ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Comprehensive and Progressive Duplicate Entities Detection

Veerisetty Ravi Kumar
Dept of CSE,
Benaiah Institute of Technology and Science.

Nagaraju Medida
Assistant Professor,
Benaiah Institute of Technology and Science.

ABSTRACT:

Duplicate detection is the process of identifying multiple representations of same real world entities. Today, duplicate detection methods need to process ever larger datasets in ever shorter time: maintaining the quality of a dataset becomes increasingly difficult. We present two novel, progressive duplicate detection algorithms that significantly increase the efficiency of finding duplicates if the execution time is limited: They maximize the gain of the overall process within the time available by reporting most results much earlier than traditional approaches. Comprehensive experiments show that our progressive algorithms can double the efficiency over time of traditional duplicate detection and significantly improve upon related work.

Keywords:

Duplicate Detection, Entity Resolution, Data Cleaning, Progressiveness, Data Separation.

INTRODUCTION:

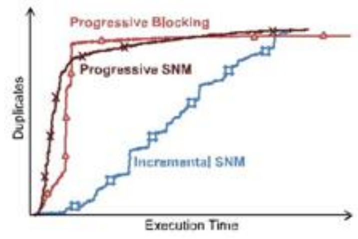
Databases play an important role in today's IT based economy. Many industries and systems depend on the accuracy of databases to carry out operations. Therefore, the quality of the information stored in the databases, can have significant cost implications to a system that relies on information to function 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 EmployeeAge=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.

Data are among the most important assets of a company. But due to data changes and sloppy data entry, errors such as duplicate entries might occur, making data cleansing and in particular duplicate detection indispensable. However, the pure size of today’s datasets renders duplicate detection processes expensive. Online retailers, for example, offer huge catalogs comprising a constantly growing set of items from many different suppliers. As independent persons change the product portfolio, duplicates arise. Although there is an obvious need for reduplication, online shops without downtime cannot afford traditional reduplication. 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. Early termination, in particular, then yields more completes results on a progressive algorithm than on any traditional approach. The incremental algorithm reports new duplicates at an almost constant frequency. This output behavior is common for state-of-the-art duplicate detection algorithms. In this work, however, we focus on progressive algorithms, which try to report most matches early on, while possibly slightly increasing their overall runtime. To achieve this, they need to estimate the similarity of all comparison candidates in order to compare most promising record pairs first. With the pair selection techniques of the duplicate detection process, there exists a trade-off between the amount of time needed to run a duplicate detection algorithm and the completeness of the results. Progressive techniques make this trade-off more beneficial as they deliver more complete results in shorter amounts of time. Furthermore, they make it easier for the user to define this trade-off, because the detection time or result size can directly be specified instead of parameters whose influence on detection time and result size is hard to guess.

We present several use cases where this becomes important: 1) A user has only limited, maybe unknown time for data cleansing and wants to make best possible use of it. Then, simply start the algorithm and terminate it when needed. The result size will be maximized. 2) A user has little knowledge about the given data but still needs to configure the cleansing process. Then, let the progressive algorithm choose window/block sizes and keys automatically. 3) A user needs to do the cleaning interactively to, for instance, find good sorting keys by trial and error. Then, run the progressive algorithm repeatedly each run quickly reports possibly large results. 4) A user has to achieve a certain recall. Then, use the result curves of progressive algorithms to estimate how many more duplicates can be found further; in general, the curves asymptotically converge against the real number of duplicates in the dataset. We propose two novel, progressive duplicate detection algorithms namely Progressive Sorted Neighborhood Method (PSNM), which performs best on small and almost clean datasets, and Progressive Blocking (PB), which performs best on large and very dirty datasets. Both enhance the efficiency of duplicate detection even on very large datasets. In comparison to traditional duplicate detection, progressive duplicate detection satisfies two conditions [1]: Improved Early Quality. Let t be an arbitrary target time at which results are needed. Then the progressive algorithm discovers more duplicate pairs at t than the corresponding traditional algorithm. Typically, t is smaller than the overall runtime of the traditional algorithm. Same Eventual Quality. If both a traditional algorithm and its progressive version finish execution, without early termination at t, they produce the same results. Given any fixed-size time slot in which data cleansing is possible, progressive algorithms try to maximize their efficiency for that amount of time. To this end, our algorithms PSNM and PB dynamically adjust their behavior by automatically choosing optimal parameters, e.g., window sizes, block sizes, and sorting keys, rendering their manual specification superfluous. In this way, we significantly ease the parameterization complexity for duplicate detection in general and contribute to the development of more user interactive applications: We can offer fast feedback and alleviate the often difficult parameterization of the algorithms. In summary, our contributions are the following: • we propose two dynamic progressive duplicate detection algorithms, PSNM and PB, which expose different strengths and outperform current approaches.



