1-1-2010

Clustering-Based Pre-Processing Approaches To Improve Similarity Join Techniques

Yufen Tan
Wayne State University

CLUSTERING-BASED PRE-PROCESSING APPROACHES TO IMPROVE SIMILARITY JOIN TECHNIQUES

by

YUFEN "LISA" TAN

DISSERTATION

Submitted to the Graduate School

of Wayne State University

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2011

MAJOR: COMPUTER SCIENCE

Approved by:

_____

Advisor Date

_____

_____

_____

_____

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Dr. Farshad Fotouhi, for his continuing support and guidance. His advice, constructive criticism, and numerous suggestions played a significant role in my research and were highly appreciated. I would like to thank Dr. William Grosky. His guidance strengthened and refined my research. I would also like to thank Dr. Horia F. Pop. His suggestions were a source of inspiration for me and my study. A series of discussions between my advisor, Dr. Farshad Fotouhi, Dr. Bill William Grosky, Dr. Horia F. Pop, and myself helped me prepare for and complete the research detailed in this dissertation. Special thanks to all my committee members, Dr. Farshad Fotouhi, Dr. Bill William Grosky, Dr. Shiyong Lu, and Dr. Ming Dong for all of their guidance, encouragement and support.

Many thanks go to my research colleagues from Wayne State University's Database Information System Laboratory and the faculty and staff of the department of Computer Science, Wayne State University. I also wish to thank my colleagues, John S. Camp who reviewed this dissertation and Joseph F. Sawasky, Morris Reynolds, Daren E. Hubbard, and Robert M. Thompson, from the Division of Computing and Information Technology, Wayne State University, for their encouragement and support to pursue a doctoral degree.

On a more personal note, I would like to thank all the members of my family for their loves and supports, especially my son Matthew who graciously gave me the time to work on my research.

TABLE OF CONTENTS

Acknowledgements …... ……ii

List of Tables ….. ……v

List of Figures …… ……vii

CHAPTER 1 – Introduction 1

SECTION 1.1 – Motivation 2

SECTION 1.2 – Contributions 8

SECTION 1.3 – Structure of the Thesis 10

CHAPTER 2 – General Similarity Join Approaches 12

SECTION 2.1 – Similarity Join Approaches 12

SECTION 2.2 – Summary 19

CHAPTER 3 – Similarity Distance Metrics 21

SECTION 3.1 – Edit Distance Metric 23

SECTION 3.2 – Q-gram Distance Metric 26

SECTION 3.3 – Summary 32

CHAPTER 4 – Attribute-Based Clustering Approach (ABCA) 33

SECTION 4.1 – Introduction 33

SECTION 4.2 – Attribute-based Clustering Approach 37

SECTION 4.2.1 – Identifying Clustered Join Attributes 37

SECTION 4.2.2 – Similarity Join on ABCA 43

SECTION 4.3 – Experimental Evaluation 44

SECTION 4.4 – Summary 51

CHAPTER 5 – Value-Based Clustering Approach (VBCA) 52
SECTION 5.1 – Introduction 52
SECTION 5.2 – Related Work 53
SECTION 5.3 – Proposed Work 56
SECTION 5.4 – Experimental Evaluation 67
SECTION 5.5 – Summary 71
CHAPTER 6 – Comparisons between ABCA and VBCA 73
SECTION 6.1 – Environmental Setup 73
SECTION 6.2 – Experimental Studies 74
SECTION 6.3 – Cost Comparisons 81
SECTION 6.4 – Distance Metric Studies 85
SECTION 6.5 – Summary 89
CHAPTER 7 – Future Work 91
SECTION 7.1 – Summary of Contributions 91
SECTION 7.1 – Future Work 93
Bibliography 97
Abstract. 106
Autobiographical Statement 108

LIST OF FIGURES

Figure 1-1 Example data and result for a similarity 5
Figure 1-2 Example for a duplicate elimination similarity join 6
Figure 3-1 Compute edit Distance between “test” and “tent” 25
Figure 3-2 Edit Distance Functions 26
Figure 3-3 Q-gram Similarity Join in SQL Version 29
Figure 3-4 Revised Q-gram Similarity Join in SQL Version 30
Figure 3-5 Q-gram Distance Function in High-Level Language Version 31
Figure 4-1 Commercial SQL Join vs Similarity Join 34
Figure 4-2 Similarity Join on Different Group Attributes 35
Figure 4-3 Attribute-Based Clustering Approach 40
Figure 4-4 BEA Clustering Algorithm 41
Figure 4-5 Binary Split Algorithm 42
Figure 4-6 Clustering Attributes and Similarity Join 43
Figure 4-7 ED Precision on Attribute(s) 46
Figure 4-8 ED Recall on Attribute(s) 47
Figure 4-9 Q-gram Precision on Attribute(s) 48
Figure 4-10 Q-gram Recall on Attribute(s) 48
Figure 4-11 ED and Q-gram Precision on Affinity Clustered Attributes 49
Figure 4-12 ED and Q-gram Recall on Affinity Clustered Join Attributes 50
Figure 4-13 ED and Q-gram F-measure on Affinity Clustered Attributes 50
Figure 5-1 Value-Based Clustering Approach 56
Figure 5-2 Symmetric Dissimilarity Matrix 58

Figure 5-3 Sammon Mapping 60
Figure 5-4 Fuzzy Clustering Results 61
Figure 5-5a Tree Structure 64
Figure 5-5b Tree Structure 65
Figure 5-5c Tree Structure 66
Figure 5-5d Tree Structure 67
Figure 5-6 ED and Q-gram precision on affinity clustered attributes 69
Figure 5–7 ED and Q-gram recall on affinity clustered join attributes 69
Figure 5–8 ED and Q-gram F-measure on affinity clustered attributes 70
Figure 5-9 Running Time of Similarity Join Algorithms 71
Figure 5-10 DB Temporary Spaces 71
Figure 6-1 Precision on ED Using Various Predicates 75
Figure 6-2 Precision on Q-gram Using Various Predicates 76
Figure 6-3 Recall on ED Using Various Predicates 77
Figure 6-4 Recall on Q-gram Using Various Predicates 78
Figure 6-5 F-measure on ED Using Various Predicates 79
Figure 6-6 F-measure on Q-gram Using Various Predicates 80
Figure 6-7 Running Time on 5,000 Records 83
Figure 6-8 Running Time on 50,000 Records 84
Figure 6-9 Running Time on 500,000 Records 84
Figure 6-10 Predicates vs. Distance Metrics 87
Figure 6-11 Pre-processing vs. Distance Metrics on Running Time 88

LIST OF TABLES

Table 4-1 Attribute Usage Matrix 38

Table 4-2 Attribute Affinity Matrix 38

Table 4-3 Clustered Affinity Matrix 39

Table 5-1a Partition Coefficient vs. Classification Entropy 64

Table 5-1b Partition Coefficient vs. Classification Entropy 65

Table 5-1c Partition Coefficient vs. Classification Entropy 65

Table 5-1d Partition Coefficient vs. Classification Entropy 66

Table 6-1 Parameters used for Pre-processing 81

Table 6-2 Cost of Pre-processing Approaches 82

CHAPTER 1

INTRODUCTION

With the tremendous success of technologies like the Internet and the Web, a significant, ever-growing amount of information is available from an increasing number of databases and other information systems. The search for highly efficient and accurate algorithms to find and consolidate all instances of similar objects in these databases is becoming increasingly important. Finding techniques that accurately join similar but complex data in minimal time is not a trivial task.

In general, similarity join approaches for finding redundancies and consolidating data are widely used in data integration [HM 04], bio-informatics, web searching, and data management environments. Those approaches construct the basis of data models in which data can be linked, queried, condensed, or cleaned based on the degree of data similarity of objects or values.

With more and more companies, institutions and even business units inside companies and department units inside institutions storing information in their local database systems, the availability of generalized and specialized information data has been multiplying enormously. Really, a very limited amount of data is needed to be processed for coping with our day-to-day tasks. The effort condensing or finding the small pieces of information users are after is the greatest challenge for providing a high-level of availability of relevant data.

Fortunately, to some degree, a turning point has been reached in resolving the above conflicts [BN 06] in Information and Computer Science. Researchers have shifted their focus from how to make information data available to how to make information data more useful. However, the biggest concern in retrieving useful data from vast available data is data duplication. Appropriate similarity join approaches are the common way to handle data duplication issues. There exist many similarity join approaches. An approach to be chosen is typically based on the purpose of user's goal or the type of applications.

This research is focused on one of the similarity join approaches' issues. Similarity join is described differently from various research communities as record linkage, entity identification, or the same-object problem. In general, the term similarity has been broadly used in a variety of areas with different usage and definition. In Computer Science, similarity is referred to as a similarity relationship to applications, objects, record sets or attributes. Similarity Join has been specifically used in fields of information retrieval [Rij 79, SM 83] and knowledge-based systems. The integration processing [SSS04] will have to exceed conventional query processing when the integration needs to handle significant amounts of data from multiple sources based on their similarity values.

1.1 Motivation

The concept of similarity join [Alb 67] was introduced during the 1960's. Since it has been used for a long period of time and in various areas, this methodology has been attracted significant attention in different research

communities including statistics, artificial intelligence and databases. Each community [ABU 79] has formulated the problem differently and different techniques have been proposed. Statistics refers to a similarity join as record linkage armed at minimizing the probability of misclassification. AI uses supervised learning to learn the parameters of a string to edit distance metrics. Database uses knowledge intensive approach to edit distance as a general record match scheme. This study focused on improving similarity joins in structured databases.

For a real-world example of the type of joins that could be of interest, consider a prospective graduate student dataset for recruitment purposes at a public university. One source of data could come from the university's relational management system that keeps track of prospective students with recruiting records stored in an SQL server database. Another source of data is the university's Oracle database of all students previously and currently enrolled at the university. One need is to identify all prospective graduate students who have never taken classes in the past and then personalize automated communications designed to attract previously enrolled and never enrolled students. In this case, a similarity join algorithm is needed to identify duplications across the databases so that appropriate actions for cleaning up the data and corresponding with students can take place.

When one tries to get a report based on both existing students' data and potential students' data, one cannot assume that there exist global identifiers for those data across two structured sources even though there are unique

identifiers in each individual record. The example shown in Figure 1-1 demonstrates some of common problems in application domains. The scenarios causing “data dirt” are typically raised by a) missing data: for data entry, some of the fusion data are produced and used by different entities for different purposes. For example, a person’s age is an important marketing fact for a recruiting purpose from a campaign's point of view, but may not be useful for accounting purposes from an administrative point of view. b) data errors: although the same naming conventions may be used in different databases, data can still contain errors**,** normally caused by mistyping; for example “John Smith” and “John Smth”. c) data duplication problem: to map real-world entities into a data set, the same person from different systems might have been created multiple times by using different conventions; For example, the same recruiting or current students may be created in different tables or schemas by slightly different but correct names, such as “John Smith”, “Smith John” or “J. Smith.” This problem is sometimes referred to in the literature as the object or instance identification problem or the record linkage problem [TKM 2001]. d) different data formats: too often standard notation isn’t imposed when people do data entry, and all kinds of free-form fields may be embedded. For example, a street field incorrectly contains the zip code and the country name. Or records use synonyms as well as abbreviations to refer to an object that is represented by full names in another record. e) data inconsistencies: for example, the age of the same person may be different in different databases for any number of reasons. Therefore, a query to correlate these databases and create a campaign report using either a natural equality join

operation or similarity string join methods might fail to produce the desired results. To effectively address this data integration problem, one needs techniques for identifying all pairs of approximately matching strings in databases [GIJMS 01, GIJS 03].

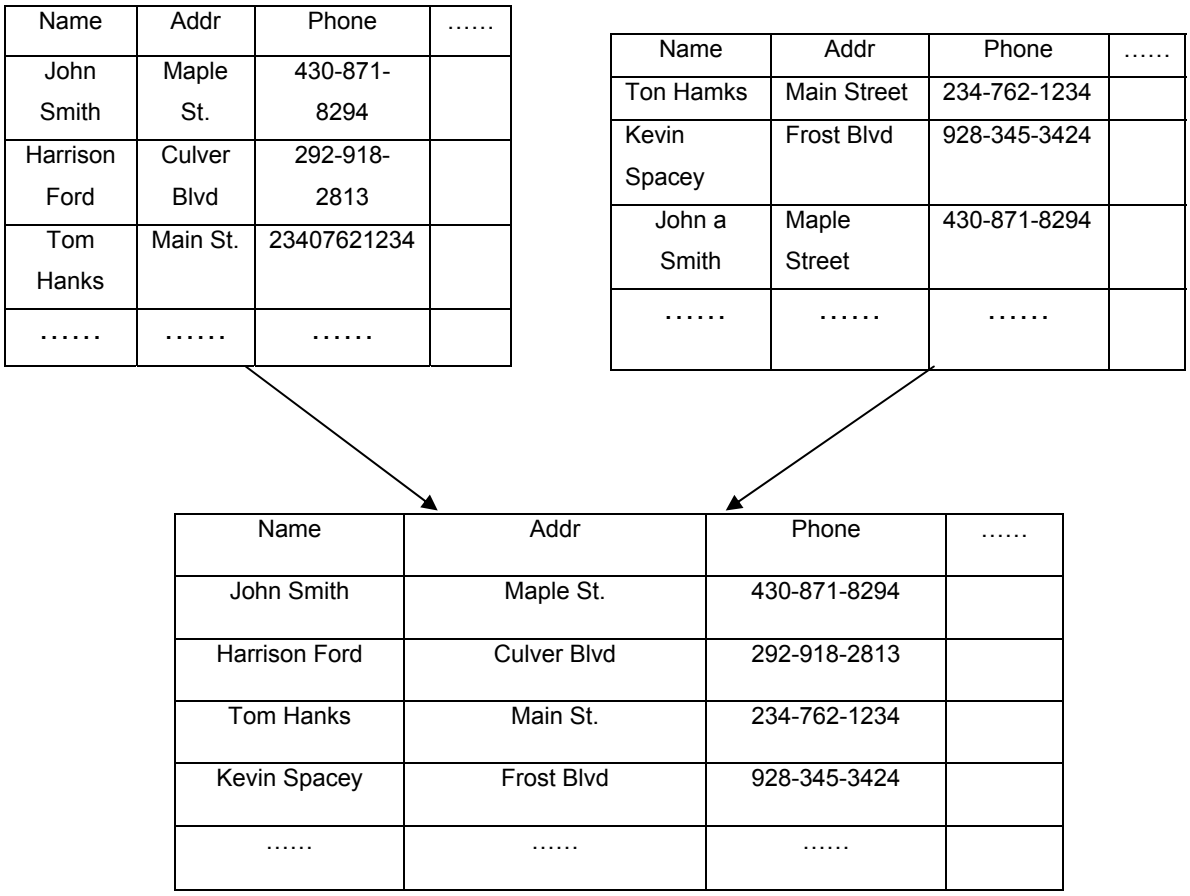
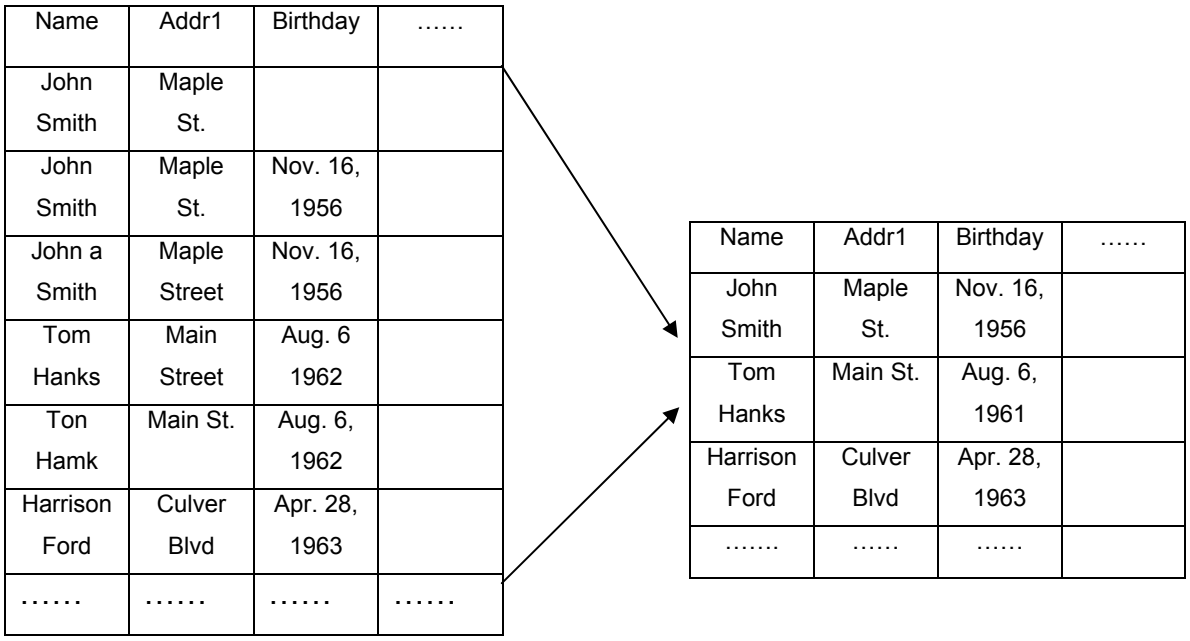


Figure 1-1: Example data and result for a similarity

The above problems could be handled by a string similarity join. However, effectively and efficiently performing a similarity join even in a local database itself is a great challenging task and topic of current research. This includes the results presented in the following chapters of this dissertation. In general, with diverse data applications, this research assumed a virtually integrated scenario where the data resides in different databases accessible only through possible

very limited query interfaces. In this case, identifying prospect entries who became students based on possibly conflicting representation of their names, addresses and phone numbers is a complex problem. This issue can be addressed by the proposed techniques developed for this study.

Figure 1-2 shows another challenge during data identification, namely the identification and reconciliation of 'tuples' representing the same real-world entity. The input relation represents the combined information on recruiting information from a number of source systems, which might overlap and provide incomplete or imprecise information. Besides that, the example also illustrates a complex similarity involving in the join like the name, addr1, or birthday, and the field value might be missed in some of them.



Name	Addr1	Birthday	……
John Smith	Maple St.		
John Smith	Maple St.	Nov. 16, 1956	
John a Smith	Maple Street	Nov. 16, 1956	
Tom Hanks	Main Street	Aug. 6 1962	
Ton Hamk	Main St.	Aug. 6, 1962	
Harrison Ford	Culver Blvd	Apr. 28, 1963	
……	……	……	……

Name	Addr1	Birthday	……
John Smith	Maple St.	Nov. 16, 1956	
Tom Hanks	Main St.	Aug. 6, 1961	
Harrison Ford	Culver Blvd	Apr. 28, 1963	
…….	……	……	

Figure 1-2: Example for a duplicate elimination similarity join

Moreover, any automated join process needs to handle the fact that more than two records might represent the same real-world entity, and among these representations might exist varying degrees of similarity. Since all of them are

related to the same real-world entity, the proposed approach in this study can identify similar objects based on clustering related fields or attributes on the integrated data sources.

The presented examples show that accurate actions based on complex similarity conditions are not trivial to make. For example although the data on current students, Tom Hanks and Ton Hamk look different, they actually represent the same person. Therefore, the design of similarity predicates as part of the design of the integrated system is a complicated task which involves the analysis of falsely identified and falsely unidentified objects.

No matter during the integration of query results from multiple data sources or in the same database schema, dirty records tremendously comprise the further tasks for improving data quality like transformation, outlier detection, data mining etc. In current existing SQL-based commercial products, this research routinely use the group by operator [CGK 06] and the key attributes of the records in combination with aggregate functions for reconciling divergent non-key attribute values to eliminate record duplications. Unfortunately, those approaches are restricted to known join attributes. In other words, all the existing approaches were under the assumption that the join attributes have been pre-defined so the existing approaches focused on the efficiency of similarity join approaches. In this research, other attributes have been introduced to help similarity join functions. The research shows the clustering will help to identify clustering related attributes to improve overall precision of the results. This dissertation proposed two approaches to identify those attributes. The first

approach is based on attribute usages from applications. Another approach is based on actual value of attributes. The experimental results have showed some improvements by utilizing identified clustering attributes as join predicates on well-known similarity join metrics, namely Edit Distance metric [CRF 03] and Q-gram Distance metric.

1.2 Contributions

Identifying clustering related attributes before applying similarity join approaches is the main contribution in this dissertation. The novel aspects of clustering approaches outlined in the following chapters are described here in more detail. Furthermore, some research results were previously published and are listed with the respective references.

Attribute-Based Clustering Approach: Using clustering related attributes as join predicates is a new concept. This research has developed an approach to find those attributes before applying similarity join. The attribute-based approach is to analyze actual attribute usages from applications to determine how attributes are closely related. In the attribute-based approach, it has also used the Greedy Method in the development to reduce the overall complexity. Furthermore, complex similarity conditions and special aspects of similarity relations are most often neglected in related research. Corresponding research results were previously published for instance in [TFG 09]. During the verification, this research developed a high-level language version of Q-gram that tremendously improved performance in sense of space and time.

Value-Based Clustering Approach: this is another novel pre-processing approach for similarity join on value-based clustering related attributes in common data integration scenarios. This data pre-processing approach on identifying clustering related attributes was developed as part of a duplication identified process and, alternatively, using the extensibility interfaces of the database management system Oracle 10g. Furthermore, the value-based clustering approach provides a natural way to find the clustering related attributes. This approach has the beauty of using the attribute value of dataset itself without any pre-knowledge of the application as the first approach does. Finally, this approach has also applied a Greedy concept to reduce overall complexity. The results of the proposed approach were previously published in [TFGPM 09].

To summarize, the work presented in this dissertation targets the inclusion of data pre-processing on similarity-based concepts in integration scenarios. This problem is addressed on choosing or identifying optimal predicates. Approaches on identifying clustering related attributes based on either attribute usage or value of the data set are introduced, suitable for a wide range of applications. Aspects of the development of such pre-processing approaches are described for homo-generous or semi-generous integration scenarios. For evaluation purposes the focus was on string similarity predicates, because there is a general lack of support for these in current data management as well as only partial solutions provided by current research.

1.3 Structure of the Dissertation

This dissertation is structured to provide a reader who has a solid comprehension of database and information retrieving all the necessary information to fully understand the scope and contents of the described research results. Literature references are used to refer to sources where detailed descriptions are described or further interests can be found in mentioned topics beyond the scope necessary to understand the content of this dissertation.

After this brief introduction to the motivation, the contributions, and structure of this dissertation, Chapter 2 will give more detail about what similarity join approaches have been proposed on the current state of distributed environments [OV 91]. Typical aspects of similarity join and resulting problems are introduced based on the commonly considered characteristics of distributed and homogeneous databases. Important aspects of similarity join research areas on query processing in distributed and homogeneous environments are briefly described. The main approaches to attribute-based clustering, related attributes, and value-based clustering related attributes are positioned based on the previously introduced characteristics and related to the contributions of this dissertation.

Chapter 3 includes an overview of common used similarity join distance metrics such as edit distance metric and Q-gram distance metric. This chapter sets the evaluation methods for the experiments in the later chapters.

Chapter 4 and 5 are the chapters describing the main contributions of the dissertation. Both chapters can be distinguished by the kind of application

knowledge they target. The approach presented in Chapter 4 described pre-processing algorithm where pre-knowledge of application is mandated and the usages of attributes affect the clustering results. This pre-processing technique groups the attributes which are commonly used together in the querying the data sources. To get the optimal attribute partitioning point [SW 85] on the attribute set, the proposed approach is to gracefully calculate partitioning quality based on the usage of the attributes. That knowledge information will significantly impact the result of approximate string similarity join. Chapter 5 proposes a different pre-processing approach by utilizing the nature of the dataset. For that purpose, identifying clustering related attributes does not rely on the previous knowledge of the applications but on the dataset itself. .This approach is generally applicable in any integration scenarios and it is much flexible and promising approach and the value of attributes impacts the clustering results. Changing attributes' value or adding new records might change the clustering results. Furthermore, the implementations for identifying clustering related attributes are outlined and their efficiency is evaluated. There is a substantial improvement on the complexity when the greedy approach is applied. Also using high-level language to implement Q-gram distance metric has great improvements on time and space.

