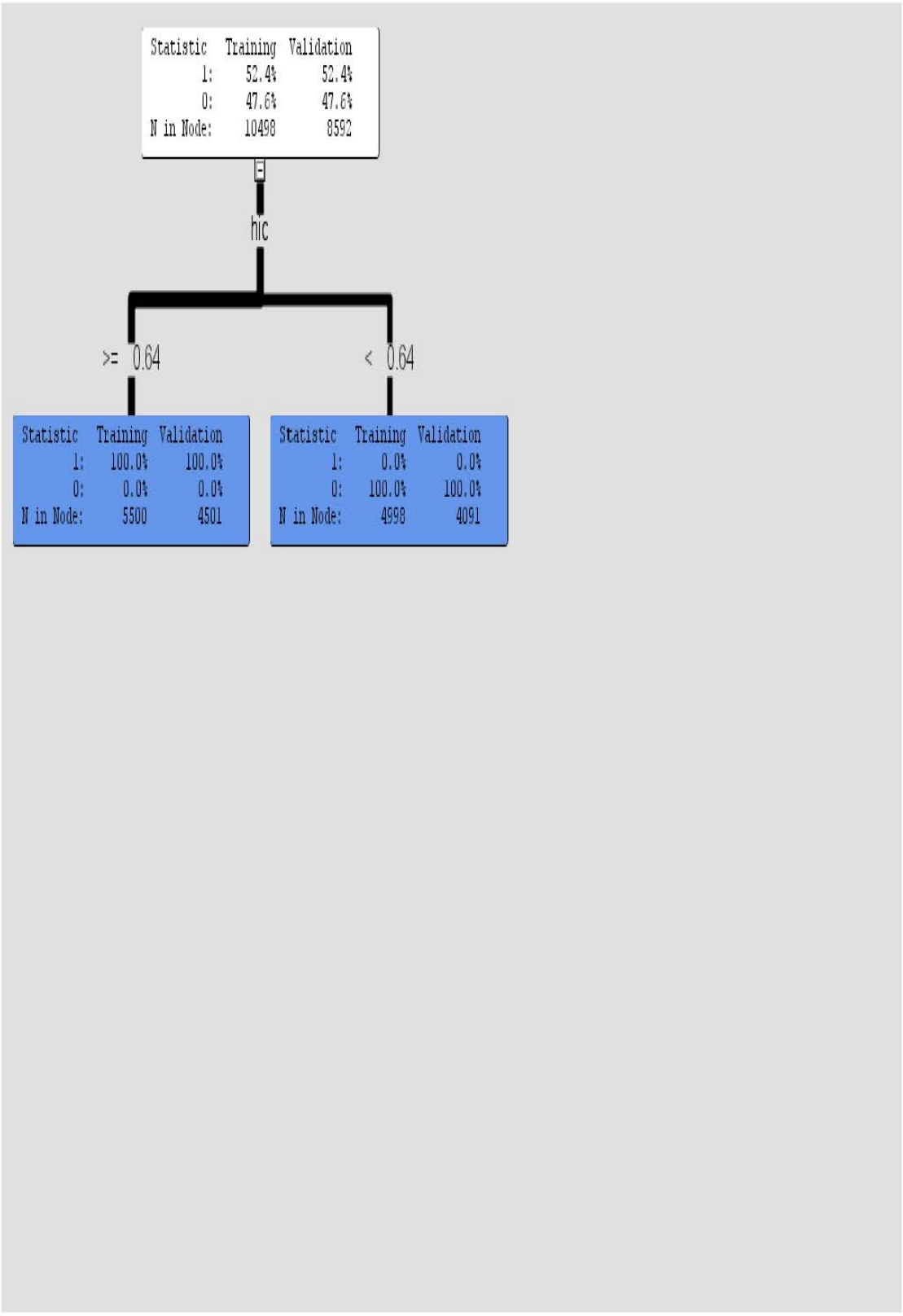
Appendix 18: Decision Tree Model Generated by Using the Jaro String Comparison Function with the First Data Set



Appendix 19: Decision Tree Model Generated by Using the Jaro-Winkler Function with the First Data Set



Appendix 20: Decision Tree Model Generated by Using the Levenshtein Edit Distance Function with the First Data Set



Appendix 21 Decision Model (Jaro) Output of the First Data Set Generated By SAS Enterprise Miner

```
*------------------------------------------------------------*
User:                dwang
Date:                28JAN08
Time:                10:55
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*
```

Variable Summary

Role	Measurement Level	Frequency Count
INPUT	INTERVAL	5
TARGET	BINARY	1

Model Events

Target	Event	Measurement Level	Number of Levels	Order	Label
matching_status	1	BINARY	2	Descending	matching_status

Predicted and decision variables

Type	Variable	Label
TARGET	matching_status	matching_status
PREDICTED	P_matching_status1	Predicted: matching_status=1
RESIDUAL	R_matching_status1	Residual: matching_status=1
PREDICTED	P_matching_status0	Predicted: matching_status=0
RESIDUAL	R_matching_status0	Residual: matching_status=0
FROM	F_matching_status	From: matching_status
INTO	I_matching_status	Into: matching_status

Variable Importance

Obs	NAME	LABEL	NRULES	IMPORTANCE	VIMPORTANCE	RATIO
1	hic	hic	2	1.00000	1.00000	1.00000
2	surname	surname	2	0.15500	0.15306	0.98753
3	given_name	given_name	3	0.15275	0.14608	0.95635
4	date_of_birth	date_of_birth	1	0.02136	0.02675	1.25251

Tree Leaf Report

Node	Depth	Training Observations	Percent 1	Validation Observations	Percent V 1

7	2	5291	1.00	4346	1.00
8	3	4880	0.00	4004	0.00
13	3	154	0.99	111	0.98
20	4	91	0.02	70	0.03
11	3	40	1.00	32	1.00
10	3	17	0.06	12	0.00
21	4	11	1.00	7	1.00
18	4	9	0.00	6	0.17
19	4	5	0.80	4	0.75

--
* Score Output
--

--
* Report Output
--

Fit Statistics

Target=matching_status

Fit Statistics	Statistics Label	Train	Validation
NOBS	Sum of Frequencies	10498.00	8592.00
SUMW	Sum of Case Weights Times Freq	20996.00	17184.00
MISC	Misclassification Rate	0.00	0.00
MAX	Maximum Absolute Error	1.00	1.00
SSE	Sum of Squared Errors	31.32	21.43
ASE	Average Squared Error	0.00	0.00
RASE	Root Average Squared Error	0.04	0.04
DIV	Divisor for ASE	20996.00	17184.00
DFT	Total Degrees of Freedom	10498.00	.

Classification Table

Data Role=TRAIN Target Variable=matching_status

Target	Outcome	Target Percentage	Outcome Percentage	Frequency Count	Total Percentage
0	0	99.8599	99.8200	4990	47.5329
1	0	0.1401	0.1273	7	0.0667
0	1	0.1636	0.1800	9	0.0857
1	1	99.8364	99.8727	5492	52.3147

Data Role=VALIDATE Target Variable=matching_status

Target	Outcome	Target Percentage	Outcome Percentage	Frequency Count	Total Percentage
0	0	99.8534	99.8778	4086	47.5559
1	0	0.1466	0.1333	6	0.0698
0	1	0.1111	0.1222	5	0.0582
1	1	99.8889	99.8667	4495	52.3161

Event Classification Table

Data Role=TRAIN Target=matching_status

False Negative	True Negative	False Positive	True Positive
7	4990	9	5492

Data Role=VALIDATE Target=matching_status

False Negative	True Negative	False Positive	True Positive
6	4086	5	4495

Assessment Score Rankings

Data Role=TRAIN Target Variable=matching_status

Posterior Percentile	Gain	Lift	Cumulative Lift	% Response	Cumulative % Response	Observation Number	Probability Mean
0	.	.	.	.	.	.	.
5	90.7120	1.90712	1.90712	99.8976	99.8976	524.9	0.99898
10	90.7015	1.90691	1.90701	99.8866	99.8921	524.9	0.99887
15	90.6980	1.90691	1.90698	99.8866	99.8903	524.9	0.99887
20	90.6962	1.90691	1.90696	99.8866	99.8894	524.9	0.99887
25	90.6952	1.90691	1.90695	99.8866	99.8888	524.9	0.99887
30	90.6945	1.90691	1.90694	99.8866	99.8884	524.9	0.99887
35	90.6940	1.90691	1.90694	99.8866	99.8882	524.9	0.99887
40	90.6936	1.90691	1.90694	99.8866	99.8880	524.9	0.99887
45	90.6933	1.90691	1.90693	99.8866	99.8878	524.9	0.99887
50	90.6931	1.90691	1.90693	99.8866	99.8877	524.9	0.99887
55	81.6904	0.91664	1.81690	48.0148	95.1720	524.9	0.48015
60	66.5626	0.00156	1.66563	0.0820	87.2478	524.9	0.00082
65	53.7621	0.00156	1.53762	0.0820	80.5428	524.9	0.00082
70	42.7903	0.00156	1.42790	0.0820	74.7956	524.9	0.00082
75	33.2814	0.00156	1.33281	0.0820	69.8146	524.9	0.00082
80	24.9610	0.00156	1.24961	0.0820	65.4564	524.9	0.00082
85	17.6196	0.00156	1.17620	0.0820	61.6108	524.9	0.00082
90	11.0939	0.00156	1.11094	0.0820	58.1925	524.9	0.00082
95	5.2551	0.00156	1.05255	0.0820	55.1341	524.9	0.00082
100	0.0000	0.00154	1.00000	0.0806	52.3814	524.9	0.00081

Data Role=VALIDATE Target Variable=matching_status

Posterior Percentile	Gain	Lift	Cumulative Lift	% Response	Cumulative % Response	Observation Number	Probability Mean
0	.	.	.	.	.	.	.
5	90.8110	1.90811	1.90811	99.9582	99.9582	429.6	0.99897
10	90.8071	1.90803	1.90807	99.9540	99.9561	429.6	0.99887
15	90.8057	1.90803	1.90806	99.9540	99.9554	429.6	0.99887
20	90.8051	1.90803	1.90805	99.9540	99.9550	429.6	0.99887
25	90.8047	1.90803	1.90805	99.9540	99.9548	429.6	0.99887
30	90.8044	1.90803	1.90804	99.9540	99.9547	429.6	0.99887
35	90.8042	1.90803	1.90804	99.9540	99.9546	429.6	0.99887
40	90.8041	1.90803	1.90804	99.9540	99.9545	429.6	0.99887
45	90.8040	1.90803	1.90804	99.9540	99.9544	429.6	0.99887

50	90.8039	1.90803	1.90804	99.9540	99.9544	429.6	0.99887
55	81.6609	0.90232	1.81661	47.2688	95.1648	429.6	0.47491
60	66.5345	0.00143	1.66534	0.0749	87.2406	429.6	0.00082
65	53.7351	0.00143	1.53735	0.0749	80.5356	429.6	0.00082
70	42.7643	0.00143	1.42764	0.0749	74.7884	429.6	0.00082
75	33.2562	0.00143	1.33256	0.0749	69.8075	429.6	0.00082
80	24.9366	0.00143	1.24937	0.0749	65.4492	429.6	0.00082
85	17.5958	0.00143	1.17596	0.0749	61.6037	429.6	0.00082
90	11.0706	0.00143	1.11071	0.0749	58.1854	429.6	0.00082
95	5.2323	0.00143	1.05232	0.0749	55.1270	429.6	0.00082
100	0.0000	0.00585	1.00000	0.3067	52.3859	429.6	0.00081

