Volume 5, Issue 11, November 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Study of Progressive Techniques for Efficient Duplicate Detection

S. Ramya*
M.E(II), Computer Science and Engineering,
Vivekanandha College of Engineering for Women,
Thiruchencode, Namakkal, Tamilnadu, India

C. Palaninehru
Assistant Professor, Computer Science and Engineering,
Vivekanandha College of Engineering for Women,
Thiruchencode, Namakkal, Tamilnadu, India

Abstract----Databases contains very large datasets, where various duplicate records are present. The duplicate records occur when data entries are stored in a uniform manner in the database, resolving the structural heterogeneity problem. Detection of duplicate records are difficult to find and it take more execution time. In this literature survey papers various techniques used to find duplicate records in database but there are some issues in this techniques. To address this Progressive algorithms has been proposed for that significantly increases the efficiency of finding duplicates if the execution time is limited and improve the quality of records.

Keywords---- Duplicate record detection,Sorted Neighborhood method, Blocking, Entity resolution.

I. INTRODUCTION

Databases play an important role in today's IT based economy. Many industries and systems depend on the accuracy of databases to carry out operations. Therefore, the quality of the information stored in the databases, can have significant cost implications to a system that relies on information to function and conduct business. In an error-free system with perfectly clean data, the construction of a comprehensive view of the data consists of linking --in relational terms, joining-- two or more tables on their key fields. Unfortunately, data often lack a unique, global identifier that would permit such an operation. Furthermore, the data are neither carefully controlled for quality nor defined in a consistent way across different data sources. Thus, data quality is often compromised by many factors, including data entry errors (e.g.,studet instead of student), missing integrity constraints (e.g., allowing entries such as EmployeeAg*e=567*), and multiple conventions for recording information To make things worse, in independently managed databases not only the values, but the structure, semantics and underlying assumptions about the data may differ as well.

Progressive duplicate detection identifies most duplicate pairs early in the detection process. Instead of reducing the overall time needed to finish the entire process, progressive approaches try to reduce the average time after which a duplicate is found. Progressive techniques make this trade-off more beneficial as they deliver more complete results in shorter amounts of time. Progressive Sorted Neighborhood method take clean dataset and find some duplicate records and Progressive Blocking take dirty datasets and detect large duplicate records in databases.

II. LITERATURE SURVEY

Steven Euijong Whang, David Marmaros,[1] Entity resolution (ER) is the problem of identifying which records in a database refer to the same entity. In practice, many applications need to resolve large data sets efficiently, but do not require the ER result to be exact. For example, people data from the Web may simply be too large to completely resolve with a reasonable amount of work. As another example, real-time applications may not be able to tolerate any ER processing that takes longer than a certain amount of time. This paper investigates how we can maximize the progress of ER with a limited amount of work using "hints," which give information on records that are likely to refer to the same real-world entity. A hint can be represented in various formats, and ER can use this information as a guideline for which records to compare first. We introduce a family of techniques for constructing hints efficiently and techniques for using the hints to maximize the number of matching records identified using a limited amount of work. Using real data sets, we illustrate the potential gains of our pay-as-you-go approach compared to running ER without using hints. An ER process is often extremely expensive due to very large data sets and compute-intensive record comparisons. The proposed a pay-as-you-go approach for Entity Resolution (ER) where given a limit in resources (e.g., work, runtime) we attempt to make the maximum progress possible.

In [11] the World Wide Web is witnessing an increase in the amount of structured content – vast heterogeneous collections of structured data are on the rise due to the Deep Web, annotation schemes like Flickr, and sites like Google Base. While this phenomenon is creating an opportunity for structured data management, dealing with heterogeneity on the web-scale presents many new challenges. In this paper, we highlight these challenges in two scenarios – the Deep Web and Google Base. We contend that traditional data integration techniques are no longer valid in the face of such heterogeneity and scale. We propose a new data integration architecture, PAYGO, which is inspired by the concept of data spaces and emphasizes pay-as-you-go data management as means for achieving web-scale data integration.

