Entity Resolution for Longitudinal Studies in Education using OYSTER

E. D. Nelson [1] **and J. R. Talburt** [2]
[1] Department of Information Quality, University of Arkansas, Little Rock, AR, USA
[2] Department of Information Quality, University of Arkansas, Little Rock, AR, USA

Abstract - *This paper describes the application of Oyster, an open source, general purpose entity resolution (ER) system, to the problem of conducting multi-year longitudinal studies of student achievement. Although originally designed to support ER instruction and research, this paper demonstrates that OYSTER can be used in practical applications with processing requirements on the order of one million records, a range that includes many existing small-scale information systems. The paper also discusses an enhancement to the basic R-Swoosh algorithm implemented in OYSTER that allows higher performance in processing student files with high duplication rates.*

Keywords: Entity Resolution, OYSTER Open Source System, R-Swoosh Algorithm, Longitudinal Studies

1 Introduction

Entity Resolution (ER) is the process of determining whether two references to real world objects are referring to the same, or to different, different objects [1]. This is done by successively locating and merging multiple records [2], [3], [4], [5]. References that refer to the same entity are said to be "equivalent references." In ER it is important to understand that entities are real world object that do not exist within information systems. Information systems only contain representations of these objects called entity references [1].

ER can be applied to any type of entity reference. When the underlying object is a customer, whether the customers are individuals or other businesses, is called customer data integration (CDI). CDI allows companies to maintain an accurate, timely, complete and comprehensive representation of a customer across multiple channels, lines of business, and enterprises. Often times, the ability to recognize a customer at any point within the enterprise allows improved customer experience and the ability to up-sale or cross-sale new services or products. If the data is product in nature it is called Product Information Management (PIM). This type of process is integral to large reseller and wholesale companies that often times product but with different descriptive attributes.

Newcombe, Kennedy, Axford, & James [6] state that ER consists of two steps locating or prospecting the records to be used in the merge step, then a comparison step to match the records. They specifically look at ways to optimize both the location and merge steps. Fellegi and Sunter [7] give the statistical basis for linking data. Newcombe, Fair, and Lalonde [8] provide a history of probabilistic record linkage and show empirical results. The SERF project [9] looks at a generic description of ER. They treat the match and merge processes as generic black boxes and describe a series of algorithms to resolve sets of references.

2 Oyster

OYSTER is an open source project sponsored by Center for Advance Research in Entity Resolution and Information Quality (ERIQ) and the University of Arkansas at Little Rock. OYSTER was originally designed to support instruction and research in ER by allowing users to configure its entire operation through XML scripts executed at run-time. The resolution engine of the current version (3.0) can be configured to run in several modes including record-linking/merge-purge, identity resolution, identity capture, and identity update. In addition the system's attributes and identity rules are defined with run-time scripts, as well as, the location and type of each reference source to be processed. OYSTER source code and documentation is freely available from the ERIQ website (1).

3 Problem Definition

We are working with an organization that will be providing data to researchers who request data. The organization receives multiple types of files from different state agencies. The initial files that were a part of this study are the student enrollment files for all of the public schools within the state. There is a total of seven years of data totaling 3.8 million records. There are also test scores, ACT records, BMI records and other records. Within these records a student can be seen across multiple years, i.e. students entering, exiting, and being promoted to higher grades. Student can also often be seen multiple times within a year, i.e. student moves for school district A to district W, the students parents get married or divorced necessitating a name change, etc. All these issues create a large level of

duplication between the files. It is the implications of this large amount of duplication on the effects of Entity Resolution processing that is the focus of this paper.

In resolving the data files 11 simple rules (Table 1) were used to determine equivalence. It was also decided to use R-Swoosh [10] to ensure that all possible merges where processed. When each file was run alone R-Swoosh worked very well. But it was noticed that when the files where run together there was a larger number of R-Swoosh iterations. The multiple processing or churn was causing excessively long run times. Initially it was thought that the implementation of the R-Swoosh algorithm was incorrect but it was determine that it was correct and that the problem was a combination of how R-Swoosh processed records and the fact that there was a large percentage of duplication between the files. This was further compounded by the distance between many of the duplications, i.e. one file would have to be completely processed before the duplicates in the next file would be seen.

	First Name	Last Name	First Middle Name	Full Name	Date of Birth	SSN
Rule 1	Exact Ignore Case	Exact Ignore Case			Exact	Exact
Rule 2				Exact Ignore Case	Exact	Exact
Rule 3	QTR(0.25)	Exact Ignore Case			Exact	Exact
Rule 4			QTR(0.25)	Exact Ignore Case	Exact	Exact
Rule 5		Exact Ignore Case			Exact	Exact
Rule 6	Exact Ignore Case				Exact	Exact
Rule 7	SUBSTRLEFT(5)	SUBSTRLEFT(5)			Exact	Exact
Rule 8	Exact Ignore Case	Exact Ignore Case				Exact
Rule 9	Exact Ignore Case	Exact Ignore Case			Exact	
Rule 10		Hyphenated			Exact	Exact
Rule 11					Exact	Exact

Table 1. Identity Rules used for all Runs

4 R-Swoosh

According to the SERF project, R-Swoosh uses two sets R and R'. R is the set of all input records and R' is the set of all non matched records. Records from R are compared to R' in a pair wise fashion. If a match is found the records are merged. The matched record is removed from R' and the merged record is placed on the bottom of R. This continues until all records have been removed from R. In this respect R-Swoosh is easy to implement and it avoids some unnecessary comparisons because of the merge and replacement of the initial entity references with the merged value.