Assessment Score Distribution

Data Role=TRAIN Target Variable=matching_status

Posterior Probability Range	Number of Events	Number of Nonevents	Posterior Probability Mean	Percentage
0.95 - 1.00	5488	8	0.99854	52.3528
0.90 - 0.95	0	0	.	0.0000
0.85 - 0.90	0	0	.	0.0000
0.80 - 0.85	4	1	0.80000	0.0476
0.75 - 0.80	0	0	.	0.0000
0.70 - 0.75	0	0	.	0.0000
0.65 - 0.70	0	0	.	0.0000
0.60 - 0.65	0	0	.	0.0000
0.55 - 0.60	0	0	.	0.0000
0.50 - 0.55	0	0	.	0.0000
0.45 - 0.50	0	0	.	0.0000
0.40 - 0.45	0	0	.	0.0000
0.35 - 0.40	0	0	.	0.0000
0.30 - 0.35	0	0	.	0.0000
0.25 - 0.30	0	0	.	0.0000
0.20 - 0.25	0	0	.	0.0000
0.15 - 0.20	0	0	.	0.0000
0.10 - 0.15	0	0	.	0.0000
0.05 - 0.10	1	16	0.05882	0.1619
0.00 - 0.05	6	4974	0.00120	47.4376

Data Role=VALIDATE Target Variable=matching_status

Posterior Probability Range	Number of Events	Number of Nonevents	Posterior Probability Mean	Percentage
0.95 - 1.00	4492	4	0.99858	52.3277
0.90 - 0.95	0	0	.	0.0000
0.85 - 0.90	0	0	.	0.0000
0.80 - 0.85	3	1	0.80000	0.0466
0.75 - 0.80	0	0	.	0.0000
0.70 - 0.75	0	0	.	0.0000
0.65 - 0.70	0	0	.	0.0000
0.60 - 0.65	0	0	.	0.0000
0.55 - 0.60	0	0	.	0.0000
0.50 - 0.55	0	0	.	0.0000
0.45 - 0.50	0	0	.	0.0000
0.40 - 0.45	0	0	.	0.0000
0.35 - 0.40	0	0	.	0.0000
0.30 - 0.35	0	0	.	0.0000
0.25 - 0.30	0	0	.	0.0000
0.20 - 0.25	0	0	.	0.0000
0.15 - 0.20	0	0	.	0.0000
0.10 - 0.15	0	0	.	0.0000

0.05 - 0.10	0	12	0.05882	0.1397
0.00 - 0.05	6	4074	0.00118	47.4860

Appendix 22 Decision Model (Jaro-Winkler) Output of the First Data Set Generated by SAS Enterprise Miner

```
*------------------------------------------------------------*
User:                dwang
Date:                18JAN08
Time:                13:43
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*
```

Variable Summary

Role	Measurement Level	Frequency Count
INPUT	INTERVAL	5
TARGET	BINARY	1

Model Events

Target	Event	Measurement Level	Number of Levels	Order	Label
matching_status	1	BINARY	2	Descending	matching_status

Decision matrix

matching_ status	Training Proportions	1	0
1	0.52383	1	0
0	0.47617	0	1

Predicted and decision variables

Type	Variable	Label
TARGET	matching_status	matching_status
PREDICTED	P_matching_status1	Predicted: matching_status=1
RESIDUAL	R_matching_status1	Residual: matching_status=1
PREDICTED	P_matching_status0	Predicted: matching_status=0
RESIDUAL	R_matching_status0	Residual: matching_status=0
FROM	F_matching_status	From: matching_status
INTO	I_matching_status	Into: matching_status
MODELDECISION	D_Dup1	Decision: matching_status
EXPECTEDPROFIT	EP_MATCHING_STATUS	Expected Profit: matching_status
COMPUTEDPROFIT	CP_MATCHING_STATUS	Computed Profit: matching_status
BESTPROFIT	BP_MATCHING_STATUS	Best Profit: matching_status

Variable Importance

Obs	NAME	LABEL	NRULES	IMPORTANCE	VIMPORTANCE	RATIO

1	hic	hic	2	1.00000	1.00000	1.00000
2	surname	surname	2	0.18419	0.17796	0.96615
3	given_name	given_name	2	0.17515	0.16175	0.92347

Tree Leaf Report

Node	Depth	Training Observations	Percent 1	Validation Observations	Percent V 1
7	2	5295	1.00	4348	1.00
4	2	4844	0.00	3981	0.00
13	3	157	0.98	111	0.98
18	4	129	0.00	94	0.01
11	3	42	1.00	33	0.97
10	3	21	0.00	14	0.00
19	4	10	1.00	11	0.73

--
* Score Output
--

--
* Report Output
--

Fit Statistics

Target=matching_status

Fit Statistics	Statistics Label	Train	Validation
NOBS	Sum of Frequencies	10498.00	8592.00
SUMW	Sum of Case Weights Times Freq	20996.00	17184.00
MISC	Misclassification Rate	0.00	0.00
MAX	Maximum Absolute Error	1.00	1.00
SSE	Sum of Squared Errors	37.83	35.90
ASE	Average Squared Error	0.00	0.00
RASE	Root Average Squared Error	0.04	0.05
DIV	Divisor for ASE	20996.00	17184.00
DFT	Total Degrees of Freedom	10498.00	.
APROF	Average Profit for matching_status	1.00	1.00
PROF	Total Profit for matching_status	10479.00	8574.00

Classification Table

Data Role=TRAIN Target Variable=matching_status

Target	Outcome	Target Percentage	Outcome Percentage	Frequency Count	Total Percentage
0	0	99.8598	99.7600	4987	47.5043
1	0	0.1402	0.1273	7	0.0667
0	1	0.2180	0.2400	12	0.1143
1	1	99.7820	99.8727	5492	52.3147

Data Role=VALIDATE Target Variable=matching_status

Target	Outcome	Target Percentage	Outcome Percentage	Frequency Count	Total Percentage
0	0	99.8044	99.7556	4081	47.4977
1	0	0.1956	0.1777	8	0.0931
0	1	0.2221	0.2444	10	0.1164
1	1	99.7779	99.8223	4493	52.2928

Event Classification Table

Data Role=TRAIN Target=matching_status

False Negative	True Negative	False Positive	True Positive
7	4987	12	5492

Data Role=VALIDATE Target=matching_status

False Negative	True Negative	False Positive	True Positive
8	4081	10	4493

Assessment Score Rankings

Data Role=TRAIN Target Variable=matching_status

Posterior Percentile	Gain	Lift	Cumulative Lift	% Response	Cumulative % Response	Observation Number	Probability Mean
0	.	.	.	.	.	.	.
5	90.6151	1.90615	1.90615	99.8469	99.8469	524.9	0.99847
10	90.5990	1.90583	1.90599	99.8300	99.8384	524.9	0.99830
15	90.5937	1.90583	1.90594	99.8300	99.8356	524.9	0.99830
20	90.5910	1.90583	1.90591	99.8300	99.8342	524.9	0.99830
25	90.5894	1.90583	1.90589	99.8300	99.8334	524.9	0.99830
30	90.5883	1.90583	1.90588	9.8300	99.8328	524.9	0.99830
35	90.5875	1.90583	1.90588	99.8300	99.8324	524.9	0.99830
40	90.5870	1.90583	1.90587	99.8300	99.8321	524.9	0.99830
45	90.5865	1.90583	1.90587	99.8300	99.8319	524.9	0.99830
50	90.5862	1.90583	1.90586	99.8300	99.8317	524.9	0.99830
55	81.5996	0.91734	1.81600	48.0517	95.1244	524.9	0.48052
60	66.4893	0.00276	1.66489	0.1445	87.2094	524.9	0.00145
65	53.7037	0.00276	1.53704	0.1445	80.5121	524.9	0.00145
70	42.7445	0.00276	1.42745	0.1445	74.7716	524.9	0.00145
75	33.2466	0.00276	1.33247	0.1445	69.7965	524.9	0.00145
80	24.9360	0.00276	1.24936	0.1445	65.4432	524.9	0.00145
85	17.6030	0.00276	1.17603	0.1445	61.6021	524.9	0.00145
90	11.0848	0.00276	1.11085	0.1445	58.1878	524.9	0.00145
95	5.2528	0.00276	1.05253	0.1445	55.1329	524.9	0.00145
100	0.0000	0.00197	1.00000	0.1032	52.3814	524.9	0.00103

Data Role=VALIDATE Target Variable=matching_status

Posterior Percentile	Gain	Lift	Cumulative Lift	% Response	Cumulative % Response	Observation Number	Probability Mean
0	.	.	.	.	.	.	.
5	88.9559	1.88956	1.88956	98.9863	98.9863	429.6	0.99847
10	89.8356	1.90715	1.89836	99.9080	99.4472	429.6	0.99830
15	90.1288	1.90715	1.90129	99.9080	99.6008	429.6	0.99830
20	90.2755	1.90715	1.90275	99.9080	99.6776	429.6	0.99830
25	90.3634	1.90715	1.90363	99.9080	99.7237	429.6	0.99830
30	90.4221	1.90715	1.90422	99.9080	99.7544	429.6	0.99830
35	90.4640	1.90715	1.90464	99.9080	99.7763	429.6	0.99830
40	90.4954	1.90715	1.90495	99.9080	99.7928	429.6	0.99830
45	90.5198	1.90715	1.90520	99.9080	99.8056	429.6	0.99830
50	90.5394	1.90715	1.90539	99.9080	99.8158	429.6	0.99830
55	81.5108	0.91226	1.81511	47.7894	95.0862	429.6	0.47728
60	66.4129	0.00336	1.66413	0.1758	87.1770	429.6	0.00145
65	53.6377	0.00336	1.53638	0.1758	80.4846	429.6	0.00145
70	42.6876	0.00336	1.42688	0.1758	74.7482	429.6	0.00145
75	33.1975	0.00336	1.33197	0.1758	69.7767	429.6	0.00145
80	24.8936	0.00336	1.24894	0.1758	65.4267	429.6	0.00145
85	17.5667	0.00336	1.17567	0.1758	61.5884	429.6	0.00145
90	11.0538	0.00336	1.11054	0.1758	58.1766	429.6	0.00145
95	5.2265	0.00336	1.05227	0.1758	55.1239	429.6	0.00145
100	0.0000	0.00696	1.00000	0.3644	52.3859	429.6	0.00108

Assessment Score Distribution

Data Role=TRAIN Target Variable=matching_status

Posterior Probability Range	Number of Events	Number of Nonevents	Posterior Probability Mean	Percentage
0.95 - 1.00	5492	12	0.99782	52.4290
0.90 - 0.95	0	0	.	0.0000
0.85 - 0.90	0	0	.	0.0000
0.80 - 0.85	0	0	.	0.0000
0.75 - 0.80	0	0	.	0.0000
0.70 - 0.75	0	0	.	0.0000
0.65 - 0.70	0	0	.	0.0000
0.60 - 0.65	0	0	.	0.0000
0.55 - 0.60	0	0	.	0.0000
0.50 - 0.55	0	0	.	0.0000
0.45 - 0.50	0	0	.	0.0000
0.40 - 0.45	0	0	.	0.0000
0.35 - 0.40	0	0	.	0.0000
0.30 - 0.35	0	0	.	0.0000
0.25 - 0.30	0	0	.	0.0000
0.20 - 0.25	0	0	.	0.0000
0.15 - 0.20	0	0	.	0.0000
0.10 - 0.15	0	0	.	0.0000
0.05 - 0.10	0	0	.	0.0000
0.00 - 0.05	7	4987	0.00140	47.5710

Data Role=VALIDATE Target Variable=matching_status

Posterior Probability Range	Number of Events	Number of Nonevents	Posterior Probability Mean	Percentage
0.95 - 1.00	4493	10	0.99789	52.4092
0.90 - 0.95	0	0	.	0.0000
0.85 - 0.90	0	0	.	0.0000
0.80 - 0.85	0	0	.	0.0000