Chapter 6 compares the complexity among Edit Distance and Q-gram on identified clustering fields. In Chapter 7 the dissertation is concluded by a summary and an outlook on directions of possible further work is given.

CHAPTER 2

GENERAL SIMILARITY JOIN APPROACHES

Similarity Join is an important operation for many applications in the current digital information era. In brief, a similarity join operation computes all records (x, y) within a defined threshold ε for any given data sets. It is widely used to solve various problems in many application domains, such as data integration, data cleansing, data de-duplication, name matching, duplicate Web documents detection and information retrieval. The following section includes descriptions of the most recent approaches for similarity joins.

2.1 Similarity Join Approaches

Xiao et al in [XWLS 09] addressed one of the traditional form of the similarity join operation concerns, which is to require a user to input a similarity threshold. In many application scenarios, the threshold is not known before hand and is likely to vary according to datasets and application scenarios. The common traditional approach is to compute similarity values for all possible record pairs and then select the top k pairs. Xiao's approach is to carefully exploit the prefix filtering principle and upper bound the similarity values of unseen pairs to reduce the number of record pairs needing to be compared. The rationale behind the prefix filtering principle is that if two records are similar, some fragments of them should overlap with each other. The proposed approach is to devise an incremental version of prefix filtering algorithm. In the incremental version, any candidate pairs whose similarity value is less than the defined threshold hold cannot be discarded. Also the largest k pairs are the only

pairs needed to be kept. The other revision from the original prefix filtering algorithm is to devise a new stopping condition. Xiao's algorithm will stop the execution when the similarity value of the current k-th result is larger than the next similarity threshold. There are several advantages to use top-k similarity join. The first one is to compute most similar record pairs without the need to specify a similarity threshold. The second one is to support interactive near duplicate detection applications [SB 02], where users are presented progressively with top-k most similar record pairs. The last one is that it produces the most meaningful results when users perform a similarity join under certain resource or time constraints. In brief, Xiao's approach provides an effective way to identify the best threshold value on the common similarity join approaches.

Gravano et al in [GIJS 03] presented a technique for building approximate string join capabilities on top of commercial database applications by exploiting facilities already available in them. It divides a known join string predicate into short substrings of length q, called Q-grams, creates an auxiliary table to hold Q-gram related information, and takes into account both positions of individual matches and the total number of such matches. The theory of this approach enables one to say when two strings are within a small edit distance, they are treated as similar strings. This occurs when they share a large number of q-grams in common. The outcome from this approach supports join string predicates like "name similar to Campbell" with an accepted error rate of ε.

Jestes et al in [JLYY 10] proposed a novel technique, called Probabilistic String Similarity Joins, to solve the problem in probabilistic string databases, using the expected edit distance as the similarity measure. Jestes' paper described two probabilistic string models to capture the uncertainty in string values in real-world applications. The string level model is complete, but is expensive to represent and process. The character-level model has a much more succinct representation when uncertainty in strings only exists at certain positions. The researchers designed efficient and effective pruning techniques that can be easily implemented in existing relational database engines for both models. Although the probabilistic string similarity join demonstrated order of magnitude improvements over the baseline, the approach is best suited to probabilistic datasets.

Liu et al in [LLFZ 08] proposed a Nondeterministic Finite-state Automation-based method for effective approximate string search. The purpose of the NFA-based approach is to eliminate false positive results from existing similarity join algorithms. To address the problem, the method models strings as 'trie' (from retrieval) and constructs an NFA on top of the trie. A trie is an ordered tree data structure that is used to store strings. The idea behind trie is that all strings sharing a common stem or prefix hang off a common node. All the descendants of any one node have a common prefix of the string associated with that node, and the root is associated with the empty string. The elements in a string can be recovered in a scan from the root to the leaf that ends a string. All strings in the trie can be recovered by a depth-first scan of the tree. Trie is used

to do a fast exact-search in large string collection. Moreover, trie provides some advantages as looking up strings is faster; tries can require small space when they contain a large number of short strings and tries help with longest-prefix matching. In brief, Liu's trie indexes and tree automata (TITAN) method employ an NFA-based method to identify similar strings based on edit distance and by adopting tree automata theory. The TITAN is best used on effective approximate string search or approximate string join area.

Chaudhuri et al. in [XWLS 09] propose Set Similarity Join (SSJoin) operator as a foundational primitive. Their operator extends Sarawagi et al. [SK 04] set overlap approach without requiring plug-in functions during the implementation of each similarity function. The basic theory behind SSJoin approach is that when two strings are almost equal, their overlap similarity is high and this somehow is a natural similarity join predicate to express. Formally, SSJoin operator is defined by mapping the strings to sets and measuring their similarity using set overlap. They propose to partition the set of strings by delimiters which is a well-known string mapping method called Q-gram. The overlap similarity is the weight of the intersection of the string sets. Their implementation of SSJoin operator can be easily integrated into a relational database system, applied to a variety of other textual similarity functions, and even composed with the top-k operator to address the form of top-K queries. The integration with top-K can find the threshold which produces the best matches. In short, their approach is used as a primitive operator SSJoin for performing similarity joins on textual or non-textual similarity functions.

Bayardo et al. in [BMS 07] proposed a simple algorithm based on novel indexing and optimization strategies. Their approach works better on a large collection of sparse vector data in a high dimensional space [NU 00]. With the Web-based applications growing, the number of distinct search queries issued over a single day to any large search engine is in the tens of millions. If any one wishes to perform collaborative filtering on data from sites such as Google or eBay, the algorithms need to scale to tens of millions of users. Their approach is to deal with those large scales of data. The rationale behind the approach is to exploit the inverted list, store the vector weights within the inverted index itself instead of scan each one individually, and accumulate scores in a hash-based map. The approach offers improved locality and avoids the logarithmic overhead of the heap structure and dramatically reduces the number of candidate pairs considered, reduces overhead such as index construction and inverted list scanning during score accumulation, and vastly decreases the search space by ordering the vectors in addition to the dimensions. Shortly, their approach aggressively exploits All Pairs Similarity Search approaches and is suitable for a large collection of sparse vector data in a high dimensional space.

Ding et al. in [DTS 08] proposed an efficient Trajectory Similarity Join (TSJoin) for large sets of moving object trajectories. With introducing more technologies on Location-Based Services (LBS), Wireless Communication Systems, Miniaturization Computing Devices and Global Positioning Systems have been bringing researchers' attention on how to handle the intrinsic characteristics of the datasets. Their previous research focused on efficient

similarity search in time series datasets. Those approaches have been improved efficiency for a variety of time series application domains. Their wDF approach intends to meet moving object trajectories' need. The moving object trajectories constitute a special category of time series data. wDF utilizes a novel distance measure based on Frechet distance to effectively identify similar trajectories, applies lower and upper bounding approximations of the exact distance measure to the spatio-temporal indices to prune a significant portion of the search space, and combines the distance calculation with incremental accesses to the spatio-temporal indices in the native space. Briefly, TSJoin approach is appropriate for location-related time series data sets.

Kriegel et al. in [KKR 10] proposed a new Probabilistic Similarity Join approach to handle vague and uncertain data. Vague and uncertain data are expressing as spatio-temporal query processing of moving objects, sensor databases or personal identification systems [TKM 01]. For example, on mobile services, the mobile devices consistently change their locations so that fixing location information is almost impossible to obtain; on multimedia databases [FL 95] such as image [SM 00] or music databases, face recognition and fingerprint analysis from personal identification systems can not be exactly determined. The uncertain data can be handled by assigning confidence intervals to the feature values, or by specifying probability density functions indicating the likelihoods of certain feature values. Their approach uses probabilistic distance functions to measure the similarity between uncertain objects. The distance function is defined as the Euclidean distance between two feature vectors. The distance

range join of two multidimensional or metric sets R and S is the set of pairs where the distance of the object does not exceed a given threshold ε. Their approach doesn't need distance functions which express the similarity between two objects by exactly one numerical value and can be applied to any uncertain data sets.

Kalashnikov et al. in [KP 07] proposed two fast similarity join approaches for multi-dimensional data. The authors conclude that most existing approaches are good for high-dimensional disk-based joins over large amounts of disk-based data. In reality, the data sets of multimedia databases, data mining databases, location based applications and time-series analysis can be high dimensional and/or low dimensional. Their approaches work well with either dimensional datasets not like the previous Grid-join and EGO*-join. The main concept on their approaches is to utilize main memory in the system instead of disk spaces in the system since the memory becomes cheap and real-time computation may be critical and require main memory evaluation. The new Grid-join is based upon a uniform grid, builds an index on the points of dataset, and processes a circle region query for each point so the performance tends to faster than the original Grid-join which builds an index on the region. The EGO*-join is based upon the original EGO-join algorithm and is able to determine non-joinable sequences. The improvement of EGO*-join comes from the large reduction of the number of sequences. Their experiments show Grid-join is good for low-dimensional datasets and EGO*-join is good for high-dimensional datasets.

Wen et al. in [WAK 08] proposed a similarity join approach for XML data. XML is an Extensible Markup Language which has been increasingly used for data exchange on the Internet and has been recommended by the World Wide Web Consortium. XML is currently popularly used in many applications because it can represent any kind of data from multiple sources. With the growing amount of XML data on various applications in different un-structured systems, the more similar contents will exist in the different sources, and the more integration will need to extract useful information from those heterogeneous sources. Their approach is to resolve this emergent need by serializing XML data as XML node sequences, extracting semantically coherent subsequences, filtering out dissimilar subsequences using textual information, and extracting pairs of subsequences as the final result by checking structure similarity. This serialization approach works extremely well on tree structure representation of XML data because it is hard to measure the similarity on the tree structure data.

2.2 Summary

This chapter includes descriptions of similarity join approaches which have been studied recently. Each approach has its strengths in a way which mostly specifically meets the needs of specific applications. Generally, what similarity join approach to be used is really depending on application domains and there is no best similarity join approach that works better than any other approach in all application domains.

However, all applications have some common characteristics, captured under the metric space model. There is a universe $\mathbb{X}$ of *objects*, and a

nonnegative *distance function* $d : \mathbb{X} \times \mathbb{X} \longrightarrow \mathbb{R}^+ \cup \{0\}$ defined among them. Objects in $\mathbb{X}$ do not necessarily have coordinates (for instance, strings and images). The distance function gives us a dissimilarity criterion to compare objects from the database. Thus, the smaller the distance between two objects, the more “similar” they are. The distance satisfies some properties in a metric space.

The metric properties hold for many reasonable similarity approaches. Typically, there exists a finite *database* or *dataset* **X** which is a subset of the universe of objects **U** and can be preprocessed to build an *index*. As the distance is expensive to compute (think, for instance, in comparing two fingerprints in the homeland security database), it is customary to define the complexity of the search as the number of distance evaluations performed, disregarding other components such as CPU time for side computations and even I/O time. Thus, the ultimate goal is to implement an approach so as to compute many fewer distances when solving similarity join queries. Also different metric approaches can be chosen to resolve different domain issues.

CHAPTER 3

SIMILARITY DISTANCE METRIC

Just as similarity join approaches vary, there are a number of distance metrics for measuring similarity. Goldstone in [Gol99] did a rough classification models for measuring similarity as geometrical models, featural models, alignment-based models and transformational models. Out of those models, transformational models are applied in approximate string matching [BN 99] which is most relevant for the research presented in this research.

The most common usage of similarity measures refers to distances in metric space defined as follows on [Wei99].

Definition 3.1 A metric space is a set S with a global distance function (the metric g) which gives the distance between every two points $a,b \in S$ as a non-negative real number $a(a,b) \in R^{+}$. A metric space must also satisfy

1. $\forall a,b \in S : g(a,b) = 0 \Leftrightarrow a = b$ (Constancy of Self-similarity)
2. $\forall a,b \in S : g(a,b) = 0 \Leftrightarrow g(b,a)$ (Symmetry)
3. $\forall a,b,c \in S : g(a,b) + g(b,c) \geq g(a,c)$ (Triangular Inequality)

Accordingly similarity metric [ABU 79] is a special case of the distance similarity or distance measure defined as:

Definition 3.2 A similarity metric is a similarity measure that satisfies all axioms for a metric.

Typical examples for a metric space are the n-dimensional Levenshtein distance space [Lev 66], Euclidean distance space, Minkowski distance,

Manhattan distance space, Chebyshev distance space and so on. There are several advantages of similarity metrics resulting from the metric axioms, especially when the metrics are used for data processing. Based on those metric properties, the approximate string similarity join is defined by Navarro in [Nav01] as follows:

Definition 3.3 Let $\sum$ be a finite alphabet of size $|\sum| = \sigma$. Let $t \in \sum^*$ be a text of length $n = |t|$. Let $p \in \sum^*$ be a pattern of length $m = |p|$. Let $k \in R$ be the maximum error allowed. Let $d : \sum^* X \sum^* \to R$ be a distance function. The problem of approximate string matching in texts is: given $t, p, k,$ and d, return the set of positions j such that there exists i for which $d(p, t[i..t[j]) \leq k$.

Rather than finding position within texts, this research focuses on finding the degree of similarity between strings in sets, which will, for instance, be the number of the difference characters between strings. Hence, the definition is slightly modified.

Definition 3.4 Let $s \in R_1$ be a source string set and $t \in R_2$ be a target string set over the same alphabet. The problem of approximate string similarity join from two datasets is given $s, t, k,$ and d, return the set of all strings $T \in R_1 \cup R_2$ such that $d(s,t) \leq k$.

Suitable distance measures for string values are transformational measures according to the classification given by Goldstone in [Gol99], i.e. they measure the dissimilarity in terms of operations necessary to transform one string to another. Various distance measures can be distinguished based by

- The kinds of operations allowed, and

- The costs assigned to these operations.

Typical operations are the deletion, insertion, replacement, or transposition of characters. Other considered operations for instance include reversals or the permutation of complete substrings, such as for instance the Block edit distance introduced by Tichy in [Tic84]. Similarly, Ukkonen in [Ukk92] described similarity of strings in terms of common substrings of a fixed length called *q*-grams. The most common string distance measures based on the typically considered operations mentioned above are Levenshtein Distance, Jaccard Distance, Generalized Edit Distance, Hamming Distance, Q-gram Distance, Euclidean Distance, etc. In this dissertation, the experiments are utilizing Edit Distance and Q-gram Distance metrics to evaluate proposed techniques. The detail information of these two distance metrics will be presented in the following sections.

3.1 Edit Distance Metric

Edit distance is a fundamental and common distance [XWL 08, KMK 09] used in various research communities. There are different types of edit distances. If different operations have different costs or the costs depend on the characters involved, the operation is referred to general edit distance. Otherwise, if all the operations cost 1, the operation is referred to Levenshtein edit distance or simple edit distance or just edit distance (LD). For simplicity this research is going to focus on the simple edit distance in this dissertation. In the simple edit distance, this research simply seeks for the minimum number of insertions, deletions and substitutions to transform one string to another. If

$x_i = x[1]x[2]...x[i]$ and $y_j = y[1]y[2]...y[j]$ are strings with all character $x[k] \in \sum, 1 \le k \le i$ and $y[l] \in \sum, 1 \le l \le j$ over one alphabet $\sum$, the edit distance of the two strings can be computed as follows:

$$LD(x_i, y_j) = \begin{cases} 0 & if \quad i = j = 0 \\ \infty & if \quad i \le 0 \vee j \le 0 \\ LD(x_{i-1}, y_{j-1}) & if \quad x[i] = y[j] \\ \min \begin{pmatrix} LD(x_i, y_{j-1}) \\ LD(x_{i-1}, y_j) \\ LD(x_{i-1}, y_{j-1}) \end{pmatrix} & else \end{cases}$$

Three properties are described as follows.

- Insertion – an insertion is a character needs to be inserted into s to make s match t at the same position.
- Deletion – a deletion is a character needs to be deleted from s to make s match t at the same position. This is the opposite of insertion.
- Substitution – a substitution is a character needs to be replaced on s to make s match t at the same position.

For example,

- If s is "test" and t is "test", then $LD(s,t) = 0$, because no transformations are needed. The strings are already identical.
- If s is "test" and t is "tent", then $LD(s,t) = 1$, because one substitution (change "s" to "n") is sufficient to transform s into t.

Generally, to compute LD, one could imagine a matrix LD will be filled with $LD(i, j)$, where the cell $LD(i, j)$ of the matrix will be set to the minimum number

of operations needed to match source string $s_{0..i}$ to target $t_{0...j}$. Based on the LD definition above, the three cases are computed as follows:

a. The cell $d_{i...0} = i$.

b. The cell $d_{0...j} = j$.

c. The cell $d_{i,j} =$ if $(s_i = t_j)$ then $d_{i-1,j-1}$

else $1 + \min(d_{i-1,j}, d_{i,j-1}, d_{i-1,j-1})$

The above rationale is obvious. First, $s_{i..0}$ and $t_{0..j}$ represent the edit distance between a string of length i or j and the empty string, where i or j deletions are needed on the nonempty string. For two nonempty strings of length i or j, this research assumes inductively that all the edit distances between shorter strings have already been computed, and try to convert $s_{1..i}$ into $t_{1..j}$. Figure 3.1 illustrates a general approach to compute LD("test," "tent").

		t	e	s	t
	0	1	2	3	4
t	1	**0**	1	2	3
e	2	1	**0**	1	2
n	3	2	1	**1**	1
t	4	3	2	2	**1**

Figure 3-1: Compute edit Distance between "test" and "tent"
The bold entries show the path to the final result

To fill in the matrix by using the algorithm as shown in the following Figure 3-2, the upper, left, and upper-left neighbors of a cell are computed prior to

computing the cell. This can be easily achieved by either a row-wise left-to-right traversal or a column-wise top-to-bottom traversal. Therefore, the complexity for the algorithm is $O(|s|*|t|)$ in the worst and average case, where the space is required only $O(\max(|s|,|t|)$ because if the approach uses a column-wise processing, only the previous column needs to be stored to compute the new one.

editDistance(s,t)

1. set $|s|$ to be the length s
2. set $|t|$ to be the length t
3. if $|s|=0$, return $|t|$
4. if $|t|=0$, return $|s|$
5. define a nxm matrix
6. initialize the first row of the nxm matrix to $d_{i,0}=i$
7. initialize the first column of the nxm matrix to $d_{0,j}=j$
8. check each character of s in sequence where i from 1 to n
9. check each character of t in sequence where j from 1 to m
10. if $s[i]$ is equal to $t[j]$, the cell $d_{i,j}=d_{i-1,j-1}$
11. if $s[i]$ isn't equal to $t[j]$, the cell $d_{i,j}=1+\min(d_{i-1,j},d_{i,j-1},d_{i-1,j-1})$
12. after finishing all characters' checking in s and t from steps (8,9,10,11), return the cell $d_{n,m}$

Figure 3-2: Edit Distance Functions

3.2 Q-gram Distance Metric

Q-gram is one of the popular distance metrics [Wang 10] that can be used to calculate similarity of two strings. The Q-gram concept is coming from an n-

gram and can be traced to an experiment by Claude Shannon's work in Information Theory [En] in early 1900. An *n*-gram is defined as a subsequence of *n* items from a given sequence. The items in the definition can be phonemes, syllables, letters, words or base pairs according to the application. *n*-gram models are widely used in statistical natural language processing. It can also be used for efficient approximate matching by converting a sequence of items to a set of n-grams. However, the set of n-grams make the approach lose information about the strings. A positional Q-gram was born to compensate some weaknesses of n-grams.

In the literature, the notion of positional Q-gram [Sutinen E. 1995] is "Given a string s, its positional Q-grams are obtained by "sliding" a window of length q over the characters of s. Since Q-grams at the beginning and the end of the string can have fewer than q characters from s, the approach introduces new characters "#" and "$" not in $\sum$ (finite alphabets), and conceptually extend the string s by prefixing it with q-1 occurrences of "#" and suffixing it with q-1 occurrences of "$". Thus, each Q-gram contains exactly q characters, though some of these may not be from the sigma."

Definition 3.6 A positional Q-gram of a string s is a pair $(i, s|i...i+q-1)$, where $s|i...i+q-1|$ is the Q-gram of s that starts at position I, counting on the extended string. The set G_s of all positional Q-grams of a string s is the set of all the $|s|+q-1$ pairs constructed from al Q-grams of s.

The rationale behind the use of Q-grams is that when two strings s and t are within a small edit distance of each other, they share a large number of Q-

grams in common [Sutinen E. 1995]. With no edits, any string with a length L will have $L+q+1$ Q-grams. For edit distance k, there could be at most k replacements, insertions or deletions which can be performed. Two strings *s* and *t* with edit distance $\leq k$ have at least $[\max(|s|,|t|)+q-1]-kq$ Q-grams in common, where $|s|$ is a number of characters in a string s and $|t|$ is a number of characters in a string t.

For example, if one wants to generate the positional Q-grams of length $q=3$ for Benjamin Peterson, its Q-gram will be {(1,##B), (2,#Be), (3,Ben), (4,enj), (5,nja), (6,jam), (7,ami), (8,min), (9, in%), (10,n%P), (11,%Pe), (12,Pet), (13,ete), (14,ter), (15,ers), (16,rso), (17,son), (18,on$), (19,n$$)}; similarly, the positional Q-grams of length $q=3$ for Peterson Benjamin are {(1,##P), (2,#Pe), (3,Pet), (4,ete), (5,ter), (6,ers), (7,rso), (8,son), (9,on%) (10,n%B), (11,%Be), (12,Ben), (13,enj), (14,nja), (15,jam), (16,ami),(17,min),(18,in$), (19,n$$)}. If one ignores the positional fields in the examples, there are a total of 12 sub-strings in common, and the positions of the corresponding match q-grams are shifts either forward by 9 positions or backward by 9 positions. This illustrates that the positional q-gram for string similarity join by its name involves the position comparison of matching q-grams after some positional shifts. Since 12 sub-strings in common are greater than [17 + 3 - 1] – 3*3 = 8 and less than [17 + 3 – 1] – 3*2 = 11, these two strings are taken to be similar when the threshold of edit distance is defined as 3.

A Q-gram has three significant properties which are count filtering, position filtering and length filtering.

- Count filtering is to compare the two sets of Q-gram if they are within the small edit distance k without considering the position.
- Position filtering is to count the difference of the same Q-grams from two sets if the difference is not more than k positions.
- Length filtering is to compare the length of two set strings if their difference is not more than k.

The intuition behind the count filtering is if two strings are within a small edit distance, they will have a large number of Q-gram in common. The position filtering restricts in k position to avoid mismatching Q-gram. The length filtering quickly eliminates strings which are not within the desired edit distance.