[13] Similarity join is a useful primitive operation underlying many applications, such as near duplicate Web page detection, data integration, and pattern recognition. Traditional similarity joins require a user to specify a similarity threshold. In this paper, we study a variant of the similarity join, termed top-k set similarity join. It returns the top-k pairs of records ranked by their similarities, thus eliminating the guess work users have to perform when the similarity threshold is unknown. An algorithm, topk-join, is proposed to answer top-k similarity join efficiently. It is based on the prefix filtering principle and employs tight upper bounding of similarity values of unseen pairs. Experimental results demonstrate the efficiency of the proposed algorithm on large-scale real datasets. Given a similarity function, a similarity join between two sets of records returns pairs of records from two sets such that their similarities are no less than a given threshold. In this paper, we study the problem of answering similarity join queries to retrieve top-k pairs of records ranked by their similarities. Existing approaches for the traditional similarity joins with a given threshold will have to make guesses on the similarity threshold and incur much redundant calculation. We propose an efficient algorithm that computes the answers in a progressive manner.

[2] Ahmed K. Elmagarmid ,Panagiotis G.Ipeirotis, Vassilios S.Verykios often, in the real world, entities have two or more representations in databases. Duplicate records do not share a common key and/or they contain errors that make duplicate matching a difficult task. Errors are introduced as the result of transcription errors, incomplete information, lack of standard formats, or any combination of these factors. In this paper, we present a thorough analysis of the literature on duplicate record detection. We cover similarity metrics that are commonly used to detect similar field entries, and we present an extensive set of duplicate detection algorithms that can detect approximately duplicate records in a database. We also cover multiple techniques for improving the efficiency and scalability of approximate duplicate detection algorithms. We conclude with coverage of existing tools and with a brief discussion of the big open problems in the area. The problem that we study has been known for more than five decades as the record linkage or the record matching problem in the statistics community. The goal of record matching is to identify records in the same or different databases that refer to the same real-world entity, even if the records are not identical.

In paper [9] Duplicate detection is the task of identifying all groups of records within a data set that represent the same real-world entity, respectively. This task is difficult, because (i) representations might differ slightly, so some similarity measure must be defined to compare pairs of records and (ii) data sets might have a high volume making a pair-wise comparison of all records infeasible. To tackle the second problem, many algorithms have been suggested that partition the data set and compare all record pairs only within each partition. One well-known such approach is the Sorted Neighbourhood Method (SNM), which sorts the data according to some key and then advances a window over the data comparing only records that appear within the same window. We propose with the Duplicate Count Strategy (DCS) a variation of SNM that uses a varying window size. It is based on the intuition that there might be regions of high similarity suggesting a larger window size and regions of lower similarity suggesting a smaller window size. Next to the basic variant of DCS, we also propose and thoroughly evaluate a variant called DCS++ which is probably better than the original SNM in terms of efficiency (same results with fewer comparisons). In this paper we have examined different strategies to adapt the window size, with the Duplicate Count Strategy as the best performing.

[7]The presence of duplicate records is a major data quality concern in large databases. To detect duplicates, entity resolution also known as duplication detection or record linkage is used as a part of the data cleaning process to identify records that potentially refer to the same real-world entity. We present the Stringer system that provides an evaluation framework for understanding what barriers remain towards the goal of truly scalable and general purpose duplication detection algorithms. In this paper, we use Stringer to evaluate the quality of the clusters (groups of potential duplicates) obtained from several unconstrained clustering algorithms used in concert with approximate join techniques. Our work is motivated by the recent significant advancements that have made approximate join algorithms highly scalable. Our extensive evaluation reveals that some clustering algorithms that have never been considered for duplicate detection, perform extremely well in terms of both accuracy and scalability.