0.75 - 0.80	0	0	.	0.0000
0.70 - 0.75	0	0	.	0.0000
0.65 - 0.70	0	0	.	0.0000
0.60 - 0.65	0	0	.	0.0000
0.55 - 0.60	0	0	.	0.0000
0.50 - 0.55	0	0	.	0.0000
0.45 - 0.50	0	0	.	0.0000
0.40 - 0.45	0	0	.	0.0000
0.35 - 0.40	0	0	.	0.0000
0.30 - 0.35	0	0	.	0.0000
0.25 - 0.30	0	0	.	0.0000
0.20 - 0.25	0	0	.	0.0000
0.15 - 0.20	0	0	.	0.0000
0.10 - 0.15	0	0	.	0.0000
0.05 - 0.10	0	0	.	0.0000
0.00 - 0.05	8	4081	0.00141	47.5908

Appendix 23 Decision Model (Levenshtein) Output of the First Data Set Generated by SAS Enterprise Miner

```
*------------------------------------------------------------*
User:                dwang
Date:                28JAN08
Time:                11:23
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*
```

Variable Summary

Role	Measurement Level	Frequency Count
INPUT	INTERVAL	5
TARGET	BINARY	1

Model Events

Target	Event	Measurement Level	Number of Levels	Order	Label
matching_status	1	BINARY	2	Descending	matching_status

Decision matrix

matching_ status	Training Proportions	1	0
1	0.52383	1	0
0	0.47617	0	1

Predicted and decision variables

Type	Variable	Label
TARGET	matching_status	matching_status
PREDICTED	P_matching_status1	Predicted: matching_status=1
RESIDUAL	R_matching_status1	Residual: matching_status=1
PREDICTED	P_matching_status0	Predicted: matching_status=0
RESIDUAL	R_matching_status0	Residual: matching_status=0
FROM	F_matching_status	From: matching_status
INTO	I_matching_status	Into: matching_status
MODELDECISION	D_Dup1	Decision: matching_status
EXPECTEDPROFIT	EP_MATCHING_STATUS	Expected Profit: matching_status
COMPUTEDPROFIT	CP_MATCHING_STATUS	Computed Profit: matching_status
BESTPROFIT	BP_MATCHING_STATUS	Best Profit: matching_status

Variable Importance

Obs	NAME	LABEL	NRULES	IMPORTANCE	VIMPORTANCE	RATIO

1	hic	hic	1	1	1	1

Tree Leaf Report

Node	Depth	Training Observations	Percent 1	Validation Observations	Percent V 1
3	1	5500	1	4501	1
2	1	4998	0	4091	0

--
* Score Output
--

--
* Report Output
--

Fit Statistics

Target=matching_status

Fit Statistics	Statistics Label	Train	Validation
NOBS	Sum of Frequencies	10498.00	8592.00
SUMW	Sum of Case Weights Times Freq	20996.00	17184.00
MISC	Misclassification Rate	0.00	0.00
MAX	Maximum Absolute Error	1.00	0.00
SSE	Sum of Squared Errors	2.00	0.00
ASE	Average Squared Error	0.00	0.00
RASE	Root Average Squared Error	0.01	0.00
DIV	Divisor for ASE	20996.00	17184.00
DFT	Total Degrees of Freedom	10498.00	.
APROF	Average Profit for matching_status	1.00	1.00
PROF	Total Profit for matching_status	10497.00	8592.00

Classification Table

Data Role=TRAIN Target Variable=matching_status

Target	Outcome	Target Percentage	Outcome Percentage	Frequency Count	Total Percentage
0	0	100.000	99.980	4998	47.6091
0	1	0.018	0.020	1	0.0095
1	1	99.982	100.000	5499	52.3814

Data Role=VALIDATE Target Variable=matching_status

Target	Outcome	Target Percentage	Outcome Percentage	Frequency Count	Total Percentage
0	0	100	100	4091	47.6141
1	1	100	100	4501	52.3859

Event Classification Table

Data Role=TRAIN Target=matching_status

False Negative	True Negative	False Positive	True Positive
0	4998	1	5499

Data Role=VALIDATE Target=matching_status

False Negative	True Negative	False Positive	True Positive
0	4091	.	4501

Assessment Score Rankings

Data Role=TRAIN Target Variable=matching_status

Posterior Percentile	Gain	Lift	Cumulative Lift	% Response	Cumulative % Response	Observation Number	Probability Mean
0	.	.	.	.	.	.	.
5	90.8727	1.90873	1.90873	99.9818	99.9818	524.9	0.99982
10	90.8727	1.90873	1.90873	99.9818	99.9818	524.9	0.99982
15	90.8727	1.90873	1.90873	99.9818	99.9818	524.9	0.99982
20	90.8727	1.90873	1.90873	99.9818	99.9818	524.9	0.99982
25	90.8727	1.90873	1.90873	99.9818	99.9818	524.9	0.99982
30	90.8727	1.90873	1.90873	99.9818	99.9818	524.9	0.99982
35	90.8727	1.90873	1.90873	99.9818	99.9818	524.9	0.99982
40	90.8727	1.90873	1.90873	99.9818	99.9818	524.9	0.99982
45	90.8727	1.90873	1.90873	99.9818	99.9818	524.9	0.99982
50	90.8727	1.90873	1.90873	99.9818	99.9818	524.9	0.99982
55	81.8182	0.91273	1.81818	47.8099	95.2389	524.9	0.47810
60	66.6667	0.00000	1.66667	0.0000	87.3023	524.9	0.00000
65	53.8462	0.00000	1.53846	0.0000	80.5868	524.9	0.00000
70	42.8571	0.00000	1.42857	0.0000	74.8306	524.9	0.00000
75	33.3333	0.00000	1.33333	0.0000	69.8419	524.9	0.00000
80	25.0000	0.00000	1.25000	0.0000	65.4768	524.9	0.00000
85	17.6471	0.00000	1.17647	0.0000	61.6252	524.9	0.00000
90	11.1111	0.00000	1.11111	0.0000	58.2016	524.9	0.00000
95	5.2632	0.00000	1.05263	0.0000	55.1383	524.9	0.00000
100	0.0000	0.00000	1.00000	0.0000	52.3814	524.9	0.00000

Data Role=VALIDATE Target Variable=matching_status

Posterio Percentile	Gain	Lift	Cumulative Lift	% Response	Cumulative % Response	Observation Number	Probability Mean
0	.	.	.	.	.	.	.
5	90.8909	1.90891	1.90891	100.000	100.000	429.6	0.99982
10	90.8909	1.90891	1.90891	100.000	100.000	429.6	0.99982
15	90.8909	1.90891	1.90891	100.000	100.000	429.6	0.99982
20	90.8909	1.90891	1.90891	100.000	100.000	429.6	0.99982
25	90.8909	1.90891	1.90891	100.000	100.000	429.6	0.99982
30	90.8909	1.90891	1.90891	100.000	100.000	429.6	0.99982
35	90.8909	1.90891	1.90891	100.000	100.000	429.6	0.99982
40	90.8909	1.90891	1.90891	100.000	100.000	429.6	0.99982
45	90.8909	1.90891	1.90891	100.000	100.000	429.6	0.99982
50	90.8909	1.90891	1.90891	100.000	100.000	429.6	0.99982
55	81.8182	0.91091	1.81818	47.719	95.247	429.6	0.47710
60	66.6667	0.00000	1.66667	0.000	87.310	429.6	0.00000
65	53.8462	0.00000	1.53846	0.000	80.594	429.6	0.00000

70	42.8571	0.00000	1.42857	0.000	74.837	429.6	0.00000
75	33.3333	0.00000	1.33333	0.000	69.848	429.6	0.00000
80	25.0000	0.00000	1.25000	0.000	65.482	429.6	0.00000
85	17.6471	0.00000	1.17647	0.000	61.631	429.6	0.00000
90	11.1111	0.00000	1.11111	0.000	58.207	429.6	0.00000
95	5.2632	0.00000	1.05263	0.000	55.143	429.6	0.00000
100	0.0000	0.00000	1.00000	0.000	52.386	429.6	0.00000

Assessment Score Distribution

Data Role=TRAIN Target Variable=matching_status

Posterior Probability Range	Number of Events	Number of Nonevents	Posterior Probability Mean	Percentage
0.95 - 1.00	5499	1	0.99982	52.3909
0.90 - 0.95	0	0	.	0.0000
0.85 - 0.90	0	0	.	0.0000
0.80 - 0.85	0	0	.	0.0000
0.75 - 0.80	0	0	.	0.0000
0.70 - 0.75	0	0	.	0.0000
0.65 - 0.70	0	0	.	0.0000
0.60 - 0.65	0	0	.	0.0000
0.55 - 0.60	0	0	.	0.0000
0.50 - 0.55	0	0	.	0.0000
0.45 - 0.50	0	0	.	0.0000
0.40 - 0.45	0	0	.	0.0000
0.35 - 0.40	0	0	.	0.0000
0.30 - 0.35	0	0	.	0.0000
0.25 - 0.30	0	0	.	0.0000
0.20 - 0.25	0	0	.	0.0000
0.15 - 0.20	0	0	.	0.0000
0.10 - 0.15	0	0	.	0.0000
0.05 - 0.10	0	0	.	0.0000
0.00 - 0.05	0	4998	0.00000	47.6091

Data Role=VALIDATE Target Variable=matching_status

Posterior Probability Range	Number of Events	Number of Nonevents	Posterior Probability Mean	Percentage
0.95 - 1.00	4501	0	0.99982	52.3859
0.90 - 0.95	0	0	.	0.0000
0.85 - 0.90	0	0	.	0.0000
0.80 - 0.85	0	0	.	0.0000
0.75 - 0.80	0	0	.	0.0000
0.70 - 0.75	0	0	.	0.0000
0.65 - 0.70	0	0	.	0.0000
0.60 - 0.65	0	0	.	0.0000
0.55 - 0.60	0	0	.	0.0000
0.50 - 0.55	0	0	.	0.0000
0.45 - 0.50	0	0	.	0.0000
0.40 - 0.45	0	0	.	0.0000
0.35 - 0.40	0	0	.	0.0000
0.30 - 0.35	0	0	.	0.0000
0.25 - 0.30	0	0	.	0.0000
0.20 - 0.25	0	0	.	0.0000
0.15 - 0.20	0	0	.	0.0000
0.10 - 0.15	0	0	.	0.0000
0.05 - 0.10	0	0	.	0.0000
0.00 - 0.05	0	4091	0.00000	47.6141

Appendix 24 Decision Model (Jaro) Output of the Modified Data Set Generated by SAS Enterprise Miner

```
*------------------------------------------------------------*
User:                dwang
Date:                30JAN08
Time:                11:50
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*
```

Variable Summary

Role	Measurement Level	Frequency Count
INPUT	INTERVAL	5
TARGET	BINARY	1

Model Events

Target	Event	Measurement Level	Number of Levels	Order	Label
matching_status	1	BINARY	2	Descending	matching_status

Decision matrix

matching_ status	Training Proportions	1	0
1	0.52383	1	0
0	0.47617	0	1

Predicted and decision variables

Type	Variable	Label
TARGET	matching_status	matching_status
PREDICTED	P_matching_status1	Predicted: matching_status=1
RESIDUAL	R_matching_status1	Residual: matching_status=1
PREDICTED	P_matching_status0	Predicted: matching_status=0
RESIDUAL	R_matching_status0	Residual: matching_status=0
FROM	F_matching_status	From: matching_status
INTO	I_matching_status	Into: matching_status
MODELDECISION	D_Dup1	Decision: matching_status
EXPECTEDPROFIT	EP_MATCHING_STATUS	Expected Profit: matching_status
COMPUTEDPROFIT	CP_MATCHING_STATUS	Computed Profit: matching_status
BESTPROFIT	BP_MATCHING_STATUS	Best Profit: matching_status