SELECT $R_1.A_0, R_2.A_0, R_1.A_i, R_2.A_j$

FROM $R_1, R_1A_iQ, R_2, R_2A_jQ$

WHERE $R_1.A_0 = R_1A_iQ.A_0$ AND

$R_2.A_0 = R_2A_jQ.A_0$ AND

$R_1A_iQ.Qgram = R_2A_jQ.Qgram$ AND

$|R_1A_iQ.Pos - R_2A_jQ.Pos| \leq k$ AND

$|strlen(R_1A_i) - strlen(R_2A_j)| \leq k$

GROUP BY $R_1.A_0, R_2.A_0, R_1.A_i, R_2.A_j$

HAVING COUNT(*) $\geq strlen(R_1A_i) - 1 - (k-1)^* q$ AND

COUNT(*) $\geq strlen(R_2A_j) - 1 - (k-1)^* q$

Figure 3-3: Q-gram Similarity Join in SQL Version

Figure 3-3 described in [17] shows three filters in the SQL expression. The approach augments the database with positional Q-grams corresponding to the original database strings. The augmented information is stored in an

auxiliary table such as $R_1A_iQ(A_0, Pos, Qgram)$ and $R_2AjQ(A_0, Pos, Qgram)$ with three attributes (). However, the auxiliary table is very expensive as a join predicate. The approach made very subtle changes to improve data processing time as shown in the following Figure 3-4. Our version has added an inexpensive UDF invocation edit_distance($R_1.A_i$, $R_2.A_j$, k) to directly filter the data without calculating the length in the where clause. The UDF invocation is in the Having clause which likely happens on just a small fraction of all possible string pairs so the performance is better than using length filter in the Where clause of the SQL.

SELECT	$R_1.A_0, R_2.A_0, R_1.A_i, R_2.A_j$
FROM	$R_1, R_1A_iQ, R_2, R_2A_jQ$
WHERE	$R_1.A_0 = R_1A_iQ.A_0$ AND
	$R_2.A_0 = R_2A_jQ.A_0$ AND
	$R_1A_iQ.Qgram = R_2A_jQ.Qgram$ AND
	$\mid R_1A_iQ.Pos - R_2A_jQ.Pos \mid \leq k$
GROUP BY	$R_1.A_0, R_2.A_0, R_1.A_i, R_2.A_j$
HAVING	COUNT(*) $\geq strlen(R_1A_i) - 1 - (k-1)^* q$ AND
	COUNT(*) $\geq strlen(R_2A_j) - 1 - (k-1)^* q$ AND
	editDistance $(R_1A_i, R_2A_j, k) \leq k$

Figure 3-4: Revised Q-gram Similarity Join in SQL Version

Obviously, either original SQL version or our revised SQL version uses only the database feature but the implementation requires a certain amount of the temporary table spaces to hold auxiliary Q-gram records on the fly. When the dataset becomes large, it needs large amount of temporary memory to hold the Q-gram data so it is very expensive to utilize any SQL version on a large

dataset. Our second implementation shown on the Figure 3-5 uses a high-level programming language to calculate Q-gram distance on the fly. The Q-gram calculation has taken into consideration three filters just like the SQL version. By calculating Q-gram distance on demand, the approach not only saved significant amount of temporary table space but also avoided heavy Cartesian join. Therefore, there is no need for building auxiliary tables to hold Q-gram and the performance has been tremendously improved.

qgramDistance (s,t,k)

set $|s|$ to be the length of source string s

set $|t|$ to be the length of target string t

set $len\_\max = \max(|s|,|t|)$

set distance value $dq = 0$ and q value as desired

if $||s|-|t||> k$ return false exit// length filtering

for $len\_\max$

form qs with ps as position in s

form qt with pt as position in t

if $qs = qt$ dq will be increased by 1

else record qs to nqs as non qs and qt to nqt as non qt

for $qs \in nqs$ and $qt \in nqt$

if $qs = qt$ dq will be increased by 1

if $|ps - ps|> k$ return false exit //position filtering

if $dq < (\max(|s|,|t|) + k - 1) - kq$ return false exit //count filtering

return true

Figure 3-5: Q-gram Distance Function in High-Level Language Version

In the above implementing algorithm, the number of q-gram for s is $|s|+q-1$ and the number of q-gram for t is $|t|+q-1$. The complexity for Q-gram

distance will be $\max(|s|+q-1,|t|+q-1)+\max(nqs,nqt) \cong 2*\max(|s|,|t|) \rightarrow O(\max(n|s|,|t|)$ which has much better complexity as Edit Distance does. It also increases true positive results and reduces false negative results. This new approach can be implemented in any high-level language. It is quite easy to be adapted to other distance metrics in the database domain.

3.3 Summary

Edit Distance is a common and fundamental distance metric for string similarity joins. It can be computed in $O(|s|x|t|)$ time and $O(\max(|s|,|t|))$ space via a standard programming approach. Q-gram distance is a very effective and widely used distance metric for approximate string matching. The newly proposed Q-gram implementation has overcome temporary memory space in SQL implementation. The complexity of Q-gram distance metric in $O(\max(|s|,|t|))$ is better than the complexity of edit distance metric in $O(|s|x|t|)$. In this dissertation, the approach is using Edit Distance and Q-gram Distance in the experiments to verify the pre-processing funding in the following chapters.

CHAPTER 4

ATTRIBUTE-BASED CLUSTERING APPROACH

String is a primary data format in a majority of applications. With the rapid growth of diverse data driven applications in the current digital world, retrieving such data from different structured sources becomes more and more significant and challenging as described earlier in this dissertation. It is true that all existing approaches have made an assumption that join predicates are always the optimal predicates. It will not matter what similarity join metrics are chosen and all of those approaches are applied on known join fields and don't consider the relationship between attributes. In reality, known or pre-defined join attributes might not give a desired or accurate result. In this chapter a pre-processing approach is proposed by combining a traditional clustering algorithm [FK 99] with a distance metric algorithm on the relational database. In the following each section is going to step through the works, which were developed by utilizing well-known edit distance metric and Q-gram distance metric as the evaluation methods.

4.1 Introduction

In the current digital information world, the more diverse applications are introduced to the world, the more backend databases are used, the more data integration is required, and the more similarity joins are needed. The primary data format for data integration is a *string.* The integration of string data is of central interest for many database integration applications, such as semantic query processing, data warehousing, data mining, and web searching. Dealing

with data dirty is a fundamental task in data integration applications. Similarity join has been used to handle data dirty and data identification. There are many possible join predicates besides the known join predicate. Identifying optimal similarity join predicates is the central focus of this research.

Example 1: assume hospital sources exist with name, age, address, and telephone as attributes; one source contains about 25K patient visits records and another has about 24K patient satisfaction survey records. The natural equijoin on one attribute produces about 15K records. The natural join with like statement on one attribute produces about 3k records. The natural equijoin on two attributes returns about 9K records. Similarity join on a single attribute returns about 18K records when the threshold is 1.

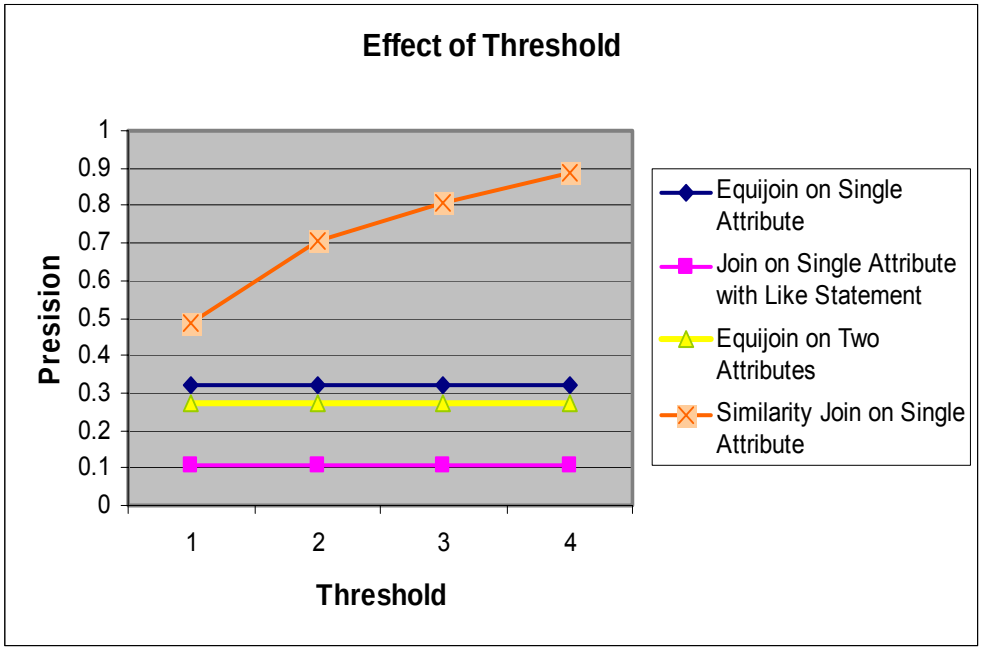


Figure 4-1: Commercial SQL Join vs Similarity Join

Figure 4-1 shows a natural equijoin on two attributes provides less precision than equijoin on a single attribute; equijoin on a single attribute

provides much higher precision than join on a single attribute with like statement; and similarity join provides better precision than any commercial SQL joins.

Figure 4-2 shows there are 18K, 20K, 15K, and 22.5K records returning when the same similarity join algorithm is applied with {name}, {name, address}, {name, address, telephone}, {name, telephone} respectively when the threshold is equal to 1.

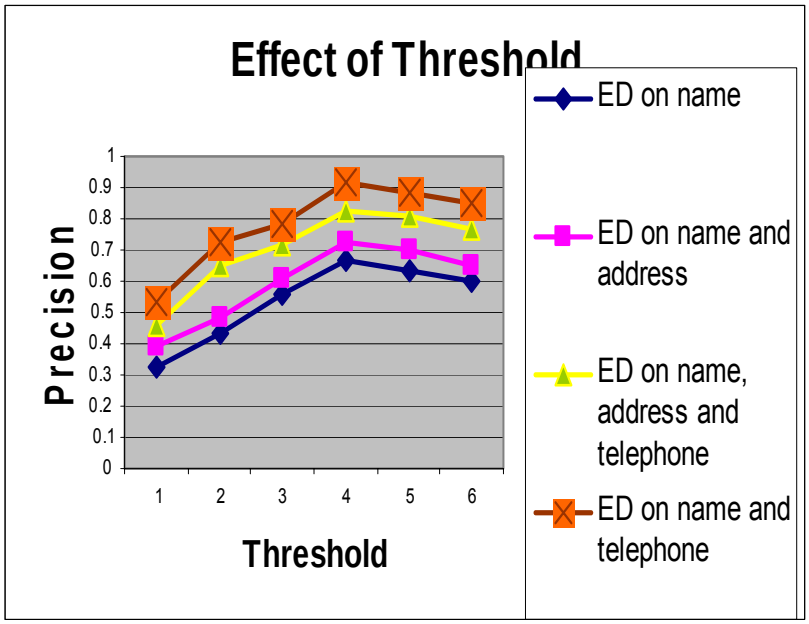


Figure 4-2: Similarity Join on Different Group Attributes

From the preceding experiments found that a commercial product couldn't return the person record if the join is based on name and the name is recorded in the patient visiting data source as Tom Hanks and in the survey data source as Ton Hamks when the threshold is equal 2. However, the similarity join was able to pick up Tom Hanks as one of the join result sets when the threshold was equal to 2. In this case, the precision of similarity join was better than natural join. In addition, applying similarity join on a single attribute might not produce the

desired result sets, and applying similarity join on multiple attributes might give much better result sets. The question arising from here is how the approach can find the best combination of the join attributes to produce more promising join result sets.

This chapter includes a pre-processing approach to improve existing string similarity join algorithms by grouping the attributes which are commonly used together in querying the data sources. The main contributions of this approach are a). To the best of our knowledge, a clustering-based pre-processing approach identifying clustered join attributes is the first attempt to use grouped attributes in the context of a relational DBMS for an approximate string match join. With an adequate threshold, the approach can reduce the number of false negatives and the number of false positives. b) The research studied challenges on applying proposed approach, including using user defined functions (UDFs) versus strict SQL and popular edit distance versus Q-grams [18]. c) This dissertation conducts a thorough experimental evaluation of the approach. The experimental results show that the technique can improve overall precision, recall, and F-measure on similarity join.

The remainder of this chapter is organized as follows. In the next section, a vertical clustering technique is described to identify groups of clustered related attributes and with corresponding experimental results; and in section 3 a clustering technique is presented to identify groups of clustered related attributes and a comparison of experimental results. Finally, section 4 summarizes the

research, describes some weaknesses to the approach, and suggests directions for refinements to the approach used in this research.

4.2 Attribute-based Clustering Approach

In this section, this dissertation describes a pre-processing method to identify join attributes which are closely related based on the number of times they are accessed together. The research refers to such group of closely related join attributes as clustered join attributes. The experiment will show that using the proposed pre-processing step, the precision of existing similarity join algorithms will be improved significantly.

4.2.1 Identifying Clustered Join Attributes

To identify clustered join attributes, the approach considers the usage of the attributes with respect to various applications. This research proposes to use an attribute clustering approach. The rationale behind the attribute clustering approach [Bez 81, HBV 02] is to produce fragments, groups of attribute columns that "closely match" the requirements of applications.

Given a relation R with attributes $A_1,...,A_b$ denoted by $R(A_1,...A_b)$, the approach uses the existing Bond Energy Algorithm (BEA) [4] as the attribute clustering approach to identify partitions as $R_1, R_2,..., R_r$, each of them containing a subset of R's attributes.

Let $Q = \{q_1,...,q_q\}$ be the set of user applications (queries) that will run on relation R. For each application q_k and each attribute A_j, the approach associates an attribute usage value, denoted as $use(q_k, A_j)$, and defined as:

$$use(q_k, A_j) = \begin{cases} 1 \text{ if attribute } A_j \text{ is referenced by application } q_k \\ 0 \text{ otherwise} \end{cases}$$

Table 4-1 shows the weight of application frequencies $acc(q_k)$, defined as how often each application q_k accesses attributes on the relation R.

Table 4-1: Attribute Usage Matrix

	Name	Phone	Birthday	Address	Access Freq
q_1	1	1	0	0	35
q_2	1	0	1	0	30
q_3	1	1	1	0	35
q_4	0	0	0	1	25

Affinity indicates how attributes are related based on attribute usages. Attribute affinity measures the bond between two attributes of a relation according to how they are accessed by applications and is defined as $aff(A_i, A_j) = \sum_{k|use(q_k, A_i)=1 \wedge use(q_k, A_j)=1} acc(q_k)$ where attribute affinity $aff(A_i, A_j)$ is the summation of access frequencies $acc(q_k)$ for all queries referring to attributes A_i and A_j.

Table 4-2: Attribute Affinity Matrix

	Name	Phone	Birthday	Address
Name	100	70	65	0
Phone	70	85	35	0
Birthday	65	35	65	0
Address	0	0	0	25

For example, in Table 4-1, both q_1 and q_3 refer to name attribute and phone attribute so $aff(A_{name}, A_{phone}) = 35 + 35 = 70$ as shown in Table 4-2.

The result of $aff(A_i, A_j)$ computation forms a nxn matrix called *attribute affinity matrix* (AA) where n is the number of attributes in the relation R. Table 4-2 shows a AA calculated from the value in Table 4-1. The research then uses BEA to find some means of grouping the attributes of a relation R based on the attribute affinity values in AA. BEA takes a AA matrix, permutes its rows and columns and generates a *clustered affinity* matrix (CA). As Table 4-3 shows, the purpose of this permutation is to maximize the global affinity measure and results in the grouping of large affinity values with large affinity attributes and small affinity values with small affinity attributes.

Table 4-3: Clustered Affinity Matrix

	Address	Birthday	Name	Phone
Address	25	0	0	0
Birthday	0	65	65	35
Name	0	65	100	70
Phone	0	35	70	70

When you look at Table 4-3 closely, the CA has three possibilities to split the set of attributes into two clustered fragments. To identify the best clustered attribute fragment from the CA, the approach is to compute split quality (SQ) based on the access model. The SQ is defined as

$SQ = af(VF_1) * af(VF_2) - af(VF_1, VF_2)^2$ where $af(VF_1)$ stands for access frequency for vertical fragment $af(VF_1)$ and $af(VF_1, VF_2)$ stands for access frequency for vertical fragment VF_1 and vertical fragment VF_2. For instance, in Table 4-3, for the first possible split {Address} and {Birthday, Name, Phone}, $SQ = 25 * (30 + 35 + 35) = 62500$; for the second possible split {Address, Birthday} and {Name, Phone}, $SQ = -(30 + 35)^2 = -4225$; for the third possible split {Address, Birthday, Name} and {Phone}, $SQ = -35^2$. The best clustered attribute group will be the cluster which produces a positive contribution for the good cases and a negative for the bad cases. This approach can be described in a high level in Figure 4-3.

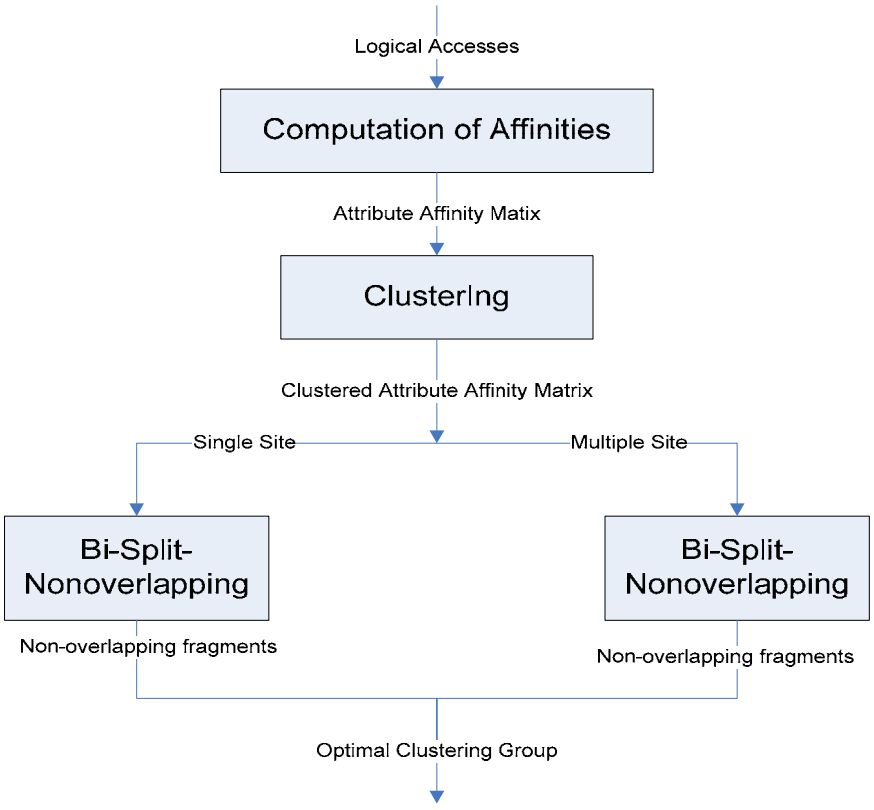


Figure 4-3: Attribute-Based Clustering Approach

In Figure 4-3, the core of the attribute-based clustering approach is presented and is to produce a clustered attribute affinity matrix and partition attributes as non-overlapping groups. The high level of clustering algorithm is shown in Figure 4-4.

```
clusteringBEA(AA)
    Set b to be the number of row in AA
    Initialize a bxb cluster attribute matrix CA
    Assign column 1 of AA to column 1 of CA
    Assign column 2 of AA to column 2 of CA
    Set Index = 3;
    While index <= b do
        For (i=1; i<=index-1; i++)
            Calculate contribution cont(A_{i-1}, A_index, A_i);
        cont(A_index, A_index, A_i)
        loc = position of the corresponding maximum cont
        for (j=index; j<=loc; j--)
            copy CA(., j-1 column to CA(.,j) column
        copy index column AA(.,index) to CA(.,loc) column
        index++;

    swapping the rows according to the relative ordering of columns
    return CA
```

Figure 4-4: BEA Clustering Algorithm

The above detail pseudo-code for the Clustering approach contains three main blocks. The first one is the initialization which places and fixes one of the columns of attribute affinity matrix arbitrarily into the clustered affinity matrix. The second one is the iteration which picks each of the remaining $b-i$ columns and

tries to place them in the remaining $i+1$ positions in the CA matrix, chooses the placement that makes the greatest contribution to the global affinity measure and continues this step until no more columns remain to be places. The last one is the row ordering which places the rows matching the relative position of the columns.

The complexity of implementing BEA Clustering approach shown in the preceding clusterBEA Algorithm is $O(b^2)$, where b is the number of attributes in the relation database table. Since the clustering approach is an attribute-based approach, the clustering group will not be changed when if the value of attributes gets changed or more records are added to the dataset.

biSplitNonoverlapping (CA)

Set n to be the number of row in AA

Set SQ to have a size $b-1$ array

$Pos = 1, \max P = 1$

While $Pos < b$ do

Calculate $af(VF_{1...Pos})$ and $af(VF_{Pos...n})$

Calculate $SQ[Pos-1] = af(VF_{1...Pos}) * af(VF_{1...Pos}) - af(VF_{1...Pos}, VF_{Pos...b})$

$Pos = Pos + 1$

$Pos = 1$

While $Pos < b$ do

Identify the biggest $\max(SQ_{Pos})$ value and set $\max P$

Return $\max P$

Figure 4-5: Binary Split Algorithm

In Figure 4-3, the clustering output will be the input of binary split non-overlapping method. Figure 4-5 shows a high level of the split approach. There are $b-1$ possible ways to split the CA matrix along the diagonal, where n is the

size of the CA (i.e., the number of attributes in the table). The complexity of the split approach is $O(b)$. Therefore the final complexity for the attribute-based approach is $O(b^2)$. In general, the approach is quite inexpensive since the number of attributes is much smaller than the number of records in the dataset.

4.2.2 Similarity Join on Attribute-Based Clustering Fields

Figure 4-6 shows a high-level approach from application access information to any general similarity join.

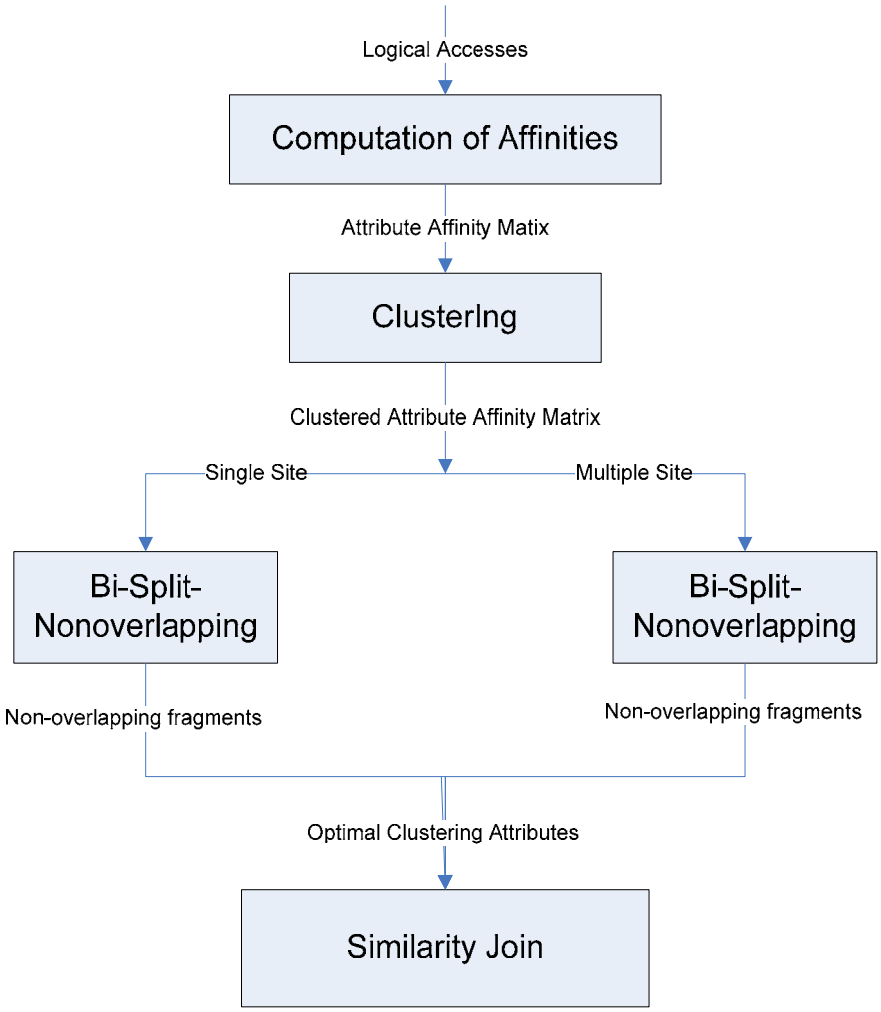


Figure 4-6: Clustering Attributes and Similarity Join

The objective of the attribute-based clustering approach is to improve the outcomes and the performance of existing similarity joins by using identified optimal clustering attributes as join predicates. When edit distance metric is used to measure join predicates, for any given relational tables R_1 with n_1 number of records and R_2 with n_2 number of records, the ED similarity join approach will measure each record as s on R_1 against each record as t on R_2 via the editDistance(s,t) described on Figure 3-2. The complexity of this approach is $O(n_1 * n_2 * |s| * |t|)$.