Mauricio a.,Hern andez,Salvatore j. Stolfo[5]The problem of merging multiple databases of information about common entities is frequently encountered in KDD and decision support applications in large commercial and government organizations. The problem we study is often called the Merge/Purge problem and is difficult to solve both in scale and accuracy. Large repositories of data typically have numerous duplicate information entries about the same entities that are difficult to cull together without an intelligent "equational theory" that identifies equivalent items by a complex, domain-dependent matching process. We have developed a system for accomplishing this Data Cleansing task and demonstrate its use for cleansing lists of names of potential customers in a direct marketing-type application. Our results for statistically generated data are shown to be accurate and effective when processing the data multiple times using different keys for sorting on each successive pass. Combing results of individual passes using transitive closure over the independent results, produces far more accurate results at lower cost. The system provides a rule programming module that is easy to program and quite good at finding duplicates especially in an environment with massive amounts of data.

[17]Existing literature in the field of transitive relations focuses mainly on dense, Boolean, undirected relations. With the emergence of a new area of intelligent retrieval, where sparse transitive fuzzy ordering relations are utilized, existing theory and methodologies need to be extended, as to cover the new needs. This paper discusses the incremental update of such fuzzy binary relations, while focusing on both storage and computational complexity issues. Moreover, it proposes a novel transitive closure algorithm that has a remarkably low computational complexity (below O(n2)) for the average sparse relation; such are the relations encountered in intelligent retrieval.

[15]Duplicate detection is the process of finding multiple records in a dataset that represent the same real-world entity. Due to the enormous costs of an exhaustive comparison, typical algorithms select only promising record pairs for comparison. Two competing approaches are blocking and windowing. Blocking methods partition records into disjoint subsets, while windowing methods, in particular the Sorted Neighborhood Method, slide a window over the sorted records and compare records only within the window. We present a new algorithm called Sorted Blocks in several variants, which generalizes both approaches. To evaluate Sorted Blocks, we have conducted extensive experiments with different datasets. These show that our new algorithm needs fewer comparisons to find the same number of duplicates.

Duplicate detection, also known as entity matching or record linkage, is the problem of identifying pairs of records that represent the same real-world entity. An exhaustive duplicate detection process involves computing the similarities of all record pairs, which can be very expensive for large datasets. Therefore, the challenge is to effectively and efficiently search for duplicates.

The performance bottleneck for duplicate detection is typically the expensive attribute comparison with similarity measures between record pairs. To avoid these prohibitively expensive comparisons of all pairs of records, a common technique is to carefully partition the records into smaller subsets and search for duplicates only within these partitions. Two competing approaches are often cited: Blocking methods partition records into disjoint subsets, for instance using zip code as partitioning key. Sorted-neighborhood based methods sort the data according to some key, such as last name, and then slide a window of fixed size across the sorted data and compare pairs only within the window.

III. CONCLUSION

The progressive sorted neighbourhood method and progressive blocking algorithms increase the efficiency of duplicate detection for situations with limited execution time they dynamically change the ranking of comparison candidates based on intermediate results to execute promising comparisons first and less promising later. This paper surveys different research papers that proposed various algorithms for detection of duplicate records. The progressive algorithms of duplicate detection is used to overcome disadvantages in various research papers.

REFERENCES

[1] S. E. Whang, D. Marmaros, and H. Garcia-Molina, "*Pay-as-you-go entity resolution*," IEEE Trans. Knowl. Data Eng., vol. 25, no. 5,pp. 1111–1124, May 2012.

[2] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "*Duplicate record detection: A survey*," IEEE Trans. Knowl. Data Eng., vol. 19,no. 1, pp. 1–16, Jan. 2007.

[3] F. Naumann and M. Herschel**,** *An Introduction to Duplicate Detection*. San Rafael, CA, USA: Morgan & Claypool, 2010.

[4] H. B. Newcombe and J. M. Kennedy, *"Record linkage: Making maximum use of the discriminating power of identifying information,"*Commun. ACM, vol. 5, no. 11, pp. 563–566, 1962.