Variable Importance

Obs	NAME	LABEL	NRULES	IMPORTANCE	VIMPORTANCE	RATIO
1	surname	surname	3	1.00000	1.00000	1.00000
2	hic	hic	4	0.37443	0.37086	0.99045
3	given_name	given_name	5	0.23073	0.19390	0.84038
4	data_of_birth	data_of_birth	3	0.09525	0.08581	0.90090
5	gender	gender	4	0.07077	0.04636	0.65504

Tree Leaf Report

Node	Depth	Training Observations	Percent 1	Validation Observations	Percent V 1
31	5	4838	0.00	3962	0.00
15	3	4788	1.00	3911	1.00
11	3	261	1.00	229	1.00
26	4	156	0.97	135	0.93
25	4	147	0.99	127	0.99
22	4	103	0.02	74	0.07
35	5	63	0.98	37	0.89
32	5	40	0.13	34	0.12
21	4	15	0.93	16	0.69
17	4	14	1.00	11	1.00
42	5	12	0.17	9	0.33
23	4	10	0.50	14	0.64
20	4	9	0.22	5	0.00
39	5	8	1.00	8	1.00
33	5	6	1.00	1	1.00
50	6	6	1.00	8	1.00
43	5	6	1.00	7	1.00
38	5	6	0.33	1	0.00
30	5	5	1.00	1	1.00
51	6	5	0.20	2	0.00

--
* Score Output
--

--
* Report Output
--

Fit Statistics

Target=matching_status

Fit Statistics	Statistics Label	Train	Validation
NOBS	Sum of Frequencies	10498.00	8592.00
SUMW	Sum of Case Weights Times Freq	20996.00	17184.00
MISC	Misclassification Rate	0.00	0.01
MAX	Maximum Absolute Error	1.00	1.00
SSE	Sum of Squared Errors	85.75	92.72
ASE	Average Squared Error	0.00	0.01
RASE	Root Average Squared Error	0.06	0.07
DIV	Divisor for ASE	20996.00	17184.00
DFT	Total Degrees of Freedom	10498.00	.
APROF	Average Profit for matching_status	1.00	0.99
PROF	Total Profit for matching_status	10450.00	8542.00

Classification Table

Data Role=TRAIN Target Variable=matching_status

Target	Outcome	Target Percentage	Outcome Percentage	Frequency Count	Total Percentage
0	0	99.3816	99.6599	4982	47.4567
1	0	0.6184	0.5637	31	0.2953
0	1	0.3099	0.3401	17	0.1619
1	1	99.6901	99.4363	5468	52.0861

Data Role=VALIDATE Target Variable=matching_status

Target	Outcome	Target Percentage	Outcome Percentage	Frequency Count	Total Percentage
0	0	99.4372	99.3400	4064	47.2998
1	0	0.5628	0.5110	23	0.2677
0	1	0.5993	0.6600	27	0.3142
1	1	99.4007	99.4890	4478	52.1182

Event Classification Table

Data Role=TRAIN Target=matching_status

False Negative	True Negative	False Positive	True Positive
31	4982	17	5468

Data Role=VALIDATE Target=matching_status

False Negative	True Negative	False Positive	True Positive
23	4064	27	4478

Assessment Score Rankings

Data Role=TRAIN Target Variable=matching_status

Posterior Percentile	Gain	Lift	Cumulative Lift	% Response	Cumulative % Response	Observation Number	Probability Mean
0	.	.	.	.	.	.	.
5	90.7981	1.90798	1.90798	99.9427	99.9427	524.9	0.99943
10	90.7929	1.90788	1.90793	99.9373	99.9400	524.9	0.99937
15	90.7912	1.90788	1.90791	99.9373	99.9391	524.9	0.99937
20	90.7904	1.90788	1.90790	99.9373	99.9387	524.9	0.99937
25	90.7899	1.90788	1.90790	99.9373	99.9384	524.9	0.99937
30	90.7895	1.90788	1.90790	99.9373	99.9382	524.9	0.99937
35	90.7893	1.90788	1.90789	99.9373	99.9381	524.9	0.99937
40	90.7891	1.90788	1.90789	99.9373	99.9380	524.9	0.99937
45	90.7890	1.90788	1.90789	99.9373	99.9379	524.9	0.99937
50	90.7210	1.90109	1.90721	99.5818	99.9023	524.9	0.99582
55	81.2693	0.86753	1.81269	45.4424	94.9514	524.9	0.45442
60	66.2195	0.00671	1.66219	0.3514	87.0681	524.9	0.00351
65	53.4849	0.00671	1.53485	0.3514	80.3976	524.9	0.00351
70	42.5696	0.00671	1.42570	0.3514	74.6800	524.9	0.00351
75	33.1097	0.00671	1.33110	0.3514	69.7247	524.9	0.00351

80	24.8323	0.00671	1.24832	0.3514	65.3889	524.9	0.00351
85	17.5287	0.00671	1.17529	0.3514	61.5632	524.9	0.00351
90	11.0366	0.00671	1.11037	0.3514	58.1625	524.9	0.00351
95	5.2279	0.00671	1.05228	0.3514	55.1198	524.9	0.00351
100	0.0000	0.00671	1.00000	0.3514	52.3814	524.9	0.00351

Data Role=VALIDATE Target Variable=matching_status

Posterior Percentile	Gain	Lift	Cumulative Lift	% Response	Cumulative % Response	Observation Number	Probability Mean
0	.	.	.	.	.	.	.
5	90.8015	1.90801	1.90801	99.9531	99.9531	429.6	0.99943
10	90.7974	1.90793	1.90797	99.9489	99.9510	429.6	0.99937
15	90.7960	1.90793	1.90796	99.9489	99.9503	429.6	0.99937
20	90.7953	1.90793	1.90795	99.9489	99.9499	429.6	0.99937
25	90.7949	1.90793	1.90795	99.9489	99.9497	429.6	0.99937
30	90.7947	1.90793	1.90795	99.9489	99.9496	429.6	0.99937
35	90.7945	1.90793	1.90794	99.9489	99.9495	429.6	0.99937
40	90.7943	1.90793	1.90794	99.9489	99.9494	429.6	0.99937
45	90.7942	1.90793	1.90794	99.9489	99.9493	429.6	0.99937
50	90.7156	1.90008	1.90716	99.5377	99.9082	429.6	0.99594
55	81.3846	0.88074	1.81385	46.1383	95.0200	429.6	0.47798
60	66.3133	0.00530	1.66313	0.2776	87.1248	429.6	0.00351
65	53.5608	0.00530	1.53561	0.2776	80.4443	429.6	0.00351
70	42.6300	0.00530	1.42630	0.2776	74.7181	429.6	0.00351
75	33.1567	0.00530	1.33157	0.2776	69.7554	429.6	0.00351
80	24.8675	0.00530	1.24868	0.2776	65.4130	429.6	0.00351
85	17.5535	0.00530	1.17554	0.2776	61.5815	429.6	0.00351
90	11.0522	0.00530	1.11052	0.2776	58.1758	429.6	0.00351
95	5.2353	0.00530	1.05235	0.2776	55.1285	429.6	0.00351
100	0.0000	0.00530	1.00000	0.2776	52.3859	429.6	0.00351

Assessment Score Distribution

Data Role=TRAIN Target Variable=matching_status

Posterior Probability Range	Number of Events	Number of Nonevents	Posterior Probability Mean	Percentage
0.95 - 1.00	5449	11	0.99799	52.0099
0.90 - 0.95	14	1	0.93333	0.1429
0.85 - 0.90	0	0	.	0.0000
0.80 - 0.85	0	0	.	0.0000
0.75 - 0.80	0	0	.	0.0000
0.70 - 0.75	0	0	.	0.0000
0.65 - 0.70	0	0	.	0.0000
0.60 - 0.65	0	0	.	0.0000
0.55 - 0.60	0	0	.	0.0000
0.50 - 0.55	5	5	0.50000	0.0953
0.45 - 0.50	0	0	.	0.0000
0.40 - 0.45	0	0	.	0.0000
0.35 - 0.40	0	0	.	0.0000
0.30 - 0.35	2	4	0.33333	0.0572
0.25 - 0.30	0	0	.	0.0000
0.20 - 0.25	3	11	0.21429	0.1334
0.15 - 0.20	2	10	0.16667	0.1143
0.10 - 0.15	5	35	0.12500	0.3810
0.05 - 0.10	0	0	.	0.0000
0.00 - 0.05	19	4922	0.00385	47.0661

Data Role=VALIDATE Target Variable=matching_status

Posterior Probability Range	Number of Events	Number of Nonevents	Posterior Probability Mean	Percentage
0.95 - 1.00	4458	17	0.99797	52.0833
0.90 - 0.95	11	5	0.93333	0.1862
0.85 - 0.90	0	0	.	0.0000
0.80 - 0.85	0	0	.	0.0000
0.75 - 0.80	0	0	.	0.0000
0.70 - 0.75	0	0	.	0.0000
0.65 - 0.70	0	0	.	0.0000
0.60 - 0.65	0	0	.	0.0000
0.55 - 0.60	0	0	.	0.0000
0.50 - 0.55	9	5	0.50000	0.1629
0.45 - 0.50	0	0	.	0.0000
0.40 - 0.45	0	0	.	0.0000
0.35 - 0.40	0	0	.	0.0000
0.30 - 0.35	0	1	0.33333	0.0116
0.25 - 0.30	0	0	.	0.0000
0.20 - 0.25	0	7	0.21587	0.0815
0.15 - 0.20	3	6	0.16667	0.1047
0.10 - 0.15	4	30	0.12500	0.3957
0.05 - 0.10	0	0	.	0.0000
0.00 - 0.05	16	4020	0.00381	46.9739

Appendix 25 Decision Model (Jaro-Winkler) Output of the Modified Data Set Generated by SAS Enterprise Miner

```
*------------------------------------------------------------*
User:                 dwang
Date:                 30JAN08
Time:                 11:50
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*
```

Variable Summary

Role	Measurement Level	Frequency Count
INPUT	INTERVAL	5
TARGET	BINARY	1

Model Events

Target	Event	Measurement Level	Number of Levels	Order	Label
matching_status	1	BINARY	2	Descending	matching_status

Decision matrix

matching_ status	Training Proportions	1	0
1	0.52383	1	0
0	0.47617	0	1

Predicted and decision variables

Type	Variable	Label
TARGET	matching_status	matching_status
PREDICTED	P_matching_status1	Predicted: matching_status=1
RESIDUAL	R_matching_status1	Residual: matching_status=1
PREDICTED	P_matching_status0	Predicted: matching_status=0
RESIDUAL	R_matching_status0	Residual: matching_status=0
FROM	F_matching_status	From: matching_status
INTO	I_matching_status	Into: matching_status
MODELDECISION	D_Dup1	Decision: matching_status
EXPECTEDPROFIT	EP_MATCHING_STATUS	Expected Profit: matching_status
COMPUTEDPROFIT	CP_MATCHING_STATUS	Computed Profit: matching_status
BESTPROFIT	BP_MATCHING_STATUS	Best Profit: matching_status

Variable Importance

Obs	NAME	LABEL	NRULES	IMPORTANCE	VIMPORTANCE	RATIO
1	surname	surname	2	1.00000	1.00000	1.00000
2	hic	hic	5	0.38501	0.37626	0.97728
3	given_name	given_name	4	0.22589	0.18544	0.82092
4	data_of_birth	data_of_birth	1	0.06832	0.05301	0.77587

Tree Leaf Report

Node	Depth	Training Observations	Percent 1	Validation Observations	Percent V 1
15	3	4828	1.00	3953	1.00
16	4	4774	0.00	3927	0.00
11	3	255	1.00	218	0.99
13	3	151	0.96	128	0.94
29	4	147	0.97	123	0.97
12	3	133	0.04	95	0.13
19	4	64	0.91	49	0.82
18	4	64	0.20	42	0.07
42	5	30	0.23	16	0.00
20	4	17	0.12	9	0.33
21	4	13	1.00	11	0.73
43	5	12	1.00	16	1.00
17	4	10	1.00	5	1.00