Q-gram is another distance measurement that the approach will apply on identified optimal clustering attributes to further improve the precision and the performance of the results. The Q-gram distance similarity join is similar to edit distance similarity join instead of calling an editDistance function, but calling qgramDistance(s,t,k) function to compare a source string s from one source R_1 with n_1 number of records to a target string t from another source R_2 with n_2 number of records. The complexity of this approach is $O(n_1 * n_2 * \max(|s|,|t|))$ which is better than $O(n_1 * n_2 * |s| * |t|)$ from edit distance similarity join approach.

4.3 Experimental Evaluation

The previous section described how to identify clustering attributes. This section demonstrates the performance improvement on two of existing similarity join approaches by using identified clustering attributes as join predicates. The two similarity join metrics are edit distance and Q-gram distance, which were introduced in the previous chapter.

4.3.1 Environment Setup

[CHS 07] indicated there is no common benchmark dataset on similarity join. [LPSL 10] used one or two attributes from public datasets to evaluate their approaches. All the data used in this experiment were from a real dataset that consists of university student information. In the development of the string similarity join, the relation R_1 consists of 980,000 data items, 60% of a student data set, and the relation R_2 contains 50,000 data items, 80% from the whole employee dataset. The attributes in the dataset are name, address, telephone, and birthday. The approach tries to retrieve all distinct pairs of records $(r_1, r_2) \in (R_1, R_2)$ such that the error rate between the corresponding fields of strings are less than or equal to the given k threshold value. The research developed Java and PL/SQL applications on the identified join attributes in the relation R_1 and R_2. The experiments were performed on a Sun Solaris 9 OS system and Oracle 9i database with 900MHz 2 CPU and 4GB memory. The experimental results were very consistent on multiple runs.

In the experiments, the approach evaluates the quality performance of retrieval records in terms of precision, recall, and F-measure. *Precision* is defined as the ratio of the number of relevant records retrieved by similarity join to the total number of retrieved records. *Recall* is defined as the ratio of the number of relevant records retrieved by similarity join to the total number of existing relevant records. It is very normal that when the threshold increases, both the true relevant records and false relevant records are increasing but false irrelevant records are decreasing so there is an inversion between precision and recall.

Therefore precision and recall are not usually discussed in isolation. F-measure is used to leverage between precision and recall. *F-measure* is defined as the weighted harmonic mean of precision and recall:

$$F = 2 * (precision * recall) / (precision + recall)$$

4.3.2 Edit Distance on Attribute-based Clustered Attributes

In this experiment, the approach applied ED on all possible combination attributes. In Figure 4-7, the results in terms of the precision performed by edit distance computation on different attributes for different k threshold values are presented. This experiment indicates that not all multiple attribute joins can produce more relevant records than a single attribute join. From the above definition of precision, the higher the precision rate is, the less false positive records are in the returned dataset, and the more the returned data consists of relevant data.

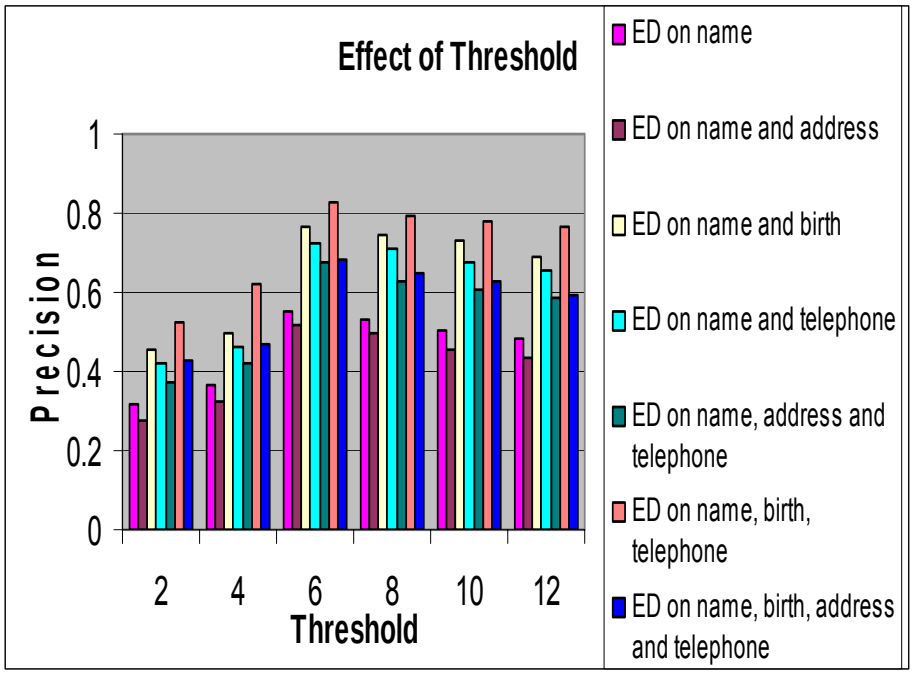


Figure 4-7: ED Precision on Attribute(s)

Figure 4-8 shows the results in term of recall performed by the edit distance on different attributes with different k values. Based on the above definition of recall, the higher the recall rate is, the less false negative records are in the returned dataset, and the more relevant data is on the returned dataset.

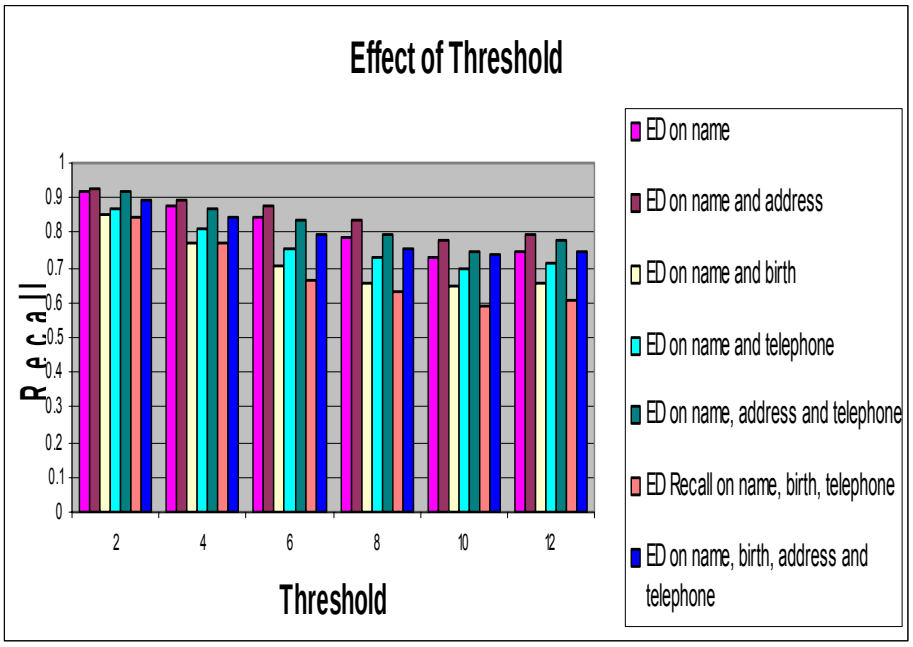


Figure 4-8: ED Recall on Attribute(s)

The above experimental result has shown the best combination attributes are name, birth, and phone which match the result of clustered join attributes calculated from the application access frequencies and attribute usage gathering from the applications in the previous section.

4.3.3. Q-gram Distance on Attribute-based Clustered Attributes

In this experiment, the approach applied Q-gram on all possible combination attributes. In Figure 4-9, the results in terms of the precision performed by Q-gram distance computation on different attributes for different k threshold values are presented. This experiment also indicates that not all

multiple attribute joins can produce more relevant records. The result shows the higher the precision rate is, the less false positive records are in the returned dataset, and the more the returned data consists of relevant data. That matches perfectly with the above definition of precision,

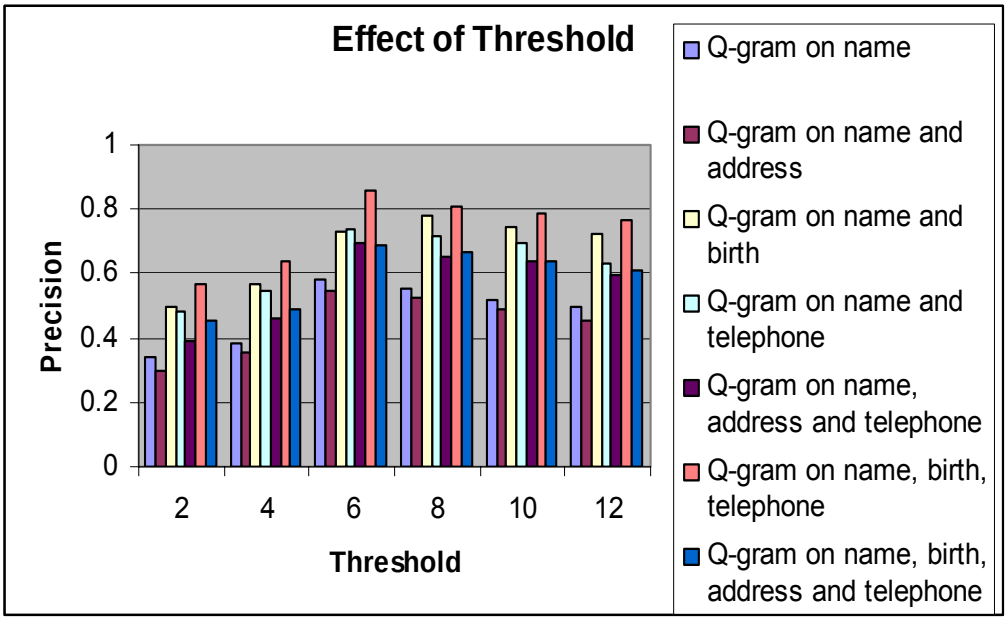


Figure 4-9: Q-gram Precision on Attribute(s)

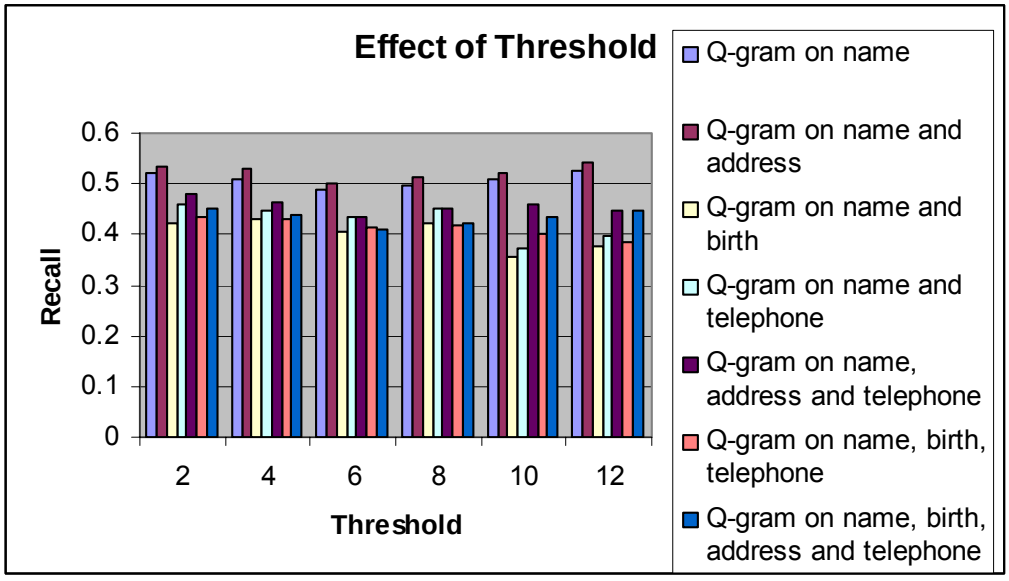


Figure 4-10: Q-gram Recall on Attribute(s)

Figure 4-10 shows the results in term of recall performed by the Q-gram distance on different attributes with different k values. The results show that the higher the recall rate is, the less false negative records are in the returned dataset, and the more relevant data is on the returned dataset.

4.3.4. ED vs. Q-gram Distance on Attribute-based Clustered Attributes

The preceding experimental result has shown the best combination attributes are name, birth, and phone which match the result of clustered join attributes calculated from the application access frequencies and attribute usage gathering from the applications in the previous section.

Figure 4-11 shows the precisions of Q-grams and edit distance on identified clustered join attributes which are name, birth, and telephone number. This figure shows that using Q-grams could return about 10% more relevant records on average, using different threshold values, than using edit distance.

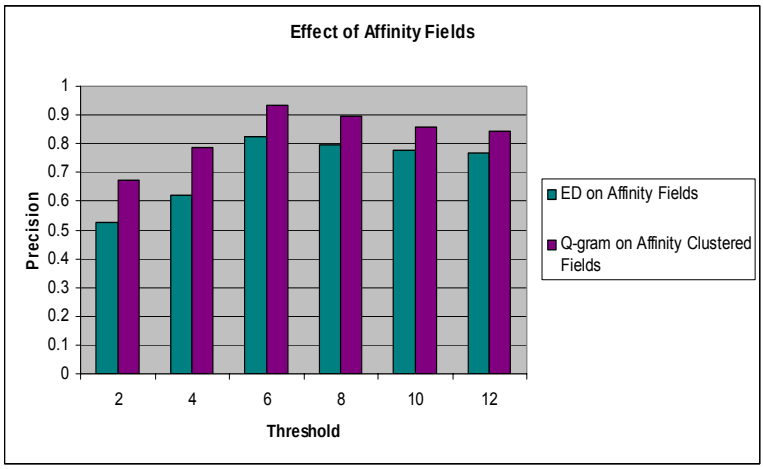


Figure 4-11: ED and Q-gram Precision on Affinity Clustered Attributes

Figure 4-12: shows the results in terms of recall performed by the edit distance on name, birth, and telephone attributes and Q-gram on name, birth,

and telephone attributes, with different k threshold values. The results show that using Q-grams on the affinity clustered join attributes produces about 5% less relevant records, on average, for different k threshold values, than using edit distance on the affinity clustered attributes.

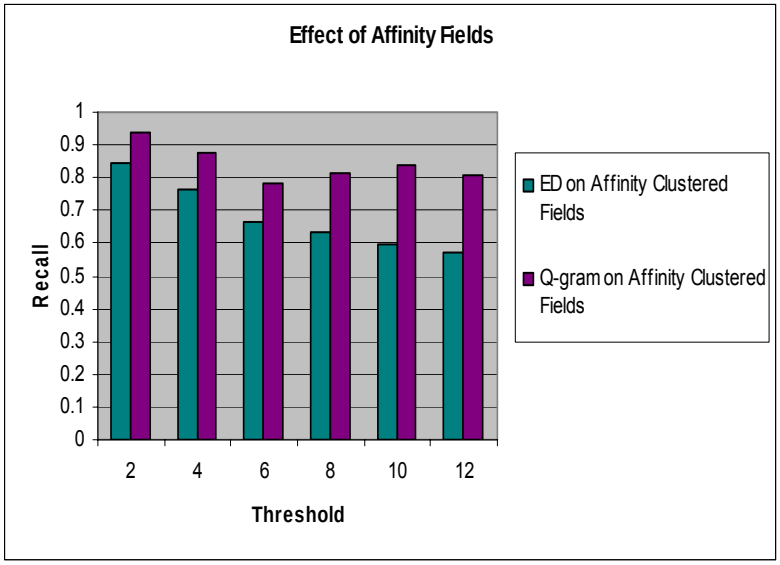


Figure 4-12: ED and Q-gram Recall on Affinity Clustered Join Attributes

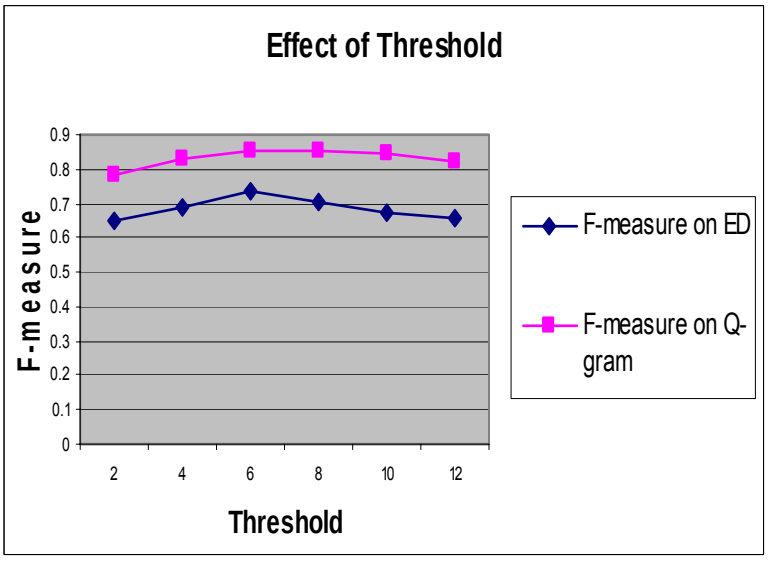


Figure 4-13: ED and Q-gram F-measure on Affinity Clustered Attributes

Based on the combined measure of recall and precision, Figure 4-13 shows the F-measure results for Figure 4-11 and Figure 4-12. These results show that Q-grams on clustered affinity attributes have about 15% better F-measure than Edit Distance on clustered affinity attributes. The results indicate that the Q-grams clustered approach can produce more relevant results than ED clustered approach.

4.4 Summary

In this chapter, the research determined how to find groups of related fields to improve the performance of existing similarity join methods.

This dissertation introduced a pre-processing approach to take into consideration groups of related fields through the well-known Energy Bond Clustering algorithm. The pre-processing approach was applied to existing similarity join algorithms. The experiment showed some promising performance improvements over existing algorithms using F-measure.

The current proposed pre-processing method is highly dependent on knowledge of the data, which sometimes is not possible to have in advance. In the next chapter, this dissertation will introduce the second method to identify clustered related attributes.

CHAPTER5

VALUE-BASED CLUSTERING APPROACH

5.1 Introduction

The previous chapter introduced a pre-processing approach for similarity join techniques that takes into consideration groups of clustered related attributes through the well-known Bond Energy Clustering Algorithm (BEA). The rationale behind BEA, a vertical partitioning method, is to produce fragments, groups of attribute columns that closely match the requirements of transactions. In the literature, the term *clustering* has been used to refer to non-disjoint fragments. Since BEA relies on the requirements of multiple transactions, information on the transactions of the applications has to be predefined to use this approach. In reality, it is likely impossible to obtain this knowledge when the join has to take place. Also the previous approach treats the data as either fully dependent or fully independent. However, this assumption is too restrictive for real-world applications.

Clustering [Bez 75] is a mathematical approach that attempts to discover structures or certain patterns in a data set, where the data inside each cluster show a certain degree of similarity. The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data. Fuzzy clustering [BP 92] is a process of assigning membership levels and using them to assign data to one or more clusters. It allows data to belong to more than one cluster with different memberships (between 0 and 1) and vague or fuzzy boundaries between

clusters. That indicates the strength of the association between the data and a particular cluster.

Fuzzy C-means (Bezeek 1981) is one of the most widely used fuzzy clustering algorithms. The FCM algorithm attempts to partition a finite collection of n data $X = \{x_1,...,x_n\}$ into a collection of C fuzzy clusters with respect to some given criterion. In the approach, it utilizes fuzzy C-means to divide data into clusters so that data in the same class is as similar as possible, and data in different classes are as dissimilar as possible. To utilize Fuzzy C-means, the proposed approach is to use a popular edit distance, which maps numerical data to categorical data, to produce the dissimilarity values between all records in between relational database tables on columns or groups of columns; runs multi-dimensional scaling on each dissimilarity matrix and generates a numerical array for every record, applies a fuzzy clustering procedure to determine the best cluster structure on each group of attributes, and then uses a greedy method to determine the best clustering of attributes.

The remainder of this chapter is organized as follows. Section 2 gives an overview of the previous approach related to clustering similarity join. Section 3 describes a new proposed approach. Section 4 presents some experimental results. Section 5 concludes the paper and suggests potential future research.

5.2 Related Work

As the previous work indicated, one of the serious concerns arising in string matching problems is how to identify string records across different data sources that refer to the same entity.

Jin [JL 05] presented a novel technique, called Selectivity Estimation of Approximate PredIcAtes (SEPIA). The approach is to solve the problem of estimating the selectivity of fuzzy (approximate) string predicates for query optimizers and supports fuzzy queries. For instance, consider a query with two predicates like *name similar to Campbell and salary > 50,000.* If there are many records that satisfy the first predicate and only a few satisfy the second, processing the second predicate first might be a good choice. The SEPIA uses the set concept to avoid the processing order issue on the existing fuzzy query technique. It groups string datasets into subsets called clusters via known horizontal clustering approaches, builds a histogram structure for each cluster, and constructs a global histogram for the database. It is based on the intuition: given a query string q, a preselected string p in a cluster, and a string s in the cluster, based on the proximity between q and p, and the proximity between p and s, one can obtain a probability distribution from the global histogram about the similarity between q and s.

Lieberman [LSS 08] presented an approach, executing on a Graphics Processing Unit (GPU), exploiting its parallelism and high data throughput. As GPUs only allow simple data operations, such as the sorting and searching of arrays, the approach uses these two operations to cast a similarity join operation as a GPU sort and search problem. Also, the approach processes each point p of the other dataset in parallel. This approach showed a good balance between time and work efficiency using the data structure.

Wu [WZZ 05]] presented an efficient and effective multi-database mining approach for classifying multiple databases. Indexing databases by features is a common technique for evaluating the relevance between different databases. The approach focuses on a transaction database and all items in a transaction database are used as features to index the database. If two databases share a significant number of common items, the two databases are relevant to each other. The approach has addressed effectively measurement on the relevance of database independent applications and search for the best database mining classification.

Zhang [ZWZ 03] presented a new multi-database mining process for helping analyze data in different sources. The process focuses on database clustering and local pattern analysis with searching for a good classification [Mac 67], identifying high-vote patterns and exceptional patterns, and synthesizing local patterns by weighting. The process has addressed some of the pressing issues on mining multi-database.

The approaches of Jin, and Lieberman are applied on string records under the assumption that the predicates, selection criteria, or join attributes are a known join condition. While the similarity approaches of Wu and Zhang focus on the multi-database mining. The solution proposed in this chapter complements existing approaches in such a way that groups of related attributes are computed and used as join attributes on database, so the overall F-measure and precision will improve.

The fuzzy paradigm [SW 85] has a much better fit in the scenario where there is no clear boundary between clusters. Since the target datasets are categorical in nature, to use the fuzzy set [Bez 74, Zah 65] paradigm, the approach have to find a way to transform categorical data to numerical data and have a way to map numerical data back to categorical data.