[5] M. A. Hern_andez and S. J. Stolfo,*"Real-world data is dirty: Data cleansing and the merge/purge problem,"* Data Mining Knowl. Discovery, vol. 2, no. 1, pp. 9–37, 1998.

[6] X. Dong, A. Halevy, and J. Madhavan,*"Reference reconciliation in complex information spaces,"* in Proc. Int. Conf. Manage. Data, 2005, pp. 85–96.

[7] O. Hassanzadeh, F. Chiang, H. C. Lee, and R. J. Miller,*"Framework for evaluating clustering algorithms in duplicate detection,"* Proc. Very Large Databases Endowment, vol. 2, pp. 1282–1293, 2009.

[8] O. Hassanzadeh and R. J. Miller,*"Creating probabilistic databases from duplicated data,"* VLDB J., vol. 18, no. 5, pp. 1141–1166, 2009.

[9] U. Draisbach, F. Naumann, S. Szott, and O. Wonneberg, *"Adaptive windows for duplicate detection,"* in Proc. IEEE 28th Int. Conf. Data Eng., 2012, pp. 1073–1083.

[10] S. Yan, D. Lee, M.-Y. Kan, and L. C. Giles,*"Adaptive sorted neighbourhood methods for efficient record linkage,"* in Proc. 7th ACM/Joint Int. Conf. Digit. Libraries, 2007, pp. 185–194.

[11] J. Madhavan, S. R. Jeffery, S. Cohen, X. Dong, D. Ko, C. Yu, and A. Halevy*, "Web-scale data integration: You can only afford to pay as you go,"* in Proc. Conf. Innovative Data Syst. Res., 2007.

[12] S. R. Jeffery, M. J. Franklin, and A. Y. Halevy,"*Pay-as-you-go user feedback for data space systems,"* in Proc. Int. Conf. Manage. Data,2008, pp. 847–860.

[13] C. Xiao, W. Wang, X. Lin, and H. Shang*"Top-k set similarity joins,"* in Proc. IEEE Int. Conf. Data Eng., 2009, pp. 916–927.

[14] P. Indyk, *"A small approximately min-wise independent family of hash functions,"* in Proc. 10th Annu. ACM-SIAM Symp. Discrete Algorithms, 1999, pp. 454–456. Fig. 10. Duplicates found in the plista-dataset.1328 IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 27, NO. 5, MAY 2015.

[15] U. Draisbach and F. Naumann,*"A generalization of blocking and windowing algorithms for duplicate detection,"* in Proc. Int. Conf.Data Knowl. Eng., 2011, pp. 18–24.

[16] H. S. Warren, Jr.,*"A modification of Warshall's algorithm for the transitive closure of binary relations,"* Commun. ACM, vol. 18,no. 4, pp. 218–220, 1975.

[17] M. Wallace and S. Kollias, "*Computationally efficient incremental transitive closure of sparse fuzzy binary relations,"* in Proc. IEEE Int. Conf. Fuzzy Syst., 2004, pp. 1561–1565.

[18] F. J. Damerau, *"A technique for computer detection and correction of spelling errors,"* Commun. ACM, vol. 7, no. 3, pp. 171–176,1964.

[19] P. Christen, "*A survey of indexing techniques for scalable record linkage and deduplication,"* IEEE Trans. Knowl. Data Eng., vol. 24, no. 9, pp. 1537–1555, Sep. 2012.

[20] B. Kille, F. Hopfgartner, T. Brodt, and T. Heintz, *"The Plista dataset,"* in Proc. Int. Workshop Challenge News Recommender Syst., 2013,pp. 16–23.

[21] L. Kolb, A. Thor, and E. Rahm, *"Parallel sorted neighbourhood blocking with MapReduce,"*in Proc.Conf. Datenbanksysteme in B€uro, Technik und Wissenschaft, 2011.