--
* Score Output
--

--
* Report Output
--

Fit Statistics

Target=matching_status

Fit Statistics	Statistics Label	Train	Validation
NOBS	Sum of Frequencies	10498.00	8592.00
SUMW	Sum of Case Weights Times Freq	20996.00	17184.00
MISC	Misclassification Rate	0.01	0.01
MAX	Maximum Absolute Error	1.00	1.00
SSE	Sum of Squared Errors	126.51	104.40
ASE	Average Squared Error	0.01	0.01
RASE	Root Average Squared Error	0.08	0.08
DIV	Divisor for ASE	20996.00	17184.00
DFT	Total Degrees of Freedom	10498.00	.
APROF	Average Profit for matching_status	0.99	0.99
PROF	Total Profit for matching_status	10429.00	8538.00

Classification Table

Data Role=TRAIN Target Variable=matching_status

Target	Outcome	Target Percentage	Outcome Percentage	Frequency Count	Total Percentage

0	0	99.1232	99.4999	4974	47.3805
1	0	0.8768	0.8001	44	0.4191
0	1	0.4562	0.5001	25	0.2381
1	1	99.5438	99.1999	5455	51.9623

Data Role=VALIDATE Target Variable=matching_status

Target	Outcome	Target Percentage	Outcome Percentage	Frequency Count	Total Percentage
0	0	99.3641	99.3156	4063	47.2882
1	0	0.6359	0.5776	26	0.3026
0	1	0.6218	0.6844	28	0.3259
1	1	99.3782	99.4224	4475	52.0833

Event Classification Table

Data Role=TRAIN Target=matching_status

False Negative	True Negative	False Positive	True Positive
44	4974	25	5455

Data Role=VALIDATE Target=matching_status

False Negative	True Negative	False Positive	True Positive
26	4063	28	4475

Assessment Score Rankings

Data Role=TRAIN Target Variable=matching_status

Posterior Percentile	Gain	Lift	Cumulative Lift	% Response	Cumulative % Response	Observation Number	Probability Mean
0	.	.	.	.	.	.	.
5	90.6491	1.90649	1.90649	99.8647	99.8647	524.9	0.99865
10	90.6399	1.90631	1.90640	99.8550	99.8598	524.9	0.99855
15	90.6368	1.90631	1.90637	99.8550	99.8582	524.9	0.99855
20	90.6353	1.90631	1.90635	99.8550	99.8574	524.9	0.99855
25	90.6343	1.90631	1.90634	99.8550	99.8569	524.9	0.99855
30	90.6337	1.90631	1.90634	99.8550	99.8566	524.9	0.99855
35	90.6333	1.90631	1.90633	99.8550	99.8564	524.9	0.99855
40	90.6330	1.90631	1.90633	99.8550	99.8562	524.9	0.99855
45	90.6327	1.90631	1.90633	99.8550	99.8561	524.9	0.99855
50	90.4544	1.88850	1.90454	98.9222	99.7627	524.9	0.98922
55	81.2620	0.89338	1.81262	46.7962	94.9476	524.9	0.46796
60	66.2135	0.00680	1.66213	0.3561	87.0649	524.9	0.00356
65	53.4801	0.00680	1.53480	0.3561	80.3950	524.9	0.00356
70	42.5658	0.00680	1.42566	0.3561	74.6780	524.9	0.00356
75	33.1067	0.00680	1.33107	0.3561	69.7232	524.9	0.00356
80	24.8300	0.00680	1.24830	0.3561	65.3877	524.9	0.00356
85	17.5271	0.00680	1.17527	0.3561	61.5623	524.9	0.00356
90	11.0356	0.00680	1.11036	0.3561	58.1620	524.9	0.00356
95	5.2274	0.00680	1.05227	0.3561	55.1196	524.9	0.00356
100	0.0000	0.00680	1.00000	0.3561	52.3814	524.9	0.00356

Data Role=VALIDATE Target Variable=matching_status

Posterior Percentile	Gain	Lift	Cumulative Lift	% Response	Cumulative % Response	Observation Number	Probability Mean
0	.	.	.	.	.	.	.
5	89.4685	1.89468	1.89468	99.2549	99.2549	429.6	0.99866
10	90.1314	1.90794	1.90131	99.9494	99.6021	429.6	0.99855
15	90.3524	1.90794	1.90352	99.9494	99.7179	429.6	0.99855
20	90.4629	1.90794	1.90463	99.9494	99.7758	429.6	0.99855
25	90.5292	1.90794	1.90529	99.9494	99.8105	429.6	0.99855
30	90.5734	1.90794	1.90573	99.9494	99.8336	429.6	0.99855
35	90.6049	1.90794	1.90605	99.9494	99.8502	429.6	0.99855
40	90.6286	1.90794	1.90629	99.9494	99.8626	429.6	0.99855
45	90.6470	1.90794	1.90647	99.9494	99.8722	429.6	0.99855
50	90.4455	1.88632	1.90445	98.8165	99.7667	429.6	0.99025
55	81.5000	0.92045	1.81500	48.2188	95.0805	429.6	0.49677
60	66.4074	0.00389	1.66407	0.2037	87.1741	429.6	0.00356
65	53.6368	0.00389	1.53637	0.2037	80.4841	429.6	0.00356
70	42.6905	0.00389	1.42690	0.2037	74.7498	429.6	0.00356
75	33.2037	0.00389	1.33204	0.2037	69.7800	429.6	0.00356
80	24.9028	0.00389	1.24903	0.2037	65.4315	429.6	0.00356
85	17.5784	0.00389	1.17578	0.2037	61.5946	429.6	0.00356
90	11.0679	0.00389	1.11068	0.2037	58.1840	429.6	0.00356
95	5.2427	0.00389	1.05243	0.2037	55.1324	429.6	0.00356
100	0.0000	0.00389	1.00000	0.2037	52.3859	429.6	0.00356

Assessment Score Distribution

Data Role=TRAIN Target Variable=matching_status

Posterior Probability Range	Number of Events	Number of Nonevents	Posterior Probability Mean	Percentage
0.95 - 1.00	5397	19	0.99649	51.5908
0.90 - 0.95	58	6	0.90625	0.6096
0.85 - 0.90	0	0	.	0.0000
0.80 - 0.85	0	0	.	0.0000
0.75 - 0.80	0	0	.	0.0000
0.70 - 0.75	0	0	.	0.0000
0.65 - 0.70	0	0	.	0.0000
0.60 - 0.65	0	0	.	0.0000
0.55 - 0.60	0	0	.	0.0000
0.50 - 0.55	0	0	.	0.0000
0.45 - 0.50	0	0	.	0.0000
0.40 - 0.45	0	0	.	0.0000
0.35 - 0.40	0	0	.	0.0000
0.30 - 0.35	0	0	.	0.0000
0.25 - 0.30	0	0	.	0.0000
0.20 - 0.25	20	74	0.21277	0.8954
0.15 - 0.20	0	0	.	0.0000
0.10 - 0.15	2	15	0.11765	0.1619
0.05 - 0.10	0	0	.	0.0000
0.00 - 0.05	22	4885	0.00448	46.7422

Data Role=VALIDATE Target Variable=matching_status

Posterior Probability Range	Number of Events	Number of Nonevents	Posterior Probability Mean	Percentage
0.95 - 1.00	4435	19	0.99644	51.8389
0.90 - 0.95	40	9	0.90625	0.5703
0.85 - 0.90	0	0	.	0.0000
0.80 - 0.85	0	0	.	0.0000
0.75 - 0.80	0	0	.	0.0000
0.70 - 0.75	0	0	.	0.0000
0.65 - 0.70	0	0	.	0.0000
0.60 - 0.65	0	0	.	0.0000
0.55 - 0.60	0	0	.	0.0000
0.50 - 0.55	0	0	.	0.0000
0.45 - 0.50	0	0	.	0.0000
0.40 - 0.45	0	0	.	0.0000
0.35 - 0.40	0	0	.	0.0000
0.30 - 0.35	0	0	.	0.0000
0.25 - 0.30	0	0	.	0.0000
0.20 - 0.25	3	55	0.21146	0.6750
0.15 - 0.20	0	0	.	0.0000
0.10 - 0.15	3	6	0.11765	0.1047
0.05 - 0.10	0	0	.	0.0000
0.00 - 0.05	20	4002	0.00436	46.8110

Appendix 26 Decision Model (Levenshtein) Output of the Modified Data Set Generated by SAS Enterprise Miner

```
*------------------------------------------------------------*
User:                 dwang
Date:                 30JAN08
Time:                 11:51
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*



Variable Summary

          Measurement    Frequency
 Role        Level         Count

INPUT      INTERVAL          5
TARGET     BINARY            1



Model Events

                                        Number
                           Measurement     of
    Target         Event       Level     Levels      Order            Label

matching_status      1        BINARY        2      Descending    matching_status



Decision matrix

matching_      Training
 status      Proportions    1    0

    1          0.52383      1    0
    0          0.47617      0    1



Predicted and decision variables

Type              Variable             Label

TARGET            matching_status      matching_status
PREDICTED         P_matching_status1   Predicted: matching_status=1
RESIDUAL          R_matching_status1   Residual: matching_status=1
PREDICTED         P_matching_status0   Predicted: matching_status=0
RESIDUAL          R_matching_status0   Residual: matching_status=0
FROM              F_matching_status    From: matching_status
INTO              I_matching_status    Into: matching_status
MODELDECISION     D_Dup1               Decision: matching_status
EXPECTEDPROFIT    EP_MATCHING_STATUS   Expected Profit: matching_status
COMPUTEDPROFIT    CP_MATCHING_STATUS   Computed Profit: matching_status
BESTPROFIT        BP_MATCHING_STATUS   Best Profit: matching_status



Variable Importance

Obs    NAME            LABEL    NRULES    IMPORTANCE    VIMPORTANCE     RATIO
```

1	hic	hic	1	1.00000	1.00000	1.00000
2	given_name		1	0.16723	0.15224	0.91032
3	surname		1	0.12105	0.14434	1.19242
4	date_of_birth		1	0.07480	0.06245	0.83484

Tree Leaf Report

Node	Depth	Training Observations	Percent 1	Validation Observations	Percent V 1
9	4	5547	0.99	4556	0.99
2	1	4828	0.00	3936	0.00
4	2	72	0.00	49	0.00
6	3	37	0.00	43	0.00
8	4	14	0.00	8	0.00

--
* Score Output
--

--
* Report Output
--

Fit Statistics

Target=matching_status

Fit Statistics	Statistics Label	Train	Validation
NOBS	Sum of Frequencies	10498.00	8592.00
SUMW	Sum of Case Weights Times Freq	20996.00	17184.00
MISC	Misclassification Rate	0.00	0.01
MAX	Maximum Absolute Error	0.99	0.99
SSE	Sum of Squared Errors	95.17	108.78
ASE	Average Squared Error	0.00	0.01
RASE	Root Average Squared Error	0.07	0.08
DIV	Divisor for ASE	20996.00	17184.00
DFT	Total Degrees of Freedom	10498.00	.
APROF	Average Profit for matching_status	1.00	0.99
PROF	Total Profit for matching_status	10450.00	8537.00

Classification Table

Data Role=TRAIN Target Variable=matching_status

Target	Outcome	Target Percentage	Outcome Percentage	Frequency Count	Total Percentage
0	0	100.000	99.040	4951	47.1614
0	1	0.865	0.960	48	0.4572
1	1	99.135	100.000	5499	52.3814

Data Role=VALIDATE Target Variable=matching_status

Target	Outcome	Frequency	Total

Target	Outcome	Percentage	Percentage	Count	Percentage
0	0	100.000	98.656	4036	46.9739
0	1	1.207	1.344	55	0.6401
1	1	98.793	100.000	4501	52.3859