5.3 Proposed Work

Computing dissimilarity value between two categorical data can map categorical data to numerical data. The similarity measure controls how the numerical data is formed.

Value-based Clustering Approach:

Input: a data set R is concatenated from two union compatible or semi-compatible relational data sets
R_1 with n_1 records and R_2 with n_2 records
Total records of R is $n = n_1 + n_2$ with $R.A_1 ... R.A_k$ attributes
Output: a group of clustered related attributes

Initialize variables
Pick a common attribute to start
For each level
for each combination of attributes on each level
Step A: Convert categorical data to numerical data and form vectors using dissimilarity values
Step B: Map vectors to a lower-dimensional space using Sammon Mapping
Step C: Partition lower-dimensional vectors to fuzzy C-Means clusters
Step D: Calculate partition entropy (PE) and partition coefficient (PC)
End for
Step E: Evaluate PE and PC
Choose the best pair of PE and PC
Return the optimal clustered related attributes

Figure 5-1: Value-Based Clustering Approach

Figure 5-1 summarizes the proposed approach in 5 steps to transform categorical data to numerical data by using an edit distance approach to compute the dissimilarity values between all records of join attributes, apply a Sammon mapping method [RD 97, Sam 69] on dissimilarity values to map high dimensional vectors to lower dimensional vectors, partition [MNAA 07] the lower dimensional vectors to C-Means clusters using the fuzzy C-Means algorithm, and then select the best cluster set of join attributes by a greedy method.

The greedy approach is to evaluate PE and PC for each level and each combination of attributes, and then pick the best PE and PC on each level. This approach can significantly reduce the overall complexity from $O(n^n)$ to $O(n^2)$ where n is the number of attributes as the approach showed on the previous section. Since the fuzzy approach is based on the dataset, the re-computation is needed when the record is inserted into the table or the record is deleted from the table. The following sub-section will give more detail information on each of the above steps.

5.3.1. Transform Categorical Data to Numerical Data and form multi-dimensional vectors

To transform categorical data to numerical data, the approach works on a single attribute or a group of attributes to build a set R and form an nxn Dissimilarity Matrix as $DM[x_{ij}]_{nxn}$ by self-joining the attribute or a group of attributes, where the x_{ij} stands for the dissimilar value of the i^{th} row and j^{th} column in the nxn matrix. To calculate the difference between these records, this research proposed to use edit distance to measure their dissimilarities.

Example 1: Consider that a relation R_1 has 10 records and a relation R_2 has 10 records with attributes name, primary email, alternative email, address, phone and birthday. After concatenating them, the approach has a new relation which has 20 records. For simplicity, the approach use bottom up approach and start from the name attribute because people normally can be identified by their name. After computing all the dissimilar value between 20 records, the approach produces a 20 by 20 symmetric dissimilarity matrix. The approach shows it as a vector form in Figure 5-2.

$$x_{1j} = \{0, 21, 18, 19, 18, 18, 17, 18, 20, 19, 19, 19, 15, 18, 20, 18, 19, 20, 18, 19\},$$
$$x_{2j} = \{21, 0, 16, 11, 13, 12, 14, 19, 11, 10, 19, 13, 18, 13, 15, 18, 14, 13, 17, 14\},$$
$$x_{2j} = \{18, 16, 0, 14, 15, 13, 14, 13, 14, 14, 18, 13, 18, 13, 14, 15, 16, 14, 15, 16\},$$
$$x_{4j} = \{19, 11, 14, 0, 14, 10, 13, 18, 11, 10, 19, 11, 18, 13, 15, 16, 13, 12, 17, 14\},$$
$$x_{5j} = \{18, 13, 15, 14, 0, 13, 15, 18, 13, 12, 20, 14, 17, 13, 16, 18, 15, 13, 15, 15\},$$
$$x_{6j} = \{18, 12, 13, 10, 13, 0, 14, 17, 12, 12, 17, 11, 15, 12, 15, 16, 13, 13, 16, 15\},$$
$$x_{7j} = \{17, 14, 14, 13, 15, 14, 0, 17, 14, 14, 17, 13, 19, 14, 15, 15, 16, 13, 18, 16\},$$
$$x_{8j} = \{18, 19, 13, 18, 18, 17, 17, 0, 19, 18, 17, 17, 20, 18, 13, 18, 19, 17, 18, 17\},$$
$$x_{9j} = \{20, 11, 14, 11, 13, 12, 14, 19, 0, 12, 18, 11, 19, 12, 14, 17, 14, 13, 16, 14\},$$
$$x_{10j} = \{19, 10, 14, 10, 12, 12, 14, 18, 12, 0, 18, 10, 19, 12, 17, 18, 14, 10, 18, 13\},$$
$$x_{11j} = \{19, 19, 18, 19, 20, 17, 17, 17, 18, 18, 0, 18, 17, 18, 20, 20, 18, 19, 19, 19\},$$
$$x_{12j} = \{19, 13, 13, 11, 14, 11, 13, 17, 11, 10, 18, 0, 18, 12, 15, 17, 15, 11, 18, 14\},$$
$$x_{13j} = \{15, 18, 18, 18, 17, 15, 19, 20, 19, 19, 17, 18, 0, 19, 19, 18, 18, 18, 19, 17\},$$
$$x_{14j} = \{18, 13, 13, 13, 13, 12, 14, 18, 12, 12, 18, 12, 19, 0, 17, 16, 15, 14, 16, 17\},$$
$$x_{15j} = \{20, 15, 14, 15, 16, 15, 15, 13, 14, 17, 20, 15, 19, 17, 0, 18, 16, 15, 18, 15\},$$
$$x_{16j} = \{18, 18, 15, 16, 18, 16, 15, 18, 17, 18, 20, 17, 18, 16, 18, 0, 16, 17, 18, 19\},$$
$$x_{17j} = \{19, 14, 16, 13, 15, 13, 16, 19, 14, 14, 18, 15, 18, 15, 16, 16, 0, 16, 17, 17\},$$
$$x_{18j} = \{20, 13, 14, 12, 13, 13, 13, 17, 13, 10, 19, 11, 18, 14, 15, 17, 16, 0, 18, 14\},$$
$$x_{19j} = \{18, 17, 15, 17, 15, 16, 18, 18, 16, 18, 19, 18, 19, 16, 18, 18, 17, 18, 0, 18\},$$
$$x_{20j} = \{19, 14, 16, 14, 15, 15, 16, 17, 14, 13, 19, 14, 17, 17, 15, 19, 17, 14, 18, 0\}$$

Figure 5-2: Symmetric Dissimilarity Matrix

5.3.2 Multi-Scaling Mapping

Multidimensional Scaling (MDS) [BG 97, MU 00, BMS 07] is a set of data analysis techniques that display the structure of similar or dissimilar data as a geometrical picture and help understand people's judgments of the similarity or dissimilarity of members of a set of objects. In this implementation, the set of objects on DM is defined as column vectors of an nxn matrix. The approach uses one of MDS techniques to detect meaningful underling dimensions to explain the observed dissimilarities between the investigated records of an attribute or groups of attributes.

Sammon mapping (Sammon 1969) [Sam 69] is one of the MDS algorithms and reveals the inherent structure of the data to explore the data, to find possible clusters, correlations or underlying distributions. The fundamental theory behind Sammon mapping is to consider a set of n dimensional vectors as $X = \{x_k \mid x_k = (x_{1k}, x_{2k}, ..., x_{nk})^T, k = 1,2,..., n\}$ and distances between the vectors as $d_{ij} = d(x_i, x_j), x_i, x_j \in X$ for the $DM[x_{ij}]_{nxn}$ matrix. Each vector is represented by one point in a n - dimensional space. The purpose of the Sammon mapping is to transform these n points into a lower, n-dimensional space $Y = \{y_k \mid y_k = (y_{1k}, y_{2k}, ..., y_{dk}), k = 1,2,..., d\}$ $(d < n)$ with distances between the output vector as $d^*_{ij} = d(y_i, y_j)$ where $y_i, y_j \in Y$, in such a way that the corresponding distances approximate the original ones as much as possible and the Sammon mapping minimizes the error function E as follows:

$$E = \frac{1}{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} d(i,j)} \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} \frac{(d(i,j) - d^*(i,j))^2}{d(i,j)}$$

Without loss of generality, the approach projects n dimensions into a 2-dimensional space for the purpose of data visualization. Continuing with the above example, the above DM_{20x20} matrix is the input of the Sammon mapping which maps 40x40 high dimensions to 20x2 lower dimensions as shown in Figure 5-3.

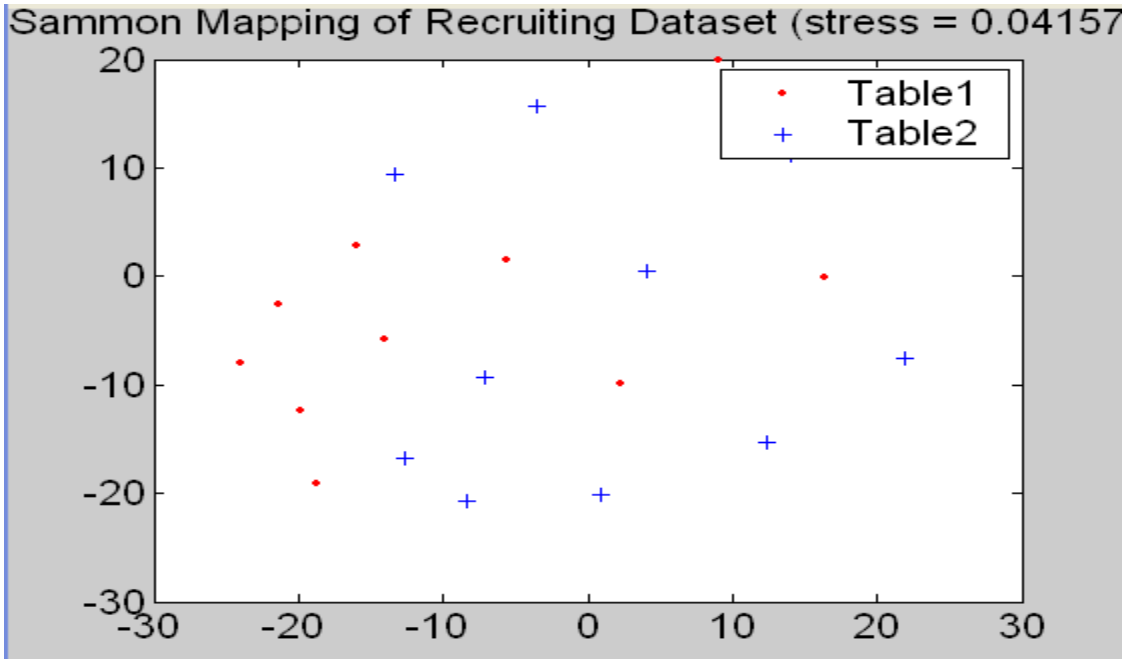


Figure 5-3: Sammon Mapping

5.3.3 Fuzzy Cluster Technique

Fuzzy clustering is used to classify datasets into related groups. In this approach, it has n two dimensional points. The process of clustering is to assign the n points into pre-defined clusters $\{c^{(k)} : k = 1,...,K\}$ by checking their closeness using the distance assignment principle.

The Fuzzy c-Means algorithm assigns points to clusters by the distance assignment principle, which assigns a new point to a cluster such that the distance from the point to the center of the cluster is the minimum over all c clusters. The fuzzy c-means clusters Sammon Mapping 20 by 2 results in Figure 5-3 and produces fuzzy c-means shown in Figure 5-4.

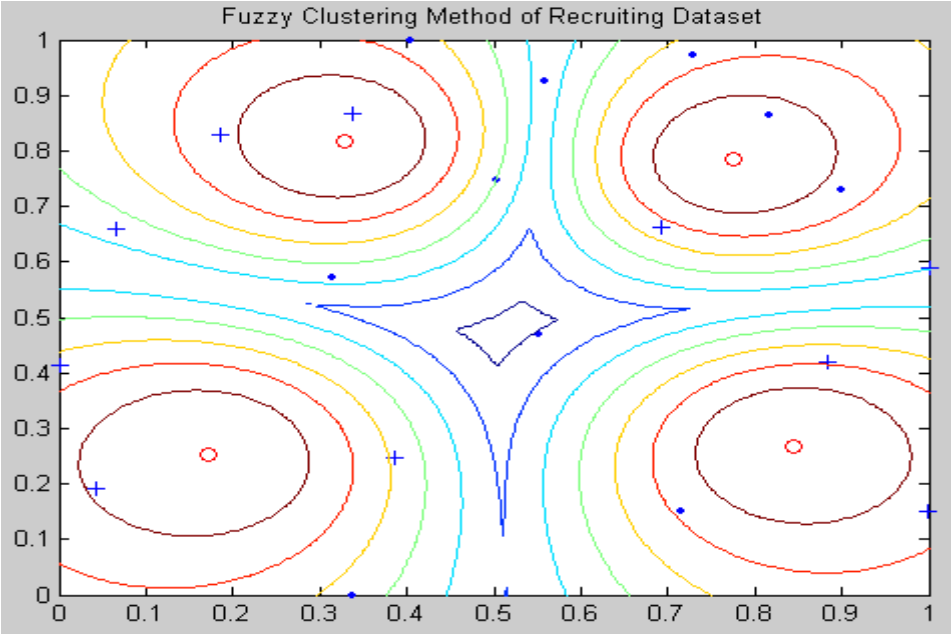


Figure 5-4: Fuzzy Clustering Results

5.3.4. Calculate Partition Coefficient and Partition Entropy

To evaluate the approach, it has group of attributes as candidates, the set of optimal attributes as predicate, partition coefficient and partition entropy as a selection and objective function. The approach uses a greedy concept to find a subset of group of attributes from a collection of candidates, where the subset must satisfy some specified criteria, such that the objective function is optimized.

Bezdek's partion coefficient (PC) [Bez 74]] is used to measure the amount of "overlap" between clusters. The partition coefficient of c, denoted by $PC(c)$,

produces an average of the squared values of the membership grades encountered in the partition matrix:

$$PC(c) = \frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{n}(u_{ij})^2$$

Where $u_{ij}(i = 1,2,...,c : j = 1,2,...,N)$ is the membership of data point j in cluster i

If each point belongs to a single cluster (hard partition), the partition coefficient assumes its maximal value of 1. If points share their membership across all clusters, with the same membership grade equal to $1/c$, this gives rise to the lowest value of $P(c)$ which in this case equals $1/c$. In other words, the coefficient quantifies the ambiguity of the partition matrices so that the approach can rank them and select the one with the lowest ambiguity.

Bezdek's partition entropy also satisfies the relation $0 <= 1 - PC(c) <= CE(c)$ for all cluster partitions c, where Partition Entropy is defined as

$$CE(c) = \frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{N}(u_{ij})\log(u^{ij}).$$

It is basically a measure for the fuzziness of the cluster partition, which is similar to the partition coefficient. The values of the partition entropy range from 0 to $\ln(N)$. If the approach considers Boolean entries of the partition matrix, the entropy is equal to 0. The highest value is obtained when there is a uniform distribution of membership grades (equal to $1/c$), which is $CE(c) = \ln(c)$.

While the partition coefficient and partition entropy exhibit interesting properties that are useful in quantifying the ambiguity of partition matrices. Both

of them provide information about the membership matrix without considering the data itself. The maximum values of the partition coefficient imply a good partition in the meaning of a least fuzzy clustering [XB 91]. The minimum values of the partition entropy imply a good partition in the meaning of a more crisp partition. The PC for the fuzzy c-means result in the previous section is 0.5919 and CE is 0.7871.

5.3.5 Select Best Set of Join Attributes

Generally, a greedy approach is any algorithm that makes the locally optimal choice at each stage with the hope of finding the global optimum. In this approach, it applies the greedy strategy to the same number of attribute sets to find the optimal path on the tree structure attribute sets. The top-down greedy approach is to choose a logical meaningful attribute as a starting solution set, add an optimal attribute to the solution set one at a time, calculate PE and CE for each set, select the best solution set with optimal PE and CE, keep adding the attribute until the optimal solution set is formed.

In other words, the greedy method involves finding a group of attributes which has an optimized PE and CE. Without using greedy method, for any number of attributes n, the approach starts with a single attribute. It has n possible groups the approach needs to evaluate. In the 2nd level, the approach will have $n-1$ possible groups to be evaluated. The method will continue adding the attributes until finishing so the complexity is $n(n\_1)(n-2)...+1=Q(n^n)$. Using greedy method, for any number of attributes n the approach has the complexity

as $n + (n\_1) + (n-2)... + 1 = Q(n^2)$. Therefore the greedy approach is less expensive than the previous approach.

Table 5-1a, 5-1b, 5-1c, and 5-1d show PC and PE values for different combination of group attributes. The partition coefficient values and partition entropy values exhibit interesting properties. These properties are useful in quantifying the ambiguity of partition matrices. Both of them provide information about the membership matrix without considering the data itself. The maximum values of the partition coefficient indicate a good partition in the meaning of a least fuzzy clustering. The minimum values of the partition entropy indicate a good partition in the meaning of a more crisp partition.

The *name* field is chosen as the starting point of the experiment. This is because the *name* field is the most meaningful identified field in all the databases. Figure 5-5a, 5-5b, 5-5c, and 5-5d show adding one attribute at a time, calculating their PEs and CEs, and picking the best combination of them as the approach moves to the next path, shown in the dashed arrow pointer. The approach iterated group of attributes until the approach found the best optimized group of clustered related attribute set.

Table 5-1a. PARTION COEFFICIENT vs CLASSIFICATION ENTROPY

	Name	Name Primary Email	Name Alternative Email	Name Address	Name Birthday	Name Phone
PC	0.4981	0.6238	0.5837	0.5537	0.5219	0.5347
PE	0.7689	0.6873	0.7456	0.7124	0.7639	0.7791

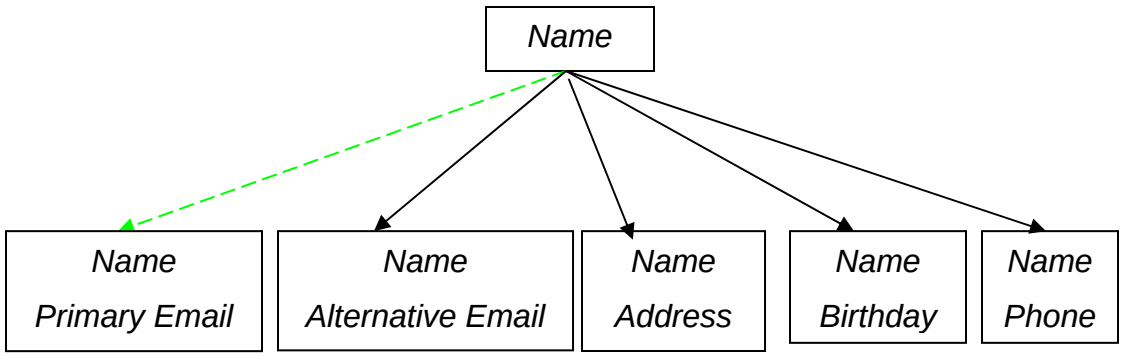


Figure 5-5a: Tree Structure

Table 5-1b. PARTION COEFFICIENT vs CLASSIFICATION ENTROPY

	Name Primary Email	Name Primary Email Alternative Email	Name Primary Email Birthday	Name Primary Email Address	Name Primary Email Phone
PC	0.6238	0.6534	0.7126	0.6428	06378
PE	0.6873	0.6739	0.6368	0.6617	0.6457

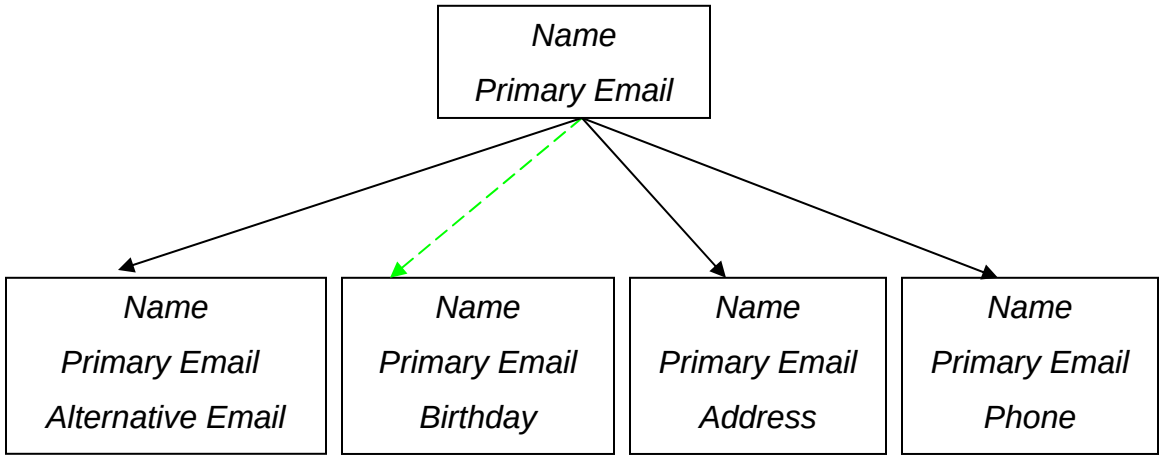


Figure 5-5b: Tree Structure

Table 5-1c. PARTION COEFFICIENT vs CLASSIFICATION ENTROPY

	Name Primary Email *Birthday*	Name Primary Email *Birthday* Alternative Email	Name Primary Email *Birthday* Address	Name Primary Email *Birthday* Phone
PC	0.7126	0.6826	0.6731	0.6548
PE	0.6368	0.6849	0.6578	0.7024

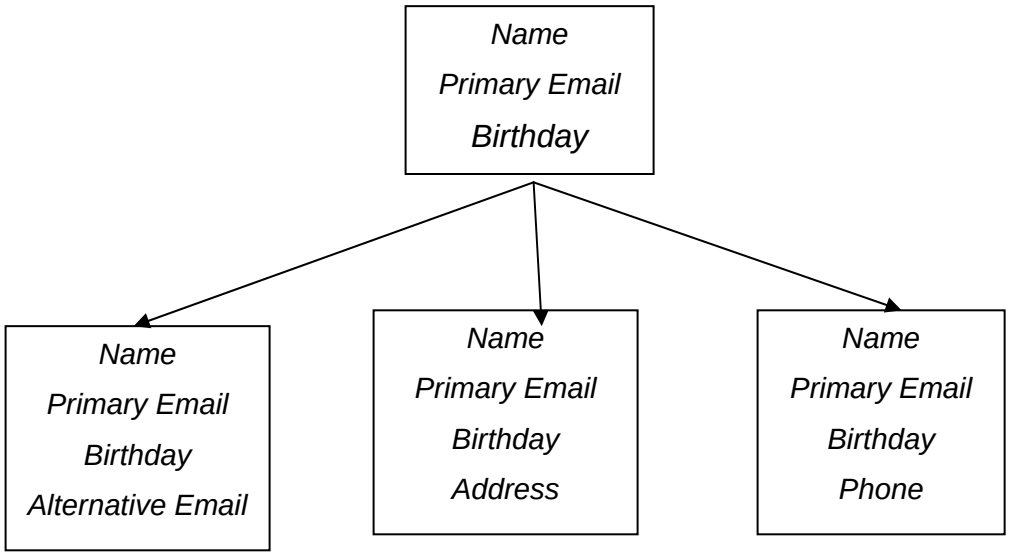


Figure 5-5c: Tree Structure

Table 5-1d. PARTION COEFFICIENT vs CLASSIFICATION ENTROPY

	Name Primary Email Birthday Alternative Email	*Name Primary Email Birthday Alternative Email Address*	*Name Primary Email Birthday Alternative Email Phone*	*Name Primary Email Birthday Alternative Email Birthday Phone*
PC	0.6826	0.6471	06559	0.6349
PE	0.6849	0.7256	07137	07429