EXISTING SYSTEM:

Much research on duplicate detection, also known as entity resolution and by many other names focuses on pair selection algorithms that try to maximize recall on the one hand and efficiency on the other hand. The most prominent algorithms in this area are Blocking and the sorted neighborhood method (SNM). Xiao et al. proposed a top-k similarity join that uses a special index structure to estimate promising comparison candidates. This approach progressively resolves duplicates and also eases the parameterization problem. Pay-As-You-Go Entity Resolution by Whang et al. introduced three kinds of progressive duplicate detection techniques, called "hints"

DISADVANTAGES OF EXISTING SYSTEM:

- A user has only limited, maybe unknown time for data cleansing and wants to make best possible use of it. Then, simply start the algorithm and terminate it when needed. The result size will be maximized.
- A user has little knowledge about the given data but still needs to configure the cleansing process.
- A user needs to do the cleaning interactively to, for instance, find good sorting keys by trial and error. Then, run the progressive algorithm repeatedly; each run quickly reports possibly large results.
- All presented hints produce static orders for the comparisons and miss the opportunity to dynamically adjust the comparison order at runtime based on intermediate results.

PROPOSED SYSTEM:

In this work, however, we focus on progressive algorithms, which try to report most matches early on, while possibly slightly increasing their overall runtime. To achieve this, they need to estimate the similarity of all comparison candidates in order to compare most promising record pairs first. We propose two novel, progressive duplicate detection algorithms namely progressive sorted neighborhood method (PSNM), which performs best on small and almost clean datasets, and progressive blocking (PB), which performs best on large and very dirty datasets. Both enhance the efficiency of duplicate detection even on very large datasets.

We propose two dynamic progressive duplicate detection algorithms, PSNM and PB, which expose different strengths and outperform current approaches. We introduce a concurrent progressive approach for the multi-pass method and adapt an incremental transitive closure algorithm that together forms the first complete progressive duplicate detection workflow. We define a novel quality measure for progressive duplicate detection to objectively rank the performance of different approaches. We exhaustively evaluate on several real-world datasets testing our own and previous algorithms

ADVANTAGES OF PROPOSED SYSTEM:

- Improved early quality
- Same eventual quality
- Our algorithms PSNM and PB dynamically adjust their behavior by automatically choosing optimal parameters, e.g., window sizes, block sizes, and sorting keys, rendering their manual specification superfluous. In this way, we significantly ease the parameterization complexity for duplicate detection in general and contribute to the development of more user interactive applications.

IMPLEMENTATION:

MODULES:

- Dataset Collection
- Preprocessing Method
- Data Separation
- Duplicate Detection
- Quality Measures

MODULES DESCSRIPTION:

Dataset Collection:

To collect and/or retrieve data about activities, results, context and other factors. It is important to consider the type of information it want to gather from your participants and the ways you will analyze that information. The data set corresponds to the contents of a single database table, or a single statistical data matrix, where every column of the table represents a particular variable. After collecting the data to store the Database.

Preprocessing Method:

Data preprocessing or Data cleaning, Data is cleansed through processes such as filling in missing values, smoothing the noisy data, or resolving the inconsistencies in the data. And also used to removing the unwanted data. Commonly used as a preliminary data mining practice, data preprocessing transforms the data into a format that will be more easily and effectively processed for the purpose of the user.

Data Separation:

After completing the preprocessing, the data separation to be performed. The blocking algorithms assign each record to a fixed group of similar records (the blocks) and then compare all pairs of records within these groups. Each block within the block comparison matrix represents the comparisons of all records in one block with all records in another block, the equidistant blocking; all blocks have the same size.

Duplicate Detection:

The duplicate detection rules set by the administrator, the system alerts the user about potential duplicates when the user tries to create new records or update existing records. To maintain data quality, you can schedule a duplicate detection job to check for duplicates for all records that match a certain criteria. You can clean the data by deleting, deactivating, or merging the duplicates reported by a duplicate detection.

Quality Measures:

The quality of these systems is, hence, measured using a cost-benefit calculation. Especially for traditional duplicate detection processes, it is difficult to meet a budget limitation, because their runtime is hard to predict. By delivering as many duplicates as possible in a given amount of time, progressive processes optimize the cost-benefit ratio.

In manufacturing, a measure of excellence or a state of being free from defects, deficiencies and significant variations. It is brought about by strict and consistent commitment to certain standards that achieve uniformity of a product in order to satisfy specific customer or user requirements.

CONCLUSION:

This paper introduced the progressive sorted neighborhood method and progressive blocking. Both 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 comparisons later. To determine the performance gain of our algorithms, we proposed a novel quality measure for progressiveness that integrates seamlessly with existing measures.

Using this measure, experiments showed that our approaches outperform the traditional SNM by up to 100 percent and related work by up to 30 percent For the construction of a fully progressive duplicate detection workflow, we proposed a progressive sorting method, Magpie, a progressive multi-pass execution model, Attribute Concurrency, and an incremental transitive closure algorithm.

The adaptations AC-PSNM and AC-PB use multiple sort keys concurrently to interleave their progressive iterations. By analyzing intermediate results, both approaches dynamically rank the different sort keys at runtime, drastically easing the key selection problem. In future work, we want to combine our progressive approaches with scalable approaches for duplicate detection to deliver results even faster.

In particular, Kolb et al. introduced a two phase parallel SNM [21], which executes a traditional SNM on balanced, overlapping partitions. Here, we can instead use our PSNM to progressively find duplicates in parallel.

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. Hernandez 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 neighborhood methods for efficient record linkage," in Proc. 7th ACM/ IEEE 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 dataspace 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.