Event Classification Table

Data Role=TRAIN Target=matching_status

False Negative	True Negative	False Positive	True Positive
0	4951	48	5499

Data Role=VALIDATE Target=matching_status

False Negative	True Negative	False Positive	True Positive
0	4036	55	4501

Assessment Score Rankings

Data Role=TRAIN Target Variable=matching_status

Posterior Percentile	Gain	Lift	Cumulative Lift	% Response	Cumulative % Response	Observation Number	Probability Mean
0	.	.	.	.	.	.	.
5	89.2555	1.89255	1.89255	99.1347	99.1347	524.9	0.99135
10	89.2555	1.89255	1.89255	99.1347	99.1347	524.9	0.99135
15	89.2555	1.89255	1.89255	99.1347	99.1347	524.9	0.99135
20	89.2555	1.89255	1.89255	99.1347	99.1347	524.9	0.99135
25	89.2555	1.89255	1.89255	99.1347	99.1347	524.9	0.99135
30	89.2555	1.89255	1.89255	99.1347	99.1347	524.9	0.99135
35	89.2555	1.89255	1.89255	99.1347	99.1347	524.9	0.99135
40	89.2555	1.89255	1.89255	99.1347	99.1347	524.9	0.99135
45	89.2555	1.89255	1.89255	99.1347	99.1347	524.9	0.99135
50	89.2555	1.89255	1.89255	99.1347	99.1347	524.9	0.99135
55	81.8182	1.07445	1.81818	56.2814	95.2389	524.9	0.56281
60	66.6667	0.00000	1.66667	0.0000	87.3023	524.9	0.00000
65	53.8462	0.00000	1.53846	0.0000	80.5868	524.9	0.00000
70	42.8571	0.00000	1.42857	0.0000	74.8306	524.9	0.00000
75	33.3333	0.00000	1.33333	0.0000	69.8419	524.9	0.00000
80	25.0000	0.00000	1.25000	0.0000	65.4768	524.9	0.00000
85	17.6471	0.00000	1.17647	0.0000	61.6252	524.9	0.00000
90	11.1111	0.00000	1.11111	0.0000	58.2016	524.9	0.00000
95	5.2632	0.00000	1.05263	0.0000	55.1383	524.9	0.00000
100	0.0000	0.00000	1.00000	0.0000	52.3814	524.9	0.00000

Data Role=VALIDATE Target Variable=matching_status

Posterior Percentile	Gain	Lift	Cumulative Lift	% Response	Cumulative % Response	Observation Number	Probability Mean
0	.	.	.	.	.	.	.
5	88.5865	1.88586	1.88586	98.7928	98.7928	429.6	0.99135
10	88.5865	1.88586	1.88586	98.7928	98.7928	429.6	0.99135
15	88.5865	1.88586	1.88586	98.7928	98.7928	429.6	0.99135
20	88.5865	1.88586	1.88586	98.7928	98.7928	429.6	0.99135
25	88.5865	1.88586	1.88586	98.7928	98.7928	429.6	0.99135

30	88.5865	1.88586	1.88586	98.7928	98.7928	429.6	0.99135
35	88.5865	1.88586	1.88586	98.7928	98.7928	429.6	0.99135
40	88.5865	1.88586	1.88586	98.7928	98.7928	429.6	0.99135
45	88.5865	1.88586	1.88586	98.7928	98.7928	429.6	0.99135
50	88.5865	1.88586	1.88586	98.7928	98.7928	429.6	0.99135
55	81.8182	1.14135	1.81818	59.7908	95.2472	429.6	0.59998
60	66.6667	0.00000	1.66667	0.0000	87.3099	429.6	0.00000
65	53.8462	0.00000	1.53846	0.0000	80.5938	429.6	0.00000
70	42.8571	0.00000	1.42857	0.0000	74.8371	429.6	0.00000
75	33.3333	0.00000	1.33333	0.0000	69.8479	429.6	0.00000
80	25.0000	0.00000	1.25000	0.0000	65.4824	429.6	0.00000
85	17.6471	0.00000	1.17647	0.0000	61.6305	429.6	0.00000
90	11.1111	0.00000	1.11111	0.0000	58.2066	429.6	0.00000
95	5.2632	0.00000	1.05263	0.0000	55.1431	429.6	0.00000
100	0.0000	0.00000	1.00000	0.0000	52.3859	429.6	0.00000

Assessment Score Distribution

Data Role=TRAIN Target Variable=matching_status

Posterior Probability Range	Number of Events	Number of Nonevents	Posterior Probability Mean	Percentage
0.95 - 1.00	5499	48	0.99135	52.8386
0.90 - 0.95	0	0	.	0.0000
0.85 - 0.90	0	0	.	0.0000
0.80 - 0.85	0	0	.	0.0000
0.75 - 0.80	0	0	.	0.0000
0.70 - 0.75	0	0	.	0.0000
0.65 - 0.70	0	0	.	0.0000
0.60 - 0.65	0	0	.	0.0000
0.55 - 0.60	0	0	.	0.0000
0.50 - 0.55	0	0	.	0.0000
0.45 - 0.50	0	0	.	0.0000
0.40 - 0.45	0	0	.	0.0000
0.35 - 0.40	0	0	.	0.0000
0.30 - 0.35	0	0	.	0.0000
0.25 - 0.30	0	0	.	0.0000
0.20 - 0.25	0	0	.	0.0000
0.15 - 0.20	0	0	.	0.0000
0.10 - 0.15	0	0	.	0.0000
0.05 - 0.10	0	0	.	0.0000
0.00 - 0.05	0	4951	0.00000	47.1614

Data Role=VALIDATE Target Variable=matching_status

Posterior Probability Range	Number of Events	Number of Nonevents	Posterior Probability Mean	Percentage
0.95 - 1.00	4501	55	0.99135	53.0261
0.90 - 0.95	0	0	.	0.0000
0.85 - 0.90	0	0	.	0.0000
0.80 - 0.85	0	0	.	0.0000
0.75 - 0.80	0	0	.	0.0000
0.70 - 0.75	0	0	.	0.0000
0.65 - 0.70	0	0	.	0.0000
0.60 - 0.65	0	0	.	0.0000
0.55 - 0.60	0	0	.	0.0000
0.50 - 0.55	0	0	.	0.0000
0.45 - 0.50	0	0	.	0.0000
0.40 - 0.45	0	0	.	0.0000
0.35 - 0.40	0	0	.	0.0000
0.30 - 0.35	0	0	.	0.0000
0.25 - 0.30	0	0	.	0.0000
0.20 - 0.25	0	0	.	0.0000
0.15 - 0.20	0	0	.	0.0000

0.10 - 0.15	0	0	.	0.0000
0.05 - 0.10	0	0	.	0.0000
0.00 - 0.05	0	4036	0.00000	46.9739

Appendix 27 Model Comparison Report of the First Data Set Generated by SAS Enterprise Miner

```
*-------------------------------------------------------------*
User:                dwang
Date:                18JAN08
Time:                13:43
*-------------------------------------------------------------*
* Training Output
*-------------------------------------------------------------*
```

Variable Summary

Role	Measurement Level	Frequency Count
TARGET	BINARY	1

Fit Statistics
Model selection based on _VAPROF_

Selected Model	Model Node	Valid: Average Profit for matching_status	Train: Average Squared Error	Train: Misclassification Rate	Valid: Average Squared Error	Valid: Misclassification Rate
Y	Tree3	1.00000	.000095239	.000095256	.000000017	0
	Tree2	0.99791	.001801987	.001809869	.002089286	.002094972
	Tree	.	.001491824	.001524100	.001246860	.001280261

Fit Statistics Table
Target: matching_status

Data Role=Train

Statistics	Tree3	Tree2	Tree
Train: Kolmogorov-Smirnov Statistic	1.00	1.00	1.00
Train: Average Profit for matching_status	1.00	1.00	.
Train: Average Squared Error	0.00	0.00	0.00
Train: Roc Index	1.00	1.00	1.00
Train: Percent Capture Response	9.54	9.53	9.53

Train: Total Degrees of Freedom	10498.00	10498.00	10498.00
Train: Divisor for ASE	20996.00	20996.00	20996.00
Train: Gain	90.87	90.60	90.70
Train: Gini Coefficient	1.00	1.00	1.00
Train: Bin-Based Two-Way Kolmogorov-Smirnov Statistic	0.95	0.95	0.95
Train: Lift	1.91	1.91	1.91
Train: Maximum Absolute Error	1.00	1.00	1.00
Train: Misclassification Rate	0.00	0.00	0.00
Train: Sum of Frequencies	10498.00	10498.00	10498.00
Train: Total Profit for matching_status	10497.00	10479.00	.
Train: Root Average Squared Error	0.01	0.04	0.04
Train: Percent Response	99.98	9.83	9.89
Train: Sum of Squared Errors	2.00	37.83	31.32
Train: Sum of Case Weights Times Freq	20996.00	20996.00	20996.00

Data Role=Valid

Statistics	Tree3	Tree2	Tree
Valid: Kolmogorov-Smirnov Statistic	1.00	1.00	1.00
Valid: Average Profit for matching_status	1.00	1.00	.
Valid: Average Squared Error	0.00	0.00	0.00
Valid: Roc Index	1.00	1.00	1.00
Valid: Percent Capture Response	9.54	9.54	9.54
Valid: Divisor for VASE	17184.00	17184.00	17184.00
Valid: Gain	90.89	89.84	90.81
Valid: Gini Coefficient	1.00	1.00	1.00
Valid: Bin-Based Two-Way Kolmogorov-Smirnov Statistic	0.95	0.95	0.95
Valid: Lift	1.91	1.91	1.91
Valid: Maximum Absolute Error	0.00	1.00	1.00
Valid: Misclassification Rate	0.00	0.00	0.00
Valid: Sum of Frequencies	8592.00	8592.00	8592.00
Valid: Total Profit for matching_status	8592.00	8574.00	.
Valid: Root Average Squared Error	0.00	0.05	0.04
Valid: Percent Response	100.00	99.91	99.95
Valid: Sum of Squared Errors	0.00	35.90	21.43
Valid: Sum of Case Weights Times Freq	17184.00	17184.00	17184.00

Event Classification Table
Model selection based on _VAPROF_

MODEL	MODELDESCRIPTION	Data Role	Target	False Negative	True Negative	False Positive	True Positive
Tree	Decision Tree	TRAIN	matching_status	7	4990	9	5492
Tree	Decision Tree	VALIDATE	matching_status	6	4086	5	4495
Tree2	Decision Tree(2)	TRAIN	matching_status	7	4987	12	5492
Tree2	Decision Tree(2)	VALIDATE	matching_status	8	4081	10	4493
Tree3	Decision Tree 3)	TRAIN	matching_status	0	4998	1	5499
Tree3	Decision Tree(3)	VALIDATE	matching_status	0	4091	.	4501

--
* Score Output
--

--
* Report Output
--

Appendix 28 Statistic Report of the First Data Set Using The Jaro String Comparison Function

```
*------------------------------------------------------------*
User:                 dwang
Date:                 01FEB08
Time:                 09:34
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*
```

Variable Summary

Role	Measurement Level	Frequency Count
INPUT	INTERVAL	5
TARGET	BINARY	1

Variable Levels Summary
(maximum 500 observations printed)

Variable	Role	Number of Levels
matching_status	TARGET	2

Class Variable Summary Statistics
(maximum 500 observations printed)

Variable	Role	Number of Levels	Missing	Mode	Mode Percentage	Mode2	Mode2 Percentage
matching_status	TARGET	2	0	1	52.38	0	47.62

Distribution of Class Target and Segment Variables
(maximum 500 observations printed)