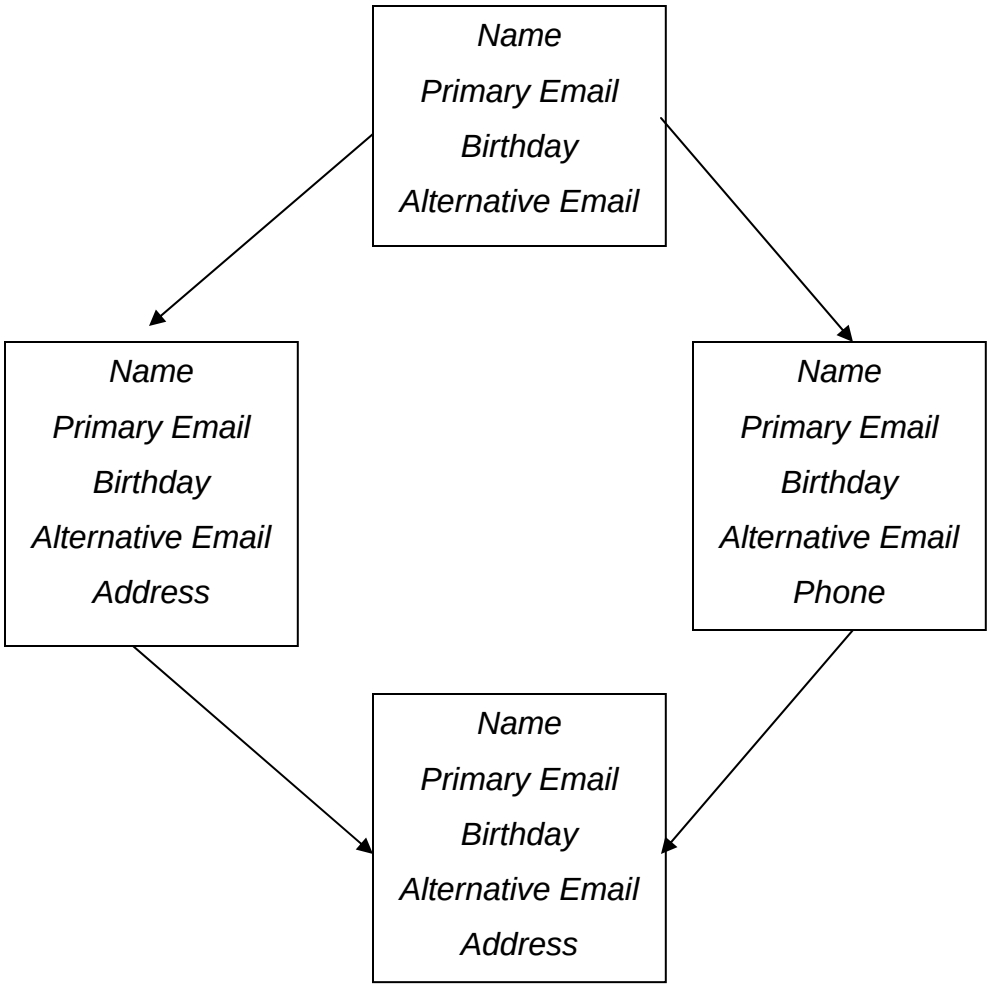


Figure 5-5d: Tree Structure

5.4 Experimental Evaluation

In the previous section, this dissertation described how to identify clustered attributes in detail using value-based clustering approach. In this section, this dissertation utilizes the identified clustering attributes as join predicates on two of existing similarity join algorithms. The approach is evaluated in term of precision, recall and F-measure.

5.4.1. Environmental Setup

As [CHS 07] indicated, there is no common benchmark dataset on similarity join. In this experiment, all the data were from a university student recruiting dataset. The data were integrated from two sources; one is student prospecting and recruiting information in a customer relationship management system that keeps track of prospecting and recruiting records stored in SQL server databases; the other was data in administrative information of returning students in an Oracle databases. In the development of the string similarity join, the relation R_1 consists of 153,000 data items, 60% of a student dataset, and the relation R_2 contains 168,000 data items, 80% from the whole recruiting dataset. The attributes in the dataset are name, primary email, alternative email, address, birthday, and phone. The approach tries to retrieve all distinct pairs of records $(r_1, r_2) \in (R_1, R_2)$ such that the error rate between the corresponding fields of strings are less than or equal to the given k threshold value. Java and PL/SQL applications were developed on the identified join attributes in the relation R_1 and R_2. The experiments were performed on a Sun Solaris 10 OS system and an Oracle 10g database, with 2 1200MHz CPU and 8GB memory. The experimental results were consistent on multiple runs.

5.4.2 ED and Q-gram on Clustering Fields

Figure 5-6 shows the precisions of Q-grams and edit distance on identified clustering join attributes. This figure shows that using Q-grams could return about 10% more relevant records on average in term of different threshold values than using edit distance.

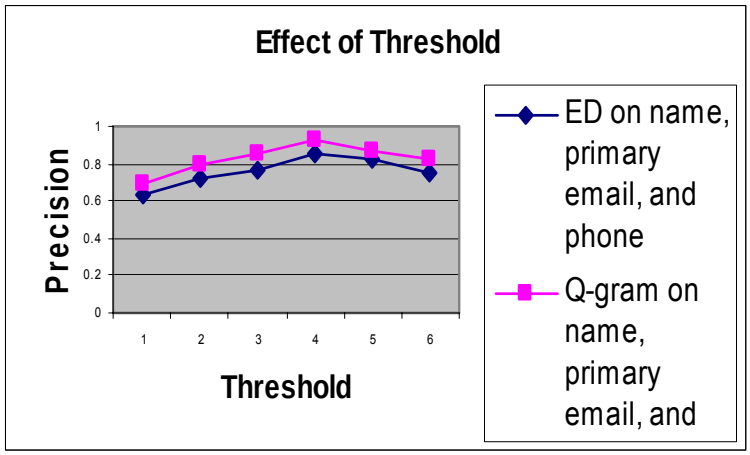


Figure 5-6: ED and Q-gram precision on affinity clustered attributes

Figure 5-6 shows the precisions of Q-grams and edit distance on identified clustered join attributes. This figure shows that using Q-grams could return about 10% more relevant records on average, using different threshold values, than using edit distance.

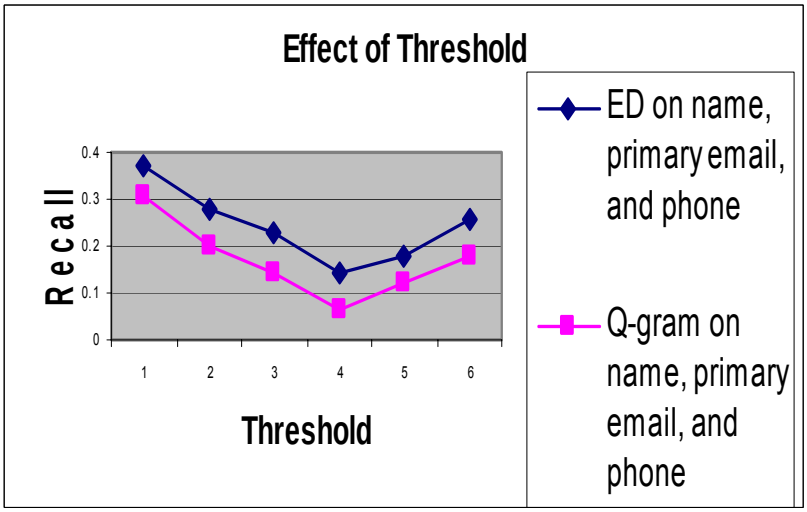


Figure 5–7: ED and Q-gram recall on affinity clustered join attributes

Figure 5-7 shows the results in term of recall performed by the edit distance on name, birth, and telephone attributes and Q-gram on name, birth, and telephone attributes, with different ε threshold values. The results show that

using Q-grams on the affinity clustered join attributes produces about 5% less relevant records, on average, for different ε threshold values, than using edit distance on the affinity clustered attributes.

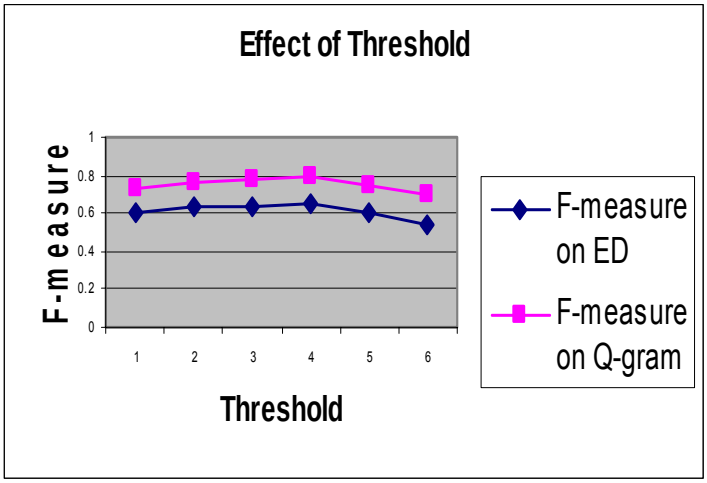


Figure 5–8: ED and Q-gram F-measure on affinity clustered attributes

Based on the combined measure of recall and precision, Figure 5-8 shows the F-measure results for Figure 5-6 and Figure 5-7. These results show that Q-grams on clustered affinity attributes have about 15% better F-measure than Edit Distance on clustered affinity attributes. The results indicate that the Q-grams clustered approach can produce more relevant results than ED clustered approach.

5.4.3 Q-gram implementation comparison

In the implementation, the approach has changed Q-gram distance from pure SQL to high level language. That gives it great performance comparing to the original version.

Figure 5-9 shows the time over number of records on Edit Distance, Q-gram Distance in Java and Q-gram Distance in SQL. The time increases

tremendously is because of the huge amount of temporary spaces. As this dissertation indicated earlier, temporary database spaces are needed to hold Q-gram records in pure SQL implementation. The spaces are growing as the records growth as well as the length of the length of records' growth. Figure 5-10 shows temporary database space on Edit Distance, Q-gram Distance in Java and Q-gram Distance in SQL.

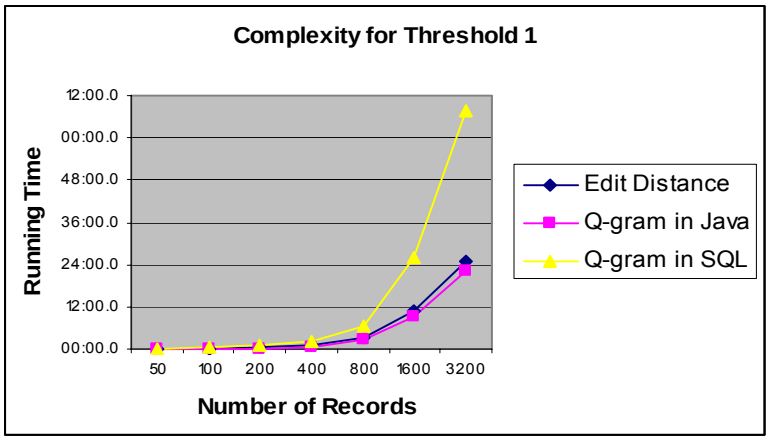


Figure 5-9: Running Time of Similarity Join Algorithms

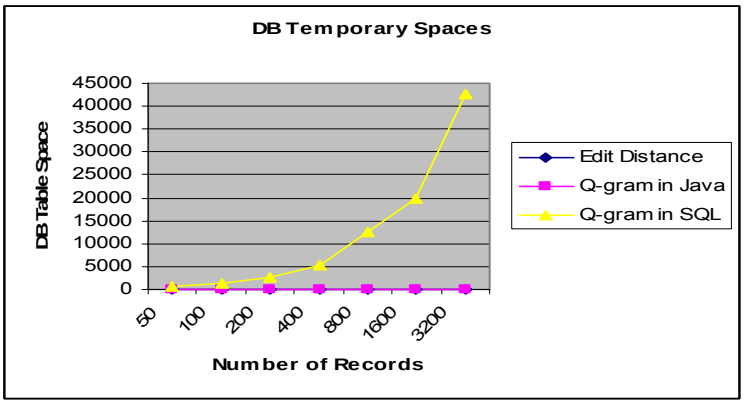


Figure 5-10: DB Temporary Spaces

5.5 Summary

In this chapter another pre-processing technique was presented to improve the true positives and decrease the false negatives of similarity joins or

similarity searches. At the beginning a brief overview of the previously proposed approach was given. That highly relies on knowledge of the data, and the concept of the previous approach, which is based upon the assumption that data are either fully dependent or fully independent. Therefore, the previous approach is not applicable or practical in some real applications.

In this current approach, it uses a fuzzy clustering paradigm. This approach is to place no sharp boundary between clusters and no pre-knowledge of the data is required. The experimental results have shown that this new approach is better than the previous approach in the sense of true positives and false negatives. The results can be applied to any approximate similarity join, approximate search, and data integration from multiple heterogeneous sources.

The current proposed approach is great for static, homogeneous and short string datasets to identify group of clustered related attributes. The potential future research will devise a way to handle static and long string datasets. Based on the nature of the datasets, the approach could identify the best clustered attributes to enhance existing similarity joins.

CHAPTER 6

PERFORMANCE COMPARISONS BETWEEN ABCA AND VBCA

The previous chapters described two proposed clustering-based pre-processing techniques to identify clustering related attributes. This chapter presents experimental results for performance and cost comparisons which should affect potential future studies. Firstly, performance comparisons were run on both edit distance and Q-gram distance using ABCA attributes, VBCA attributes and know join attributes as join predicates. Secondly, cost comparisons were made between pre-processing similarity join and non pre-processing similarity join. Thirdly, distance metrics were compared to further study the proposed approaches. Lastly the summary of the chapter is given.

6.1. Environmental Setup

Currently there are no commonly accepted benchmark datasets [CHS 07] for similarity joins. However, some researchers have used public datasets in their experimental studies. Although those public datasets are not benchmark datasets, they were used consistently in some recent studies [SK 04, XWL 08, LPSS 10]. In this experiment, the Digital Bibliography & Library Project (DBLP) dataset was used. DBLP is an important landmark snapshot for the Database community. It contains almost 1.4M records. Each record includes the author, title, publisher, year and pages. The author and title are quite often used as join attributes because they are the most meaningful fields to identify the records in the datasets and so they were treated as know join attributes for this study.

Java and PL/SQL applications were developed to identify attribute-based clustering attributes, value-based clustering attributes, and all distance metric functions. The experiments were performed on a Sun Solaris 10 OS system and running an Oracle 11g database, with four 1200MHz CPUs and 16GB of main memory. The experimental results were consistent on multiple runs.

6.2. Experimental Studies

Previous chapters described how edit distance and Q-gram distance were used in the experiments to evaluate the pre-processing results. This section includes descriptions of how the comparison experiments were conducted by applying edit distance and Q-gram distance to groups of clustering related attributes (Author, Title, Year) identified by ABCA, groups of clustering related attributes (Author, Title, Page Number) identified by VBCA, and known join attributes (Author, Title) identified by common sense.

The experiment conducted in this section is as follows: we chose two datasets of 5,000 records each. Each dataset was randomly selected from DBLP. We then performed similarity join operation between the two datasets on attributes Author and Title obtaining the resulting relation T. To calculate true positive (TP) in T, we first identified all the records that were exact match using equi-join operation. Assume that the number of such records is t_1. For the remaining tuples in T, we then performed an equijoin operation on attribute Page Number to find out which tuples in T are truly similar. Let us assume the number of such tuples is t_2. Therefore, we calculate the number of true positive to be $t_1 + t_2$. We performed this experiment 50 times and calculated the average

of TPs. For this experiment, we selected threshold to be 3 because this was determined by Gravano [GIJS 01]and others to be the best possible threshold.

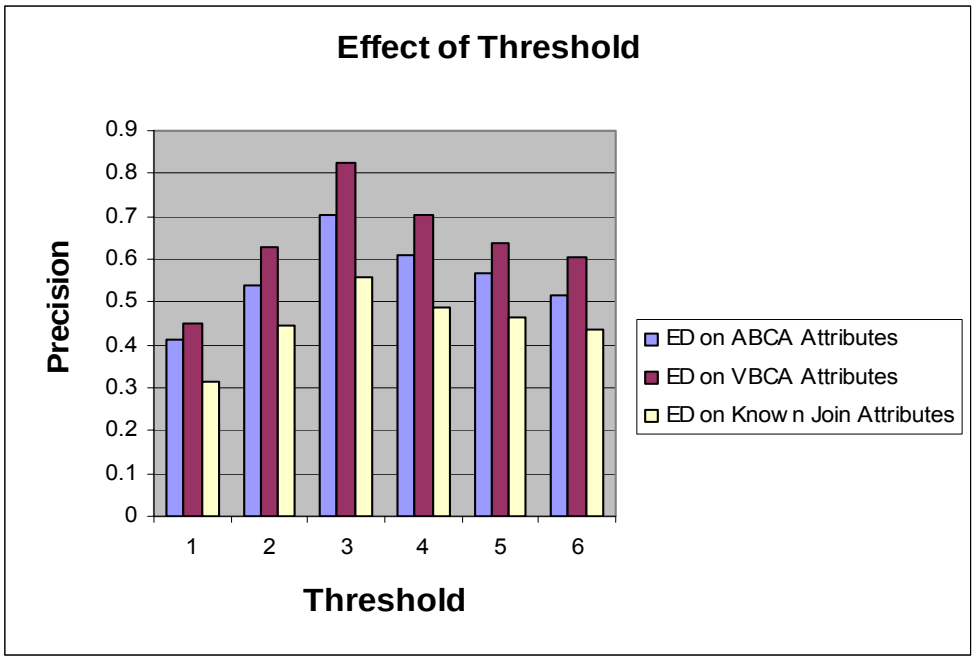


Figure 6-1: Precision on ED Using Various Predicates

Figure 6-1 shows the precision of edit distance on attribute-based clustering related attributes, value-based clustering related attributes and known join attributes. This figure shows that applying ED on value-based clustering related attributes will return about 10-15% more relevant records on average, using different threshold values, than applying ED on attribute-based clustering related attributes. It also shows that applying ED on attribute-based clustering related attributes will return about 10% more relevant records on average, using different threshold values, than applying ED on known join attributes. These observations validate the assumption, presented in previous chapters, that if the result of VBCA, the result of ABCA and the know join attributes are different,

VBCA can provide more accurate results than ABCA does because the value-based approach is more accurate than an attribute-based approach.

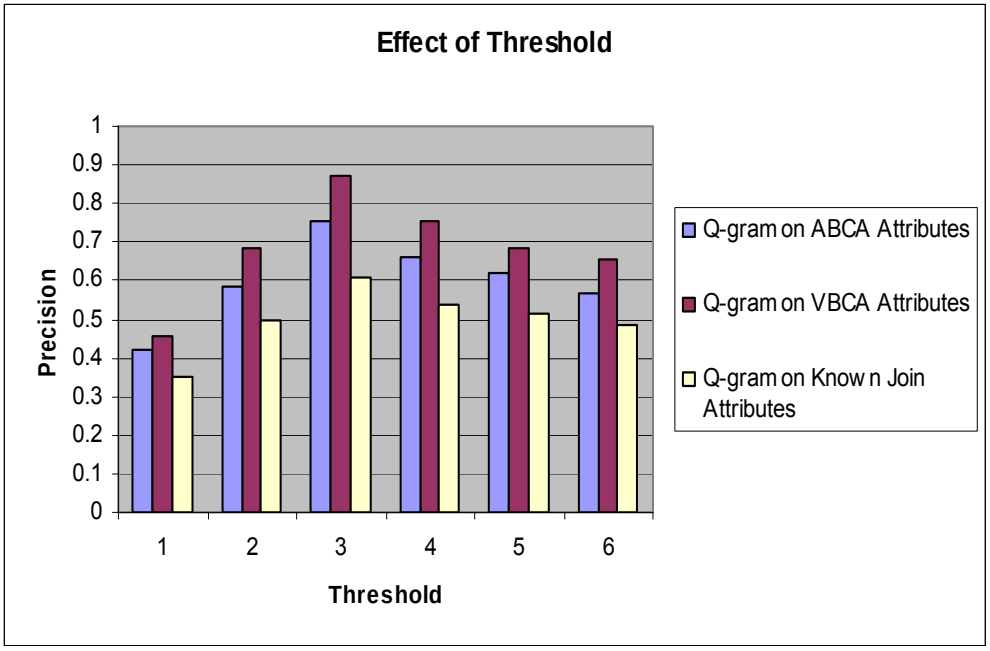


Figure 6-2: Precision on Q-gram Using Various Predicates

Similarly, Figure 6-2 shows the precision of Q-gram distance on attribute-based clustering related attributes, value-based clustering related attributes and known join attributes. This figure shows that applying Q-gram on value-based clustering related attributes will return about 8~12% more relevant records on average, using different threshold values, than applying Q-gram on attribute-based clustering related attributes. It also shows that applying Q-gram on attribute-based clustering related attributes will return about 10% more relevant records on average, using different threshold values, than applying Q-gram on known join attributes. These observations further show that when the result of VBCA, the result of ABCA and the know join attributes are different, VBCA can provide more accurate results than ABCA does since VBCA is based on the

value of datasets but ABCA is based on the application access information. The access information will vary upon some business needs.

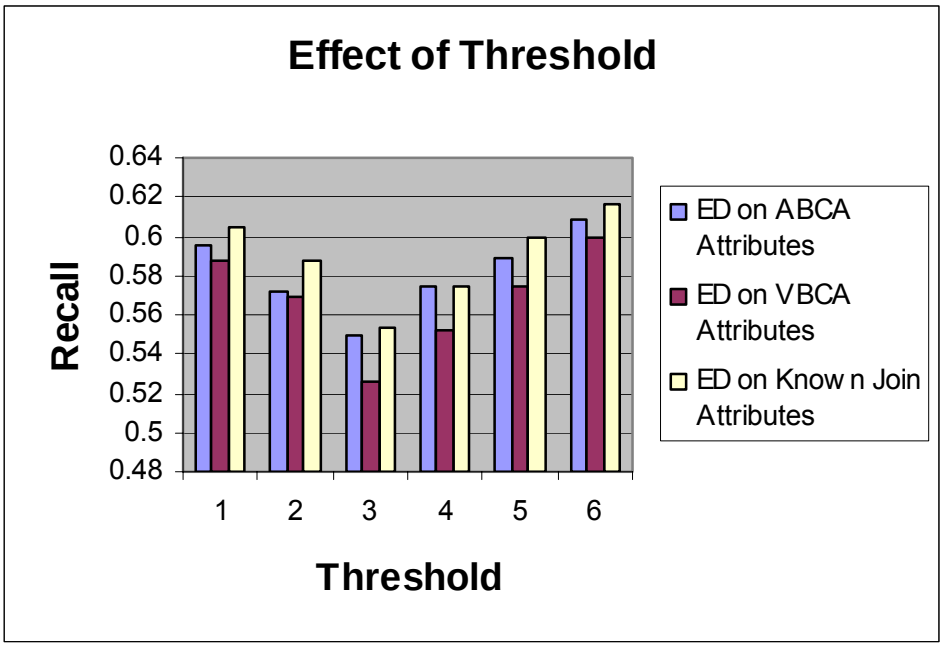


Figure 6-3: Recall on ED Using Various Predicates

Figure 6-3 shows the recalls of ED distance on attribute-based clustering related attributes, value-based clustering related attributes and known join attributes. This figure shows that applying ED on value-based clustering related attributes could return about 5-10% less relevant records on average, using different threshold values, than applying ED on attribute-based clustering related attributes. It also suggests that applying ED on attribute-based clustering related attributes could return about 5% less relevant records on average, using different threshold values, than applying ED on know join attributes.