Variable	Role	Formatted Value	Frequency Count	Percent
matching_status	TARGET	1	10000	52.3834
matching_status	TARGET	0	9090	47.6166

Interval Variable Summary Statistics
(maximum 500 observations printed)

Variable	ROLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
date_of_birth	INPUT	0.66092	0.35052	19090	0	0	0.78	1
gender	INPUT	0.87552	0.12940	19090	0	0	0.89	1
given_name	INPUT	0.61958	0.34816	19090	0	0	0.70	1
hic	INPUT	0.67934	0.30441	19090	0	0	0.74	1
surname	INPUT	0.61339	0.35394	19090	0	0	0.70	1

Interval Variable Summary Statistics by Class Target
(maximum 500 observations printed)

Variable=date_of_birth

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		date_of_birth	0.66092	0.35052	9090	0	0	0.78	1.00
matching_status	0	date_of_birth	0.48528	0.31853	9090	0	0	0.67	0.92
matching_status	1	date_of_birth	0.82058	0.29798	10000	0	0	0.92	1.00

Variable=gender

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		gender	0.87552	0.12940	19090	0	0.00	0.89	1
matching_status	0	gender	0.86000	0.14001	9090	0	0.72	0.72	1
matching_status	1	gender	0.88964	0.11717	10000	0	0.00	0.89	1

Variable=given_name

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		given_name	0.61958	0.34816	19090	0	0	0.70	1
matching_status	0	given_name	0.33691	0.26112	9090	0	0	0.45	1
matching_status	1	given_name	0.87653	0.17540	10000	0	0	0.89	1

Variable=hic

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		hic	0.67934	0.30441	19090	0	0	0.74	1.00
matching_status	0	hic	0.42150	0.23388	9090	0	0	0.51	0.85
matching_status	1	hic	0.91373	0.10867	10000	0	0	0.90	1.00

Variable=surname

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		surname	0.61339	0.35394	19090	0	0	0.70	1
matching_status	0	surname	0.31560	0.24975	9090	0	0	0.44	1
matching_status	1	surname	0.88408	0.16899	10000	0	0	0.90	1

Chi-Square Statistics
(maximum 500 observations printed)

Target=matching_status

Input	Chi-Square	Df	Prob
hic	16794.5990	4	<.0001
surname	16302.0243	4	<.0001
given_name	15201.8080	4	<.0001
date_of_birth	12102.9831	3	<.0001

```
gender            2245.7141     3    <.0001


*-------------------------------------------------------------*
* Score Output
*-------------------------------------------------------------*


*-------------------------------------------------------------*
* Report Output
*-------------------------------------------------------------*
```

Appendix 29 Statistic Report of the First Data Set Using the Jaro-Winkler Method

```
*-------------------------------------------------------------*
User:                dwang
Date:                18JAN08
Time:                13:26
*-------------------------------------------------------------*
* Training Output
*-------------------------------------------------------------*



Variable Summary

          Measurement    Frequency
 Role        Level         Count

INPUT      INTERVAL          5
TARGET     BINARY            1


Variable Levels Summary
(maximum 500 observations printed)

                            Number
                              of
   Variable         Role     Levels

matching_status    TARGET       2


Class Variable Summary Statistics
(maximum 500 observations printed)

                            Number
                              of                          Mode              Mode2
   Variable         Role     Levels   Missing   Mode   Percentage   Mode  Percentage

matching_status    TARGET       2        0       1       52.38       0      47.62


Distribution of Class Target and Segment Variables
(maximum 500 observations printed)

                            Formatted    Frequency
   Variable         Role       Value        Count      Percent

matching_status    TARGET        1          10000      52.3834
matching_status    TARGET        0           9090      47.6166


Interval Variable Summary Statistics
(maximum 500 observations printed)

                                  Std.      Non
Variable       ROLE     Mean   Deviation  Missing Missing Minimum  Median  Maximum

date_of_birth  INPUT  0.69730  0.34735    19090     0       0     0.82      1
gender         INPUT  0.88318  0.12669    19090     0       0     0.92      1
given_name     INPUT  0.62932  0.35254    19090     0       0     0.73      1
hic            INPUT  0.69073  0.30729    19090     0       0     0.79      1
surname        INPUT  0.62323  0.35882    19090     0       0     0.73      1
```

```
Interval Variable Summary Statistics by Class Target
(maximum 500 observations printed)

Variable=date_of_birth

                    LABEL OF
               Target  FORMER            Std.      Non
Target         Level  VARIABLE   Mean  Deviation Missing Missing Minimum Median Maximum

_OVERALL_          date_of_birth 0.69730 0.34735   19090    0       0     0.82   1.00
matching_status 0 date_of_birth 0.54485 0.33116    9090    0       0     0.73   0.95
matching_status 1 date_of_birth 0.83587 0.30053   10000    0       0     0.95   1.00
Variable=gende

                    LABEL OF
               Target  FORMER           Std.      Non
Target         Level  VARIABLE  Mean  Deviation Missing  Missing Minimum Median Maximum

_OVERALL_              gender  0.88318 0.12669   19090     0      0.00    0.92    1
matching_status  0     gender  0.86000 0.14001    9090     0      0.72    0.72    1
matching_status  1     gender  0.90426 0.10904   10000     0      0.00    0.92    1


Variable=given_name

                    LABEL OF
               Target  FORMER              Std.    Non
Target         Level  VARIABLE   Mean  Deviation Missing Missing Minimum Median Maximum
_OVERALL_             given_name 0.62932 0.35254  19090     0        0     0.73    1
matching_status  0    given_name 0.34062 0.26323   9090     0        0     0.46    1
matching_status  1    given_name 0.89175 0.17219  10000     0        0     0.93    1


Variable=hic

                    LABEL OF
               Target  ORMER            Std.      Non
Target         Level  VARIABLE  Mean  Deviation  Missing Missing Minimum Median Maximum
_OVERALL_              hic     0.69073  0.30729   19090     0       0     0.79   1.00
matching_status  0     hic     0.42716  0.23489    9090     0       0     0.51   0.91
matching_status  1     hic     0.93032  0.09774   10000     0       0     0.94   1.00


Variable=surname

                    LABEL OF
               Target  FORMER            Std.      Non
Target         Level  VARIABLE  Mean  Deviation  Missing Missing Minimum Median Maximum

_OVERALL_             surname  0.62323  0.35882   19090     0       0      0.73    1
matching_status  0    surname  0.31887  0.25176   9090      0       0      0.44    1
matching_status  1    surname  0.89990  0.16559   10000     0       0      0.93    1


*------------------------------------------------------------*
* Score Output
*------------------------------------------------------------*


*------------------------------------------------------------*
* Report Output
*------------------------------------------------------------*
```

Appendix 30 Statistic Report of the First Data Set Using the Levenshtein Edit Distance Metric

```
*------------------------------------------------------------*
User:                dwang
Date:                18JAN08
Time:                13:27
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*
```

Variable Summary

Role	Measurement Level	Frequency Count
INPUT	INTERVAL	5
TARGET	BINARY	1

Variable Levels Summary
(maximum 500 observations printed)

Variable	Role	Number of Levels
matching_status	TARGET	2

Class Variable Summary Statistics
(maximum 500 observations printed)

Variable	Role	Number of Levels	Missing	Mode	Mode Percentage	Mode2	Mode2 Percentage
matching_status	TARGET	2	0	1	52.38	0	47.62

Distribution of Class Target and Segment Variables
(maximum 500 observations printed)

Variable	Role	Formatted Value	Frequency Count	Percent
matching_status	TARGET	1	10000	52.3834
matching_status	TARGET	0	9090	47.6166

Interval Variable Summary Statistics
(maximum 500 observations printed)

Variable	ROLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
data_of_birth	INPUT	0.59465	0.32402	19090	0	0.0	0.62	1
gender	INPUT	0.83651	0.16263	19090	0	0.5	0.83	1
given_name	INPUT	0.50601	0.38496	19090	0	0.0	0.50	1
hic	INPUT	0.52670	0.39827	19090	0	0.0	0.71	1
surname	INPUT	0.50145	0.39389	19090	0	0.0	0.57	1

Interval Variable Summary Statistics by Class Target
(maximum 500 observations printed)

Variable=date_of_birth

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		date_of_birth	0.59465	0.32402	19090	0	0.0	0.62	1.00
matching_status	0	date_of_birth	0.33103	0.19950	9090	0	0.0	0.38	0.88
matching_status	1	date_of_birth	0.83429	0.20892	10000	0	0.3	0.88	1.00

Variable=gender

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		gender	0.83651	0.16263	19090	0	0.50	0.83	1
matching_status	0	gender	0.83500	0.16501	9090	0	0.67	0.67	1
matching_status	1	gender	0.83788	0.16042	10000	0	0.50	0.83	1

Variable=given_name

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		given_name	0.50601	0.38496	19090	0	0.0	0.50	1
matching_status	0	given_name	0.13717	0.14253	9090	0	0.0	0.14	1
matching_status	1	given_name	0.84129	0.16840	10000	0	0.3	0.86	1

Variable=hic

Target	Targe Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		hic	0.52670	0.39827	19090	0	0.00	0.71	1.00
matching_status	0	hic	0.12673	0.11807	9090	0	0.00	0.14	0.71
matching_status	1	hic	0.89027	0.11185	10000	0	0.71	0.86	1.00

Variable=surname

Targe	Targe Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		surname	0.50145	0.39389	19090	0	0	0.57	1
matching_status	0	surname	0.11591	0.11807	9090	0	0	0.12	1
matching_status	1	surname	0.85191	0.15988	10000	0	0	0.86	1

--
* Score Output
--

--
* Report Output
--

Appendix 31 Model Comparison Report of the Modified Data Set Generated by SAS Enterprise Miner

```
*------------------------------------------------------------*
User:                 dwang
Date:                 30JAN08
Time:                 11:51
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*
```

Variable Summary

Role	Measurement Level	Frequency Count
TARGET	BINARY	1

Fit Statistics
Model selection based on _VAPROF_

Selected Model	Model Node	Valid: Average Profit for matching_status	Train: Average Squared Error	Train: Misclassification Rate	Valid: Average Squared Error	Valid: Misclassification Rate
Y	Tree	0.99418	.004084271	.004572299	.005395735	.005819367
	Tree2	0.99372	.006025642	.006572681	.006075262	.006284916
	Tree3	0.99360	.004532734	.004572299	.006330224	.006401304

Fit Statistics Table
Target: matching_status

Data Role=Train

Statistics	Tree	Tree2	Tree3
Train: Kolmogorov-Smirnov Statistic	0.99	0.99	0.99
Train: Average Profit for matching_status	1.00	0.99	1.00
Train: Average Squared Error	0.00	0.01	0.00
Train: Roc Index	1.00	1.00	1.00
Train: Percent Capture Response	9.54	9.53	9.46
Train: Total Degrees of Freedom	10498.00	10498.00	10498.00
Train: Divisor for ASE	20996.00	20996.00	20996.00
Train: Gain	90.79	90.64	89.26
Train: Gini Coefficient	1.00	0.99	0.99
Train: Bin-Based Two-Way Kolmogorov-Smirnov Statistic	0.95	0.95	0.95

Train: Lift	1.91	1.91	1.89
Train: Maximum Absolute Error	1.00	1.00	0.99
Train: Misclassification Rate	0.00	0.01	0.00
Train: Sum of Frequencies	10498.00	10498.00	10498.00
Train: Total Profit for matching_status	10450.00	10429.00	10450.00
Train: Root Average Squared Error	0.06	0.08	0.07
Train: Percent Response	99.94	99.86	99.13
Train: Sum of Squared Errors	85.75	126.51	95.17
Train: Sum of Case Weights Times Freq	20996.00	20996.00	20996.00