Figure 6-4 shows the recall of Q-gram distance on attribute-based clustering related attributes, value-based clustering related attributes and known

join attributes. This figure shows the same performance patterns as Figure 6-3 does.

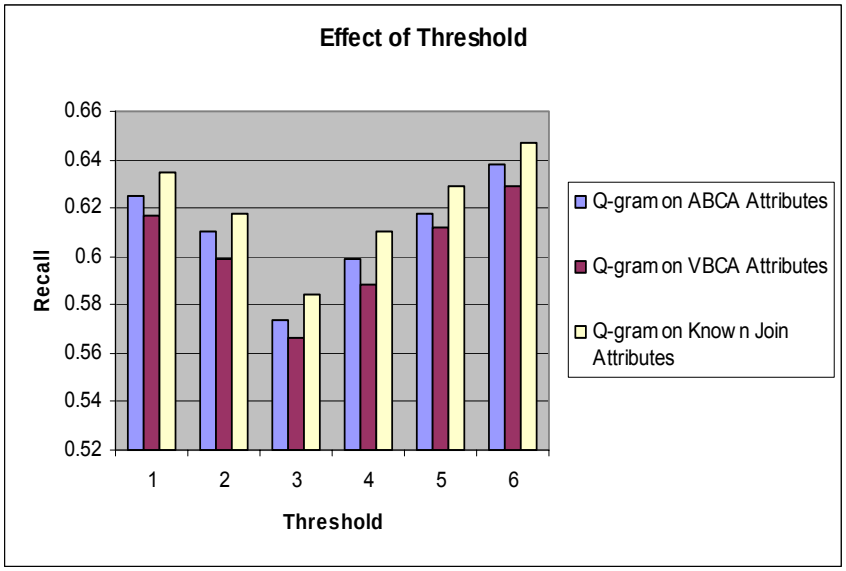


Figure 6-4: Recall on Q-gram Using Various Predicates

The observations in the above figures confirmed the common theory, described in the previous chapter, that when threshold increases, the true relevant records and false relevant records increase but false irrelevant records decrease. These inverse results are shown by comparing Figure 6-1 and Figure 6-3 as well as Figure 6-2 and Figure 6-4.

Figure 6-5 shows the F-measure of ED distance on attribute-based clustering related attributes, value-based clustering related attributes and known join attributes. This figure shows that applying ED on value-based clustering related attributes could return about 4~7% more relevant records on average, using different threshold values, than applying ED on attribute-based clustering related attributes. It also shows that applying ED on attribute-based clustering related attributes could return about 4-7% more relevant records on average,

using different threshold value, than applying ED on known join attributes. These experimental observations confirm the theoretical assumption presented in previous chapters, that overall VBCA can provide more accurate results than ABCA does because a value-based approach is based on the value of datasets and more precise than an attribute-based approach, where the attribute-based approach depends on how the applications use the attributes.

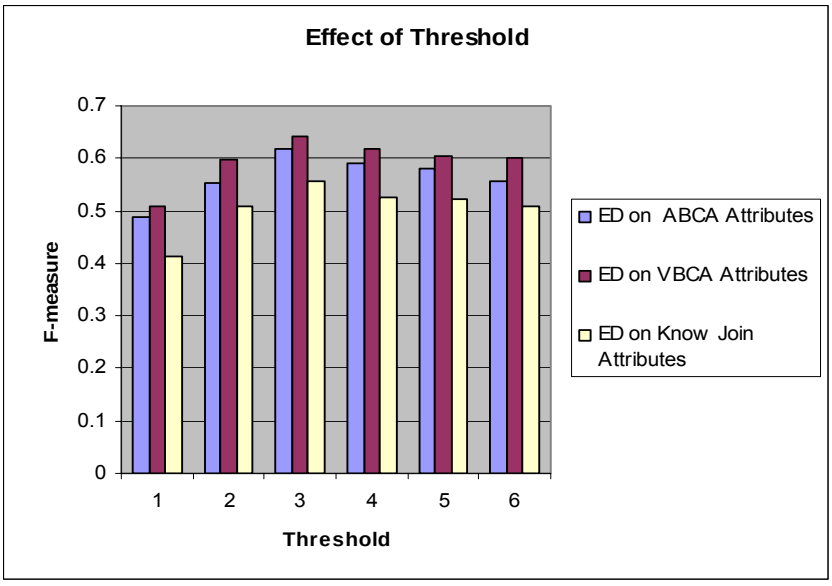


Figure 6-5: F-measure on ED Using Various Predicates

Figure 6-6 shows the F-measure of Q-gram distance on attribute-based clustering related attributes, value-based clustering related attributes and known join attributes. Figure 6-6 presents a similar performance pattern as Figure 6-5 does. These experimental observations further show that if the clustering attributes are different than know join attributes, applying similarity join approaches on pre-processing clustering attributes will produce better performance results than applying similarity join on know join attributes because the clustering attributes can identify entities more accurately than no clustering

attributes. Both Figure 6-5 and Figure 6-6 also show overall VBCA can return more accurate results than ABCA does.

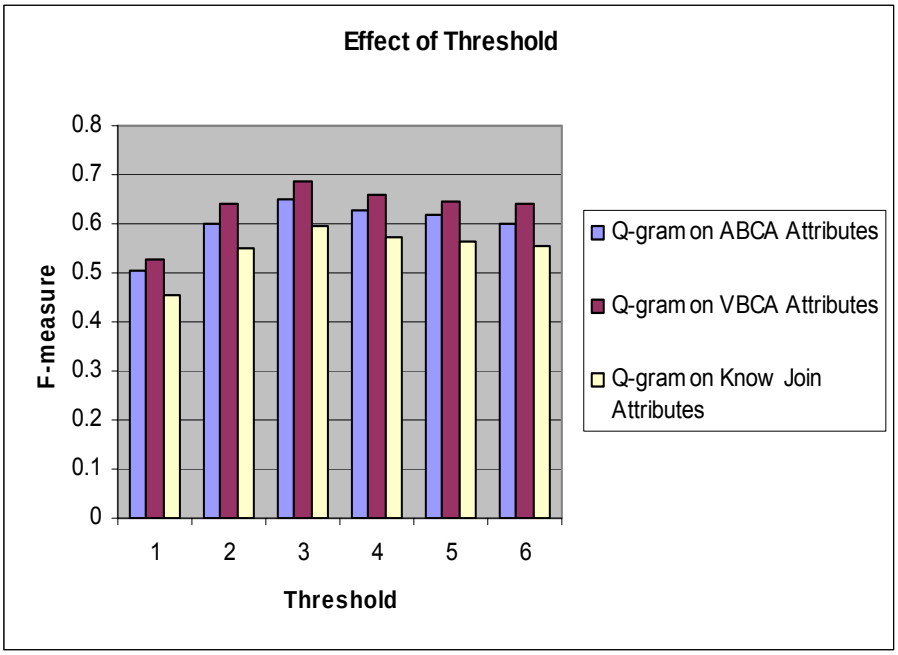


Figure 6-6: F-measure on Q-gram Using Various Predicates

In summary, the preceding figures present the results in terms of the precision, in terms of the recall, and in terms of F-measure, of these three join approaches. In general, the proposed pre-processing clustering-based approaches in this study produced some improvements in overall precision and F-measure compared with using ED distance and Q-gram distance on known join attributes if identified clustering attributes are different than known join attributes. Also VBCA used value of datasets to identify clustering attributes which were used as join predicates and returned more true positive results than ABCA did. Therefore, the proposed pre-processing approaches can unveil join predicates, reflect true attribute affinity on either ABCA or VBCA, and result in promising outcomes.

6.3. Cost Comparisons between Pre-processing and Non-preprocessing Approaches

This section includes cost comparisons that were conducted between two proposed clustering-based pre-processing approaches and non pre-processing by complexities and running time of the approaches. These comparisons can be used to determine if the pre-processing approaches are necessary when similarity join techniques are applied in the applications. This discussion presents some pros and cons about the approaches and focuses on edit distance metric and q-gram distance metric, that serve as base-line, efficient and effective similarity join techniques. More distance metric studies will be presented in the following sections to extend the choices of evaluation approaches.

Table 6-1: Parameters used for Pre-processing

Parameter	Meaning
a	Number of applications
b	Number of attributes in datasets
$n_1 = \mid A \mid$	Number of records in the first datasets A
$n_2 = \mid B \mid$	Number of records in the second datasets B
n	Sum $(n = n_1 + n_2)$ of records in both datasets
$\mid s \mid$	Length of source string s
$\mid t \mid$	Length of target string t
q	Number of the q-gram

Table 6-1 lists parameters needed for the following analysis. All the parameters are either known or calculated before pre-processing except for number of the q-gram q, which can be based on the applications' needs. In this experiment, q was chosen as 3, that is based on the experimental results in Gravano's paper [GIJMS 01].

Table 6-2: Cost of Pre-processing Approaches

	Complexity	Space
ABCA	$a*b^2$	$a+2*b$
VBCA	$b*n^2$	$2*n$
Edit Distance Metric	$\|s\|*\|t\|$	$2*(\|s\|,\|t\|)$
Q-gram Distance Metric	$\max(\|s\|,\|t\|)$	$\max(\|s\|,\|t\|)$

Table 6-2 shows the cost summary from the previous chapters. Since the cost of ABCA is $O(a*b^2)$ and the cost of VBCA is $O(b*n^2)$, the attribute-based clustering approach is much less expensive than value-based clustering approach. The cost of general edit distance is $|s|*|t|$ and the cost of Q-gram is $\max(|s|,|t|)$, therefore ED uses more time than Q-gram does.

The experiment conducted in this section is as follows: we chose two datasets of 5,000 records, 50,000 records, and 500,000 records each. Each dataset were randomly selected from DBLP. We assume the running time for ABCA is m_a and the running time for VBCA is m_v. We then performed similarity join operation between pair wise datasets on attributes Author and Title obtaining

the resulting relation T. Assume the running time is m_1. To calculate true positive in T, we first identified all the records that were exact match using equi-join operation. Assume that the running time is m_2. For the remaining tuples in T, we then performed an equijoin operation on attribute Page Number to find out which tuples in T are truly similar. Let us assume the running time is m_3. Therefore, we calculate the running time is $m_1 + m_2 + m_3$ for known join attributes, for $m_a + m_1 + m_2 + m_3$ for ABCA, and $m_v + m_1 + m_2 + m_3$ for VBCA. We performed this experiment 50 times and calculated the average of those running times. For this experiment, we selected threshold to be 3 because this was determined by Gravano [GIJS 01]and others to be the best possible threshold.

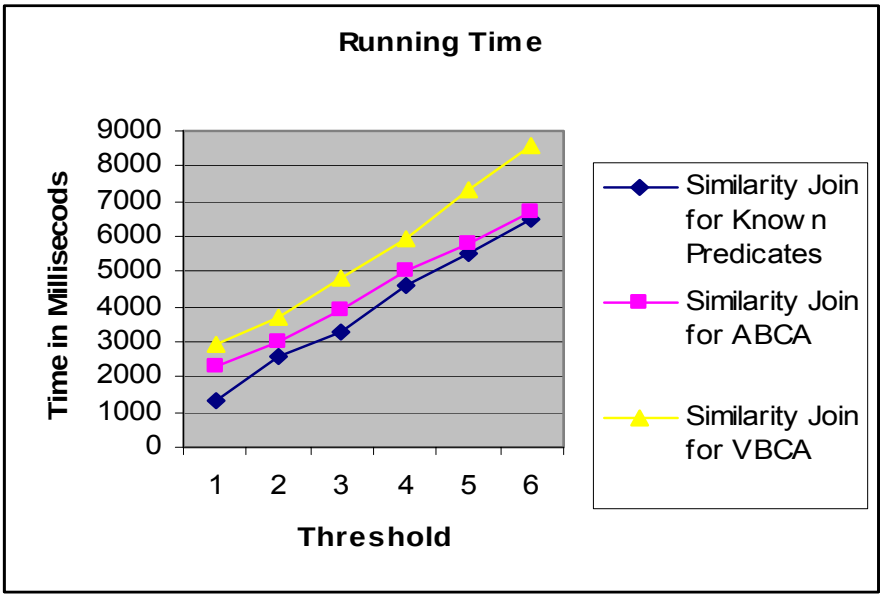


Figure 6-7: Running Time on 5,000 Records

Figure 6-7 shows the running time increases for all approaches when the threshold is increased. The figure presents the time using an ABCA approach is closer to VBCA approaches when the threshold is small. However, when the

threshold gets large, the time on ABCA approach is close to known join attributes' approaches.

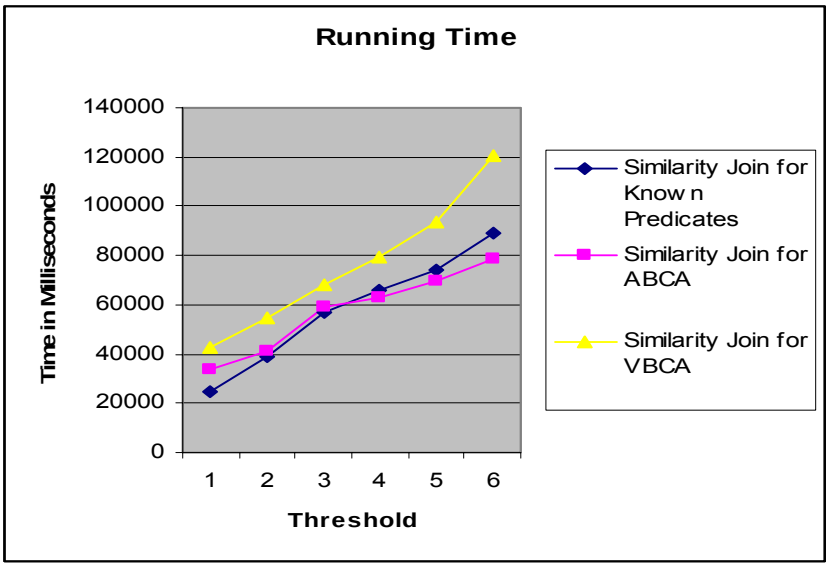


Figure 6-8: Running Time on 50, 000 Records

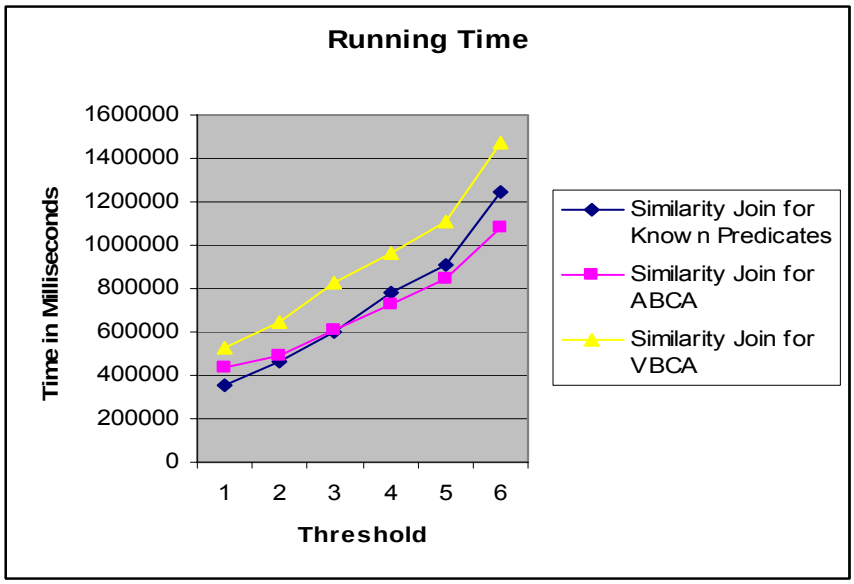


Figure 6-9: Running Time on 500,000 Records

Figure 6-8 and Figure 6-9 show similar experimental outcomes, except one can tell there is an intersection between threshold 3 and 4. The cross point

indicates after the intersection point, ABCA approach requires less running time than a known join attribute approach.

6.4. Distance Metric Studies

As some authors [CHS 07, LPSL 10] found, there are various similarity distance metrics, such as Euclidean distance, Cosine distance, Q-gram distance, and edit distance, used to quantify similarity values between entities or objects. Generally, there is no metric that is universally best for all kinds of application domains so which metric is to be used depends on application domains. Wikipedia [Wiki 09] has a comprehensive summary of all the distance metrics. This study dealt with string applications in the database domain.

In the database domain, there are 9 commonly used distance metrics [Wiki 09], which are: Hamming distance [Ham 50, PWP 08], Levenshtein distance [Lev 66], Jaro distance [Jar 89, Jar 95], Jaro Winkler distance [Jar 99], SoundEx distance, Jaccard similarity [Jac 01] or Jaccard Coefficient or Tanimoto coefficient [Tan 57], Euclidean distance, Cosine similarity , and Q-gram [GIJMS 01]. These distance metrics have been implemented for different types of applications in the database domain. Hamming distance is defined as the number of bits which differ between two binary strings and suitable for exact length comparisons. Levenshtein distance is the basic distance and good for any type of strings. Jaro Winkler distance is an extension of Jaro distance which takes into account typical spelling deviations. Soundex distance is a coarse phonetic indexing scheme focused upon individual names so it has not been provably applied to a more general context. Jaccard similarity is a token based

vector space similarity measure and frequently used as similarity measure for chemical compounds. Euclidean distance is a vector space similarity distance and a standard metric for geometrical problems. Cosine similarity is a common vector based similarity measure and a great similarity measure for document text. More precisely, cosine distance is a measure of similarity between two vectors of n dimensions by finding the cosine of the angle between them. Q-gram distance is typically used in approximate string matching.

Wei [Wei 10] summarized different areas of similarity join metrics and found Q-gram distance metric is the best approach to reduce false positives since q-gram is a substring of the original string and the short string comparison is more accurate than the long string comparison. Chandel and others [CHS 07] grouped the types of similarity distance metrics, showed accuracy and performance on similarity functions, and observed cosine similarity metric had comparatively good accuracy and performance on detecting errors from string datasets. An efficient similarity join algorithm is derived from cosine similarity distance.

The experiment conducted for different distance metric comparison is as follows: we chose two datasets of 5,000 records each. Each dataset were randomly selected from DBLP. We assume the running time for ABCA is m_a and the running time for VBCA is m_v. We then performed similarity join operation between the two datasets on attributes Author and Title obtaining the resulting relation T. Assume the running time is m_1. To calculate true positive in T, we first identified all the records that were exact match using equi-join operation.

Assume that the number of such records is t_1 and the running time is m_2. For the remaining tuples in T, we then performed an equijoin operation on attribute Page Number to find out which tuples in T are truly similar. Let us assume the number of such tuples is t_2 and the running time is m_3 Therefore, we calculate the number of true positive to be $t_1 + t_2$ and he running time is $m_1 + m_2 + m_3$ for known join attributes, for $m_a + m_1 + m_2 + m_3$ for ABCA, and $m_v + m_1 + m_2 + m_3$ for VBCA. We performed this experiment 50 times and calculated the average of TP_s and running times. For this experiment, we selected threshold to be 3 because this was determined by Gravano [GIJS 01] and others to be the best possible threshold.

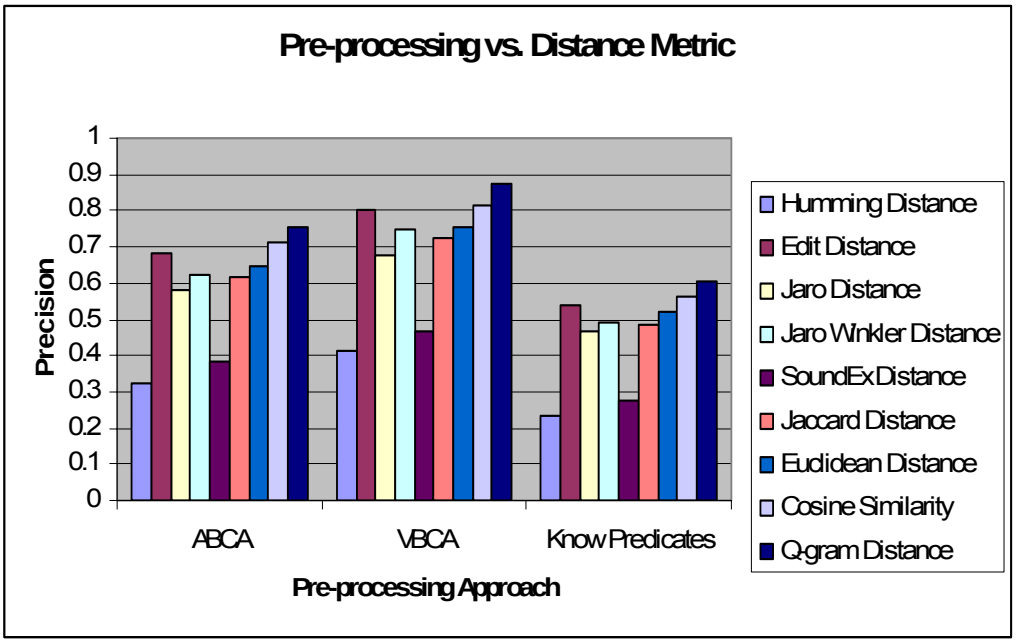


Figure 6-10: Predicates vs Distance Metrics

Figure 6-10 shows the experimental results from different predicates run with various string distance metrics. The Q-gram distance metric was the best

distance metric for string similarity join applications. Based on characteristics of string distance metrics described previously, humming distance was good for exact length comparisons and DBLP dataset contains strings in various length so it has the worst precision on the figure; Jaro and Jaro Winkler distances were good for spelling deviations, and SoundEx distance was good for phonetic applications although it doesn't have good precision when data are dirty caused by a numerous reasons such as typo, different name convention, etc., Euclidean distance, Jaccard similarity and Cosine similarity are great for document matching and searching applications although they are not optimally used on short string dataset but they still return sort of good precision but not as good as Q-gram distance. For running time comparisons, including pre-processing time and similarity join time, this study considered Edit distance, Cosine similarity and Q-gram distance.

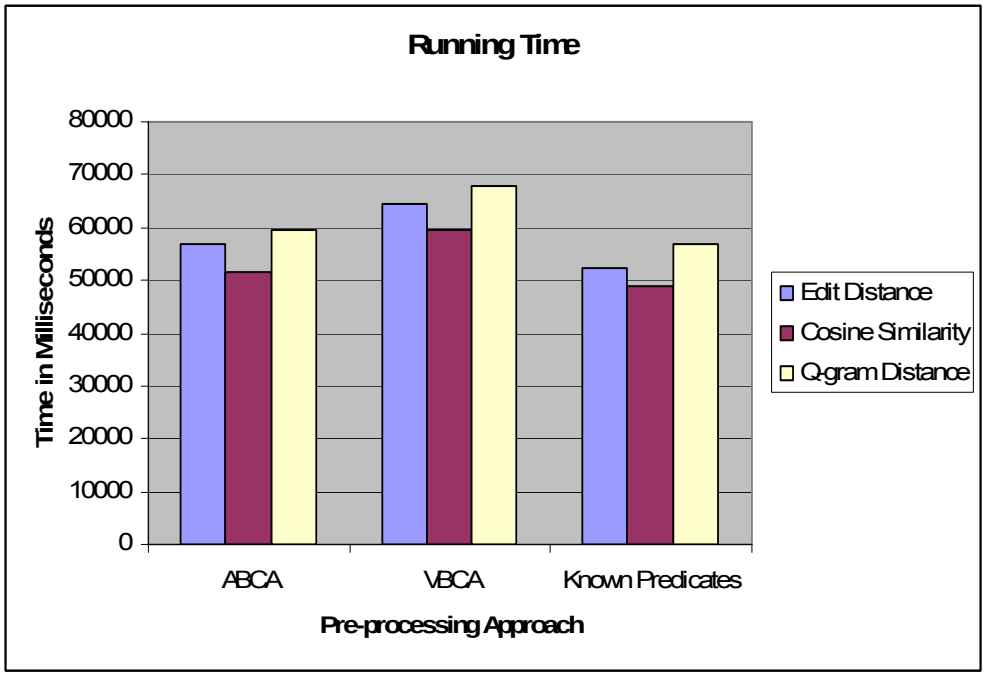


Figure 6-11: Pre-processing vs. Distance Metric on Running Time

Figure 6-11 shows the experimental results on pre-processing approaches against distance metric. The results showed that because it takes time to break strings into Q-gram substrings and compare those substrings, Q-gram is the most expensive distance and Cosine similarity is the least expensive distance. By considering the precision result from Figure 6-10, Q-gram distance should be used if datasets contain more short strings; Cosine similarity should be used if datasets contain more long strings.

6.5. Summary

In this chapter, theoretical and experimental comparisons have been given between clustering approaches ABCA and VBCA. The results showed that both clustering-based approaches will improve the performance of similarity join techniques. However, those improvements added additional cost. The value-based clustering approach had better performance results than attribute-based clustering approach does. The choice of approaches should depend on the applications. If the application data are more static, VBCA is a better choice because VBCA provides better performance results. If the data are more dynamic, ABCA is a better choice because there is no need for re-clustering when the data are changed. Distance metric studies were summarized in this chapter along with the pros and cons of the different distance metrics. Among those metrics, Q-gram gave the best precision results and Cosine gave the best running time result. To get the better running time results, Q-gram is good when it is used with short string datasets and Cosine is good when it is used with long string datasets. These comparisons not only produced some trade-offs between

the two approaches and distance metrics but also suggested some potential areas for additional research.

CHAPTER 7

FUTURE WORK

This study addressed one of similarity join issues in data integration. Similarity join has been a topic of research for more than forty years and has gained in popularity over recent years because of the continuously increasing amount and availability of data from local and global sources. While the similarity join in early research tackled issues mostly related to the availability of data, the focus of recent research has shifted toward aspects of data usability in distributed and heterogeneous environments.

Data usability is highly dependent on the correctness of data output results. Dirty data caused by missing data, data errors, data duplication, different data format, and data inconsistencies has attracted a significant amount of attention. Therefore, there are many similarity join approaches having been proposed to address immediate issues. Unfortunately, those approaches assumed that the join predicates were pre-defined and optimal. This study addressed the issues of similarity join predicates.

7.1 Summary of Contributions

This study focused on developing, implementing and evaluating two clustering-based pre-processing approaches to improve existing string similarity join techniques.

Attribute-Based Clustering approach: The dissertation proposed and implemented an attribute-based clustering pre-processing approach for improving existing similarity join techniques. This first proposed approach

was based on the usage of attributes in the applications. Changing the value of datasets will not impact clustering results, but changing names of attributes or access frequencies of attributes might impact clustering results. Results showed that Attribute Based Clustering was an excellent approach for dynamic datasets. It can be embedded with existing similarity join techniques and would be effective and extendable for more general similarity techniques.

Value-Based Clustering approach: The second clustering-based pre-processing approach was based on the values of attributes in the datasets. This second proposed approach was used fuzzy c-means to find clusters in a collection of unlabeled data. Changing the value of datasets might impact the clustering results; however, changing names of attributes or access frequencies of attributes will not impact the clustering results. The experiment showed this approach is superior for static datasets. Contrasted with Attribute Based Clustering, this approach can be easily applied when there is no prior-knowledge about the applications using the datasets. The approach also works well when there are vague or fuzzy boundaries between clusters. The experiments also have shown a value- based clustering approach is more reliable and accurate than an attribute- based clustering approach.

Greedy Approach: Greedy strategy was utilized on the implementation of the value-based clustering approach. The greedy strategy evaluates every possibility of attribute combinations and picks the best options at each

level before moving to the next level. The approach will eliminate the $l-1$ paths on each level where l is the total number of possible ways to cluster attributes. The overall of complexity improvement was not only demonstrated by the experiments but also showed on the mathematical calculation.

New Q-gram Implementation: The typical Q-gram implementation used the native of database language – SQL and temporary table space created on the fly to store Q-grams. In this study, the approach used the nature of high-level language to do database connection calls to get the data and handle data comparisons on the fly. The experiments showed that this new Q-gram approach provided improvements on both complexity and table spaces.

7.2 Future Work

String similarity join for data integration has been gaining noticeable attention in various research areas due to its significance in many applications as well as the expanding and increasing use of eTechnology. The pressure on online repository, oral human-machine communication, the heterogeneity, and spelling errors presented in textual databases, web searching, data warehouse and bio-informatics drives the research to produce much higher precision results. This research only addressed one of the many open issues related to string similarity join, namely, how to find a group of clustering related attributes to improve the performance of existing similarity join techniques.

Both of the proposed approaches in this study showed some promising results when they were used in short strings, static schema, static attribute usage, static datasets and homogenous environments. However, in the real-world, databases will be of various forms and even in the same database the structure of data schema can be different. Data are frequently changed on the fly; the usage of applications is changed based on the application needs, the size of datasets is often massive especially in medical fields and the types of data are various based on specific business requirements. All of those changes require more sophisticated approaches to identify and simplify similarity records to help in diverse database settings and very large datasets [ZRK 96].

To meet the above potential research challenges on reducing false negative and increasing true positives, this research was focused on extending the proposed approaches to efficiently and effectively handle long strings, heterogeneous, dynamic data, dynamic schema and dynamic attributes usage,

Clustering-based Pre-processing Approach for Long String: For long strings, researchers must deal with different data types like LOB which is used in many online applications for collecting comments or medical explanation or documents. Those kinds of long strings contain many articles and conjunctions which might not need to be exactly the same to produce the same meaning so extracting key words before doing similarity join might be one of the approaches to avoid the long string comparisons.

Clustering-based Pre-processing Approach for Heterogeneous Datasets: In current diversified environments, it is normal that schemas are different between different databases even in the same database. This study assumed that the source schemas are the same, if they are not the same, they can be easily mapped. In future work, researchers could extend these approaches to deal with more general schema. The potential approach is to look at ontology methodologies and use the ontology to map the schema between different naming convention databases before a clustering-based pre-processing approach is applied.

Clustering-based Pre-processing Approach for Dynamic Datasets: in current eTechnoloy era, data are changed often for numerous reasons. In this study, the approaches were applied on simply datasets and datasets did not consider dynamic. When data are changed, the cluster might get changed with a value-based clustering approach. The potential future work should consider developing an approach to limit or eliminate masterly re-classification from to cope with data changed on the datasets. The possible solution is to analyze the change records to see how serious impact could be on partitioning entropy and partitioning co-efficiency. Based on analyzing the results, the approach should be able to conditionally decide if the master recalculation is necessary or not. If this approach can be implemented, it will reduce the overall complexity for re-classification.

Clustering-based Pre-processing Approach for Dynamic Attribute Usages: in real-world, it is likely usages of attributes are being changed because of changes in the application requirements. In this study, the ABCA approach fit well with the static attribute usage. A potential solution for changes on the usages of attributes is to analyze the attribute usage change to see if there is a need to do a re-classification. Finding an efficient re-classification approach should produce a significant benefit when using an ABCA approach.

Clustering-based Pre-procession Approach for dynamic schema: there are many applications that use large amounts of attribute fields to meet their application needs. The approaches in this study work well with small amount of attributes, for example, in ABCA, bi-non-overlapping split approach was used to find the best clustering group. When the amount of attributes is increased, the bi-non-overlapping split approach might not work as well. The possible approach to deal this change is to analyze the access frequency before deciding how many split segments will need to be generated. Finding a right number of attribute splits for applications is other challenging task on this type of research fields.

The most pressing challenge among the above is to avoid master recalculation when attribute usages are getting changed on attribute-based clustering approach and the new records are added or the some records are deleted on value-based clustering approach.

Bibliography

[ABU 79] A. H. Aho, C Beeri and J.D. Ullman “The Theory of Join Relational Databases,” ACM Trans. Database Syst. vol. 4, no. 3, 1979, pp. 297-314.

[Alb 67] Cyril N. Alberga, “String Similarity and misspellings,” Commun. ACM 10, Vol. 5, 1967, pp. 302-313

[Bez 74] James C. Bezdek, “Cluster Validity with Fuzzy Sets,” Journal of Cybernetics, vol. 3, no. 3, 1974, pp. 58-72.

[Bez 75] James C. Bezdek, “Mathematical Models for Systematics and Taxonomy,” Proceedings of the 8-th International Conference on Numeric Taxonomy, San Francisco, CA, 1975, pp. 143-166.

[Bez 81] James C. Bezdek, “Pattern Recognition with Fuzzy Objective Function Algorithms”, Plenum Press, NY, 1981.

[BG 97] I. Borg and P. Groenen, “Modern Multidimensional Scaling: Theory and Applications”, Springer-Verlag, New York, 1997.

[BMS 07] Roberto J. Bayardo, Yiming Ma, and Ramakrishnan Srikant, “Scaling Up All Pairs Similarity Search”, Proceeding of the 16th Int’l Conf. on World Wide Web, Canada. 2007, pp.131-140

[BN 06] Jens Bleiholder and Felix Naumann, “Conflict Handling Strategies in an integrated Information System”, WWW Workshop in Information Integration on the Web (IIWeb). Edinburgh, UK. 2006, pp. 246-259

[BN 99] R. Baeza-Yates and G. Navarro, “Very fast and Simple Approximate String Matching”, in Proc. Of the 10th Annual Symposium on Combinatorial Pattern Matching (CPM’99) LNCS 1645, 1999, pp.163-185

[BP 92] James C. Bezdek and S. K. Pal, “Fuzzy Models for Pattern Recognition: Methods That Search For Structures in Data”, IEEE Press, New York, 1992.

[CGK 06] S. Chaudhuri, V. Ganti, and R. Kaushik. ”A primitive operator for similarity joins in data cleaning”. In Proc. of the 22nd Intl. Conf. on Data Engineering, 2006, pp.5-22

[CHS 07] Amit Chandel, Oktie Hassanzadeh, and Divesh Srivastava, “Benchmarking Declarative Approximate Selection Predicates”, Processings of the 2007 ACM SIGMOD international conference on Management of data, 2007, pp. 353-364

[CRF 03] William Cohen, P. Ravikumar, and S. E. Fienberg, “A comparison of string distance metrics for name-matching tasks”, In Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03), to appear. 2003, pp. 73-78

[DTS 08], Hui Ding, Goce Trajcevski, Peter Scheuermann, “Efficient Similarity Join of Large Sets of Moving Object Trajectories”. 15th International Symposium on Temporal Representation and Reasoning, TIME 2008, pp. 79-87

[FK 99] H Frigui, R. Krishnapuram, “A Robust Competitive Clustering Algorithm with Applications in Computer Version,” IEEE Transactions on Pattern Analysis and Macine Intelligence, vol. 21, no. 5, 1999, pp. 450-465.

[FL 95] Christos Faloutsos and King-Ip (David) Lin, “FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets”, Proceedings of the 1995 ACM SIGMOD international conference on Management of data, San Jose, USA, 1995, pp.163-175

[GIJMS 01] L. Gravano, P. Iperirotis, H. V. Jagadish, N Koudas, S. Muthukrishnam and D. Srivastav, “Approximate String Joins in a Database (Almost) for Free,” Proceedings of the 27th VLDB, 2001, pp. 491-500.

[GIJS 03] L. Gravano, P. Iperirotis, H. V. Jagadish, N. Koudas and D. Srivastava, “Text joins in an RDBMS for web data integration”, In IEEE, May, 2003, pp. 90-101

[Ham 50] Richard W. Hamming, “Error detecting and error correcting codes”, Bell System Technical Journal, Vol. 29, 1950, pp. 147-160

[HBV 02] M. Halkidi, Y. Batistakis, M. Vazirgiannis, “Cluster Validity Methods”, SIGMOD Record, Part I and II, 2002, pp. 458-469.

[HM 04] Yingping Huang and Gregory Madey, “Web Data Integration Using Approximate String Join”, Proceedings of the 13th International World Wide Web conference on Alternate track papers & posters, NY USA 2004, pp.557-584.

[Jac 01] Paul Jaccard, “Étude comparative de la distribution florale dans une portion des Alpes et des Jura”, Bulletin de la Société Vaudoise des Sciences Naturelles, Vol. 37, 1901, pp.547–579.

[Jar 89] M. A. Jaro, "Advances in record linkage methodology as applied to the 1985 census of Tampa Florida". *Journal of the American Statistical Society* **84** (406), 1989, pp. 414–420.

[Jar 95] M. A. Jaro, “Probabilistic linkage of large public health data file “, Statistics in Medicine, Vol. 14, 1995, pp. 491-498

[KKR 10] Hans-Peter Kriegel, Peer Kroger, and Matthias Renz, “Techniques for efficiently searching in spatial, temporal, spatio-temporal, and multimedia databases”, Proceeedings of the 26th International Conference on Data Engineering, ICDE 2010, California, USA, 2010, pp.1218-1219

[KMK 09] Manish Kumar, Shane Moriah, Srikumar Krishnamoorthy, “Performance Evaluation of Similarity Join for Real Time Information Integration”, Proceedings of the 2nd Bangalore Annual Compute Conference, 2009, pp. 476-483

[KP 07] Dmitri V. Kalashnikov and Sunil Prabhakar. “Fast similarity join for multi-dimensional data”, Information Systems, Vol. 32. 2007, pp.160-177

[JLYY 10] Jeffrey Jestes, Feifei Li, Zhepeng Yan, and Ke Yi, “Probabilistic string similarity joins”, Indiana, USA, 2010, pp.327-338

[JL 05] Liang Jin and Chen Li, “Selectivity Estimation For Fuzzy String Predicates in Large Data Sets”, Proceedings of the 31st VLDB conference, Trondheim, Norway, 2005, pp. 397-408.

[JS 08] E. H. Jacox and H. Samet, “Metric space similarity joins”, *ACM Trans. Database Syst.*, 33(2), 2008, pp.517-529

[Lev 66] V. I. Levenshtein, “Binary Codes Capable of Correcting Deletions, Insertions and Reversals,” Soviet Physics-Doklady, vol. 10, 1966, pp. 707-710.

[LLFZ 08] Xuhui Liu, Guoliang Li, Jianhua Feng, and Lizhu Zhou, “Effective Indices for Efficient Approximate String Search and Similarity Join”, Proceedings of the 2008 The Ninth International Conference on Web-Age Information Management, 2008, pp.127-134

[LPSL 10] Dogjoo Lee, Jaehui Park, Junho Shim, and Sang-goo Lee, “An Efficient Similarity Join Algorithm with Cosine Similarity Predicate”, DEXA 2010, Part II, LNCS 6262, 2010, pp. 422-436

[LSS 08] Michael D. Lieberman, Jagan Sankaranarayanan, Hanan Samet, “A Fast Similarity Join Algorithm Using Graphics Processing Units,” In Proceedings of the 24th IEEE International Conference on Data Engineering, Cancun, Mexico, April 2008, pp. 1111-1120.

[Mac 67] J. B. MacQueen, “Some Methods for Classification and Analysis of Multivariate Observations,” Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, vol. 1, 1967, pp. 281-297.

[MNAA 07] Farhi Marir, Yahiya Najjar, Mahmoud Y. AlFaress, and Hassan I. Abdalla, “An Enhanced Grouping Algorithm for Vertical Partitioning

Problem in DDBS", 22nd International Symposium on Computer and Information Sciences, Ankara, Turkey, 2007, pp.364-371

[MSW 72] W. T. McCormick, P. J. Schweitzer and T. W. White, "Problem Decomposition and Data Reorganization by a Clustering Technique", Oper. Res. (1972), 20(5), 1972, pp. 993-1009

[MU 00] Nigam K. McCallum and L. H. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching", In Knowledge Discovery and Data Mining, 2000, pp. 169–178

[Nav 01] Gonzalo Navarro, "A Guided Tour to Approximate String Matching," ACM Computing Surveys, vol. 33, no.1, March, 2001, pp. 31-88.

[NU 00] McCallum, K. Nigam and L. H. Ungar, Efficient clustering of high-dimensional data sets with application to reference matching, In Knowledge Discovery and Data Mining, 2000, pp.169–178

[OV 91] M. Tamer Ozsu and Patrick Vlduriez, "Principles of Distributed Database Systems," Prentice Hall, Englewood Cliffs, New Jersey, 1991, pp. 104-135.

[PWP 08] C. D. Pilcher, J. k. Wong, S. K. Pillai, "Inferring HIV transmission dynamics from phylogenetic sequence relationships", PloS Med. Vol. 5(3), 2008, pp. 69-84

[RD 97] Dick de Ridder, Robert P. W. Duin, "Sammon's Mapping using Neural Networks," Pattern Recognition Letters, vol. 18, 1997, pp. 1307-1316.

[Rij 79] C.V. Van Rijsbergen, "Infromation Retrieval", London: Boston. Butterworth, 2nd Edition, ISBN 0-408-70929-4, 1979.

[Sam 69] J. W. Sammon, “A Nonlinear Mapping for Data Structure Analysis,”IEEE Transactions on Computer, vol. C, no. 18, 1969, pp. 401-409.

[SB 02] Sunita Sarawagi and Anuradha Bhamidipaty, “Interactive deduplication using active learning”, In Proc. of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2002), Edmonton, Canada, July 2002, pp. 269-278

[SK 04] S. Sarawagi and A. Kirpal, “Efficient set joins on similarity predicates”, In Proceedings of the 2004 ACM SIGMOD international conference on Management of data, 2004, pp. 743-754

[SM 00] Jianbo Shi and Jitendra Malik, “Normalized Cuts and Image Segmentation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, August 2000, pp. 888-905.

[SM 83] G Salton and M. J. McGill, “Introduction to Modern Information Retrieval”, McGraw-Hill, 1983.

[SSS 04] Eike Schallehn, Kai-Uwe Sattler, Gunter Saake, “Efficient similarity-based operations for data integration”, Data & Knowledge Engineering, Vol. 48(3), 2004, pp.361-387.

[SW 85] Domenico Sacca and Gio Wiederhold, “Database Partitioning in aCluster of Processors”, ACM Trans. Database Syst. 10(1), 1985, pp. 29-56

[Tan 57] T. T. Tanimoto, “IBM Internal Report”, 1957, pp.637-653.

[TFG 09] Lisa Tan, Farshad Fotouhi, William Grosky, “Improving Similarity Join Algorithms using Vertical Clustering Techniques,” ICADIWT, 2009, pp. 491-496.

[TFGPM 09] Lisa Tan, Farshad Fotouhi, William Grosky, Horia F. Pop, Noureddine Mouaddib, “Improving Similarity Join Algorithms Using Fuzzy Clustering Technique, ” IEEE International Conference on Data Mining, 2009, pp.545-550

[TKM 01] S. Tejada, C. A. Knoblock and S. Minton, “Learning object identification rules for information integration”, Information Systems 26(8), 2001, pp. 607–633.

[WAK 08] Lianzi Wen, Toshiyuki Amagasa, and Hiroyuki Kitagawa, “An approach for XML similarity join using tree serialization”, Proceedings of the 13th international conference on Database systems for advanced applications, India, 2008, pp.562-570

[Wang 10] Wei Wang, “Similarity Joins as Stronger Metric Operations”, SIGSPATIAL, Vol. 2, 2010, pp.24-27

[Wiki 09] Wikipedia, http://en.wikipedia.org/wiki/String_metric, 2009

[Win 99] W. E. Winkler, “The state of record linkage and current research problems”, Statistics of Income Division, Internal Revenue Service Publication, Vol. 4, 1999, pp. 350-365

[WXLZ 09] W. Wang, C. Xiao, X. Lin, and C. Zhang, “Efficient approximate entity extraction with edit constraints”, In *SIMGOD*, 2009, pp.759-770

[WZZ 05] Xindong Wu, Chengqi Zhang and Shichao Zhang, “Database Classification for Multi-Database Mining,” Information Systems, vol. 30, 2005, pp. 71-88.

[XB 91] L. X. Xie and G. Beni, “Validity Measure for Fuzzy Clustering,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 3, no. 8, 1991, pp. 841-847.

[XWL 08] Chuan Xiao, Wei Wang, and Xuemin Lin, “Ed-Join: An Efficient Algorithm for Similarity Joins With Edit Distance Constraints”, VLDB, 08, pp.933-944.

[XWLS 09] Chuan Xiao, Wei Wang, Xuemin Lin, Haichuan Shang, “Top-k Set Similarity Joins”, Proceeding of ICDE, 2009 pp. 916–927.

[Zah 65] L. Zaheh. “Fuzzy sets,” Information and Control, vol. 8, 1965, pp. 338-352.

[ZRK 96] T. Zhang, R. Ramakrishnan, and M. Livny, BIRCH: “An Efficient Data Clustering Method for Very Large Databases,” Proceedings of ACM SIGMOD Conference, Montreal, Canada, 1996, pp. 103–114.

[ZWZ 03] Shichao Zhang, Xindong Wu and Chengqi Zhang, “Multi-Database Mining,” IEEE Computational Intelligence Bulletin, vol. 2, no. 1, June 2003, pp. 5-13.

ABSTRACT

CLUSTERING-BASED PRE-PROCESSING APPROACHES TO IMPORVE SIMILARITY JOIN TECHNIQUES

by

YUFEN (LISA) TAN

May 2011

Advisor: Dr. Farshad Fotouhi

Major: Computer Science

Degree: Doctor of Philosophy

Research on similarity join techniques is becoming one of the growing practical areas for study, especially with the increasing E-availability of vast amounts of digital data from more and more source systems. This research is focused on pre-processing clustering-based techniques to improve existing similarity join approaches.

Identifying and extracting the same real-world entities from different data sources is still a big challenge and a significant task in the digital information era. Dissimilar extracts may indeed represent the same real-world entity because of inconsistent values and naming conventions, incorrect or missing data values, or incomplete information. Therefore discovering efficient and accurate approaches to determine the similarity of data objects or values is of theoretical as well as practical significance.

Semantic problems are raised even on the concept of similarity regarding its usage and foundation. Existing similarity join approaches often have a very specific view of similarity measures and pre-defined predicates that represent a narrow focus on the context of similarity for a given scenario. The predicates have been assumed to be a group of clustering [MSW 72] related attributes on the join. To identify those entities for data integration purposes requires a broader view of similarity; for instance a number of generic similarity measures are useful in a given data integration systems.

This study focused on string similarity join, namely based on the Levenshtein or edit distance and Q-gram. Proposed effective and efficient pre-processing clustering–based techniques were the focus of this study to identify clustering related predicates based on either attribute value or data value that improve existing similarity join techniques in enterprise data integration scenarios.

AUTOBIOGRAPHICAL STATEMENT

YUFEN "LISA" TAN

EDUCATIONS:

2011	Ph.D.	Department of Computer Science Wayne State University, Detroit, MI
2000	M.S.	Department of Computer Science Wayne State University, Detroit, MI
1999	B.S.	Computer & Information Science Queen's University, Ontario, CA
1986	B.E.	Department of Automation Control Central South University of Technology Changsha, China

PROFESSIONAL EXPERIENCES:

2001 - Current	Lead System Integrator, Computing & Information Technology, Wayne State University, Detroit, MI
1986 – 1996	Sr. Researcher, Guangzhou Nonemet Research Institute, Guangzhou, China