Data Role=Valid

Statistics	Tree	Tree2	Tree3
Valid: Kolmogorov-Smirnov Statistic	0.99	0.99	0.99
Valid: Average Profit for matching_status	0.99	0.99	0.99
Valid: Average Squared Error	0.01	0.01	0.01
Valid: Roc Index	1.00	1.00	0.99
Valid: Percent Capture Response	9.54	9.54	9.43
Valid: Divisor for VASE	17184.00	17184.00	17184.00
Valid: Gain	90.80	90.13	88.59
Valid: Gini Coefficient	1.00	1.00	0.99
Valid: Bin-Based Two-Way Kolmogorov-Smirnov Statistic	0.95	0.95	0.95
Valid: Lift	1.91	1.91	1.89
Valid: Maximum Absolute Error	1.00	1.00	0.99
Valid: Misclassification Rate	0.01	0.01	0.01
Valid: Sum of Frequencies	8592.00	8592.00	8592.00
Valid: Total Profit for matching_status	8542.00	8538.00	8537.00
Valid: Root Average Squared Error	0.07	0.08	0.08
Valid: Percent Response	99.95	99.95	98.79
Valid: Sum of Squared Errors	92.72	104.40	108.78
Valid: Sum of Case Weights Times Freq	17184.00	17184.00	17184.00

Event Classification Table
Model selection based on _VAPROF_

MODEL	MODELDESCRIPTION	Data Role	Target	False Negative	True Negative	False Positive	True Positive
Tree	Decision Model(Jaro)	TRAIN	matching_status	31	4982	17	5468
Tree	Decision Model(Jaro)	VALIDATE	matching_status	23	4064	27	4478
Tree2	Decision Model(JK)	TRAIN	matching_status	44	4974	25	5455
Tree2	Decision Model(JK)	VALIDATE	matching_status	26	4063	28	4475
Tree3	Decision Model(LED)	TRAIN	matching_status	0	4951	48	5499
Tree3	Decision Model(LED)	VALIDATE	matching_status	0	036	55	4501

--
* Score Output
--

--
* Report Output
--

Appendix 32 Statistic Report of the Modified Data Set Using The Jaro String Comparison Function

```
*------------------------------------------------------------*
User:                 dwang
Date:                 20JAN08
Time:                 15:28
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*
```

Variable Summary

Role	Measurement Level	Frequency Count
INPUT	INTERVAL	5
TARGET	BINARY	1

Variable Levels Summary
(maximum 500 observations printed)

Variable	Role	Number of Levels
matching_status	TARGET	2

Class Variable Summary Statistics
(maximum 500 observations printed)

Variable	Role	Number of Levels	Missing	Mode	Mode percentage	Mode2	Mode2 Percentage
matching_status	TARGET	2	0	1	52.38	0	47.62

Distribution of Class Target and Segment Variables
(maximum 500 observations printed)

Variable	Role	Formatted Value	Frequency Count	Percent
matching_status	TARGET	1	10000	52.3834
matching_status	TARGET	0	9090	47.6166

Interval Variable Summary Statistics
(maximum 500 observations printed)

Variable	ROLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
data_of_birth	INPUT	0.63434	0.33934	19090	0	0	0.75	1
gender	INPUT	0.85641	0.13324	19090	0	0	0.83	1
given_name	INPUT	0.59393	0.33388	19090	0	0	0.67	1
hic	INPUT	0.64850	0.29536	19090	0	0	0.71	1
surname	INPUT	0.58622	0.33929	19090	0	0	0.66	1

Interval Variable Summary Statistics by Class Target
(maximum 500 observations printed)

Variable=date_of_birth

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		date_of_birth	0.63434	0.33934	19090	0	0	0.75	1.00
matching_status	0	date_of_birth	0.48528	0.31853	9090	0	0	0.67	0.92
matching_status	1	date_of_birth	0.76983	0.29841	10000	0	0	0.83	1.00

Variable=gender

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		gender	0.85641	0.13324	19090	0	0.00	0.83	1
matching_status	0	gender	0.86000	0.14001	9090	0	0.72	0.72	1
matching_status	1	gender	0.85315	0.12670	10000	0	0.00	0.83	1

Variable=given_name

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		given_name	0.59393	0.33388	19090	0	0	0.67	1
matching_status	0	given_name	0.33691	0.26112	9090	0	0	0.45	1
matching_status	1	given_name	0.82755	0.19027	10000	0	0	0.87	1

Variable=hic

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		hic	0.64850	0.29536	19090	0	0	0.71	1.00
matching_status	0	hic	0.42150	0.23388	9090	0	0	0.51	0.85
matching_status	1	hic	0.85485	0.16551	10000	0	0	0.90	1.00

Variable=surname

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		surname	0.58622	0.33929	19090	0	0	0.66	1
matching_status	0	surname	0.31560	0.24975	9090	0	0	0.44	1
matching_status	1	surname	0.83221	0.18966	10000	0	0	0.87	1

--
* Score Output
--

--
* Report Output
--

Appendix 33 Statistic Report of the Modified Data Set Using the Jaro-Winkler Method

```
*------------------------------------------------------------*
User:                 dwang
Date:                 20JAN08
Time:                 15:32
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*
```

Variable Summary

Role	Measurement Level	Frequency Count
INPUT	INTERVAL	5
TARGET	BINARY	1

Variable Levels Summary
(maximum 500 observations printed)

Variable	Role	Number of Levels
matching_status	TARGET	2

Class Variable Summary Statistics
(maximum 500 observations printed)

Variable	Role	Number of Levels	Missing	Mode	Mode Percentage	Mode2	Mode2 Percentage
matching_status	TARGET	2	0	1	52.38	0	47.62

Distribution of Class Target and Segment Variables
(maximum 500 observations printed)

Variable	Role	Formatted Value	Frequency Count	Percent
matching_status	TARGET	1	10000	52.3834
matching_status	TARGET	0	9090	47.6166

Interval Variable Summary Statistics
(maximum 500 observations printed)

Variable	ROLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
data_of_birth	INPUT	0.67297	0.33876	19090	0	0	0.78	1
gender	INPUT	0.86540	0.13028	19090	0	0	0.87	1
given_name	INPUT	0.60527	0.33926	19090	0	0	0.67	1
hic	INPUT	0.66142	0.29795	19090	0	0	0.71	1
surname	INPUT	0.59787	0.34499	19090	0	0	0.67	1

Interval Variable Summary Statistics by Class Target
(maximum 500 observations printed)

Variable=date_of_birth

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		data_of_birth	0.67297	0.33876	19090	0	0	0.78	1.00
matching_status	0	data_of_birth	0.54485	0.33116	9090	0	0	0.73	0.95
matching_status	1	data_of_birth	0.78944	0.30151	10000	0	0	0.90	1.00

Variable=gender

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		gender	0.86540	0.13028	19090	0	0.00	0.87	1
matching_status	0	gender	0.86000	0.14001	9090	0	0.72	0.72	1
matching_status	1	gender	0.87032	0.12056	10000	0	0.00	0.87	1

Variable=given_name

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		given_name	0.60527	0.33926	19090	0	0	0.67	1
matching_status	0	given_name	0.34062	0.26323	9090	0	0	0.46	1
matching_status	1	given_name	0.84583	0.18763	10000	0	0	0.88	1

Variable=hic

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		hic	0.66142	0.29795	19090	0	0	0.71	1.00
matching_status	0	hic	0.42716	0.23489	9090	0	0	0.51	0.91
matching_status	1	hic	0.87436	0.15521	10000	0	0	0.91	1.00

Variable=surname

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		surname	0.59787	0.34499	19090	0	0	0.67	1
matching_status	0	surname	0.31887	0.25176	9090	0	0	0.44	1
matching_status	1	surname	0.85148	0.18577	10000	0	0	0.89	1

--
* Score Output
--

--
* Report Output
--

Appendix 34 Statistic Report of the Modified Data Set Using the Levenshtein Edit Distance Metric

```
*------------------------------------------------------------*
User:                dwang
Date:                20JAN08
Time:                15:39
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*
```

Variable Summary

Role	Measurement Level	Frequency Count
INPUT	INTERVAL	5
TARGET	BINARY	1

Variable Levels Summary
(maximum 500 observations printed)

Variable	Role	Number of Levels
matching_status	TARGET	2

Class Variable Summary Statistics
(maximum 500 observations printed)

Variable	Role	Number of Levels	Missing	Mode	Mode Percentage	Mode2	Mode2 Percentage
matching_status	TARGET	2	0	1	52.38	0	47.62

Distribution of Class Target and Segment Variables
(maximum 500 observations printed)

Variable	Role	Formatted Value	Frequency Count	Percent
matching_status	TARGET	1	10000	52.3834
matching_status	TARGET	0	9090	47.6166

Interval Variable Summary Statistics
(maximum 500 observations printed)

Variable	ROLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
date_of_birth	INPUT	0.45246	0.32199	19090	0	0.0	0.50	1
gender	INPUT	0.80825	0.17027	19090	0	0.5	0.75	1
given_name	INPUT	0.32829	0.31956	19090	0	0.0	0.20	1
hic	INPUT	0.48425	0.37265	19090	0	0.0	0.57	1
surname	INPUT	0.31950	0.32409	19090	0	0.0	0.17	1

Interval Variable Summary Statistics by Class Target
(maximum 500 observations printed)

Variable=date_of_birth

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		date_of_birth	0.45246	0.32199	19090	0	0.00	0.50	1
matching_status	0	date_of_birth	0.34266	0.21698	9090	0	0.00	0.38	1
matching_status	1	date_of_birth	0.55228	0.36635	10000	0	0.01	0.62	1

Variable=gender

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		gender	0.80825	0.17027	19090	0	0.50	0.75	1
matching_status	0	gender	0.83500	0.16501	9090	0	0.67	0.67	1
matching_status	1	gender	0.78394	0.17133	10000	0	0.50	0.75	1

Variable=given_name

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		given_name	0.32829	0.31956	19090	0	0.00	0.20	1
matching_status	0	given_name	0.13406	0.13523	9090	0	0.00	0.14	1
matching_status	1	given_name	0.50485	0.33594	10000	0	0.01	0.62	1

Variable=hic

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		hic	0.48425	0.37265	19090	0	0.00	0.57	1.00
matching_status	0	hic	0.12673	0.11807	9090	0	0.00	0.14	0.71
matching_status	1	hic	0.80924	0.17496	10000	0	0.43	0.86	1.00

Variable=surname

Target	Target Level	LABEL OF FORMER VARIABLE	Mean	Std. Deviation	Non Missing	Missing	Minimum	Median	Maximum
OVERALL		surname	0.31950	0.32409	19090	0	0.00	0.17	1
matching_status	0	surname	0.11589	0.11839	9090	0	0.00	0.12	1
matching_status	1	surname	0.50457	0.34034	10000	0	0.01	0.62	1

--
* Score Output
--

--
* Report Output
--

CURRICULUM VITAE

NAME: Xiaoyi Wang

ADDRESS: Department of Industrial Engineering
J B Speed School of Engineering
University of Louisville
Louisville, KY 40292

DOB: Zhengjiang, Jiangsu, China, August 31st, 1962

EDUCATION
& TRAINING: B.S., Telecommunication and Computer Engineering
Tongji University
1980-1985

M.S., Computer Information Systems
College of Business Administration
University of Detroit Mercy
1998-1999

Ph.D., Program in Industrial Engineering
College of Engineering
Wayne State University
2004-2006

Ph.D., Industrial Engineering
J B Speed School of Engineering
University of Louisville
2006-2008

PROFESSIONAL SOCIETIES:
American Society for Quality
Institute of Industrial Engineering
INFORMS

PUBLICATIONS:
"A Use Case Driven Approach to Systematic Functional Decomposition" – *Information and Management 2005*

"Matching Records in Multiple Databases Using a Hybridization of Several Technologies" – *International Industrial Engineering and Management Conference, Coco Beach, Florida, 2007*