Supposed that you have the records shown in Table 2, for a record to match at least four attribute fields must be an exact match. Merging is a simple merge of each attribute field into an attribute field set. Using R-Swoosh and processing each record in order none of the records match until record 7 is read. It then takes an additional 6 matches to determine that all 7 references refer to the same entity. In fact it doesn't matter which order the records are in it will always take 7 matches. A walk through in Listing 1 shows the entire progression.

	Field1	Field2	Field3	Field4			Field1	Field2	Field3	Field4
1	A	B	C	D		Rule 1	Exact	Exact	Exact	
2	D	E	F	G		Rule 2	Exact	Exact		Exact
3	A	B	F	G		Rule 3	Exact		Exact	Exact
4	D	E	C	D		Rule 4		Exact	Exact	Exact
5	A	E	F	D						
6	D	B	C	G						
7	A	E	F	G						

Table 2. Example Data and Rules

1. Read in record 1, No match, New Entity
2. Read in record 2, No match, New Entity
3. Read in record 3, No match, New Entity
4. Read in record 4, No match, New Entity
5. Read in record 5, No match, New Entity
6. Read in record 6, No match, New Entity
7. Read in record 7, Matches Record 2 on Rule 4, Merge records and put on bottom of the list as RS1.
8. Read in record RS1, Matches Record 5 on Rule 1, Merge records and put on bottom of the list as RS2.
9. Read in record RS2, Matches Record 4 on Rule 2, Merge records and put on bottom of the list as RS3.
10. Read in record RS3, Matches Record 6 on Rule 3, Merge records and put on bottom of the list as RS4.
11. Read in record RS4, Matches Record 1 on Rule 1, Merge records and put on bottom of the list as RS5.
12. Read in record RS5, Matches Record 3 on Rule 1, Merge records and put on bottom of the list as RS6.
13. Read in record RS6, No match, New Entity
14. No more records, end

# of Consolidation Steps: 6

Listing 1. R-Swoosh applied to example data

A simple example shows the benefits of this slight change. In the enhanced R-Swoosh method, again no records are matched until record 7 is read. At that time records 2, 3 & 5 are all found to match. At that point they are all merged with record 7 and put on the bottom of the list as RS1. The next record read is RS1 since it is the only record in the list and it is found to match to 1, 4, & 6 (Listing 2).

1. Read in record 1, No match, New Entity
2. Read in record 2, No match, New Entity
3. Read in record 3, No match, New Entity
4. Read in record 4, No match, New Entity
5. Read in record 5, No match, New Entity
6. Read in record 6, No match, New Entity
7. Read in record 7, Matches Record 2, 3 & 5, Merge records and put on bottom of the list as RS1.
8. Read in record RS1, Matches Record 1, 4 & 6, Merge records and put on bottom of the list as RS2
9. Read in record RS2, No match, New Entity
10. No more records, end

# of Consolidation Steps: 2

Listing 2. Enhanced R-Swoosh Applied to Example Data

5 R-Swoosh Enhanced

The enhanced version uses three sets but instead of reading in all the records into R, records are read one at a time from the input stream (R). This initially reduces the memory requirements needed since R is transient by nature. Records that are not matched are placed in R'. Because ER is expensive [10] a simple inverted index is used to reduce any unnecessary comparisons. R-Swoosh compares records in a pair wise fashion. In the R-Swoosh algorithm, when a match is found the records are combined and they are both placed back into the record queue at the end to be recheck at some later time for additional matches. Matching continues until the queue is emptied. The enhanced version makes one slight change; with the use of the index it pulls back a candidate list of all possible matches to R'. If multiple records are found to match they are all merged with the input record, removed from R' and placed on R''. R'' is used to hold all records that have been matched and merged. In R-Swoosh these are placed on the end of R but since R in the enhanced version is a stream R'' represents the end of the stream. Once the stream is extinguished R'' takes the place of R.

This slight change was able to reduce the number if iterations from 6 to 2. But here order is important, move the glue record (record 7) to a different point in the file can change the number of iterations required to fully resolve the entity. Even then the number of consolidation steps is less than regular R-Swoosh. If we take the permutation of the order of these 7 records we get 5040 different dataset. Running R-Swoosh on each one shows that order is unimportant as they all return 6 consolidation steps. If we run the enhanced R-Swoosh on the same data sets we find that in 3,168 (62.9%) of the data sets it takes 3 consolidation steps and 1,872 (37.1%) it takes 2 consolidation steps. But in all cases this requires fewer steps than R-Swoosh to produce the same results.

6 Results

To determine how the Enhanced R-Swoosh compares to R-Swoosh, three files containing First and Last Name, SSN and DOB were processed with the Oyster system. The files were run separately and then together using both the R-Swoosh and Enhanced R-Swoosh methods. File A contained 588,279 records and file B contains 463,405 records and file C contains 585,409 records. Each test is run on a Dell Server running Linux (CentOS release 5.5) using 8 GB of main memory and Java 1.6 VM.

Run	Total Records	Clusters	Max Cluster Size	Min Cluster Size > 1	Average Cluster Size	Num. of Duplicate Records	Duplication Rate	Elapsed Seconds	
								R-Swoosh	Enhanced R-Swoosh
File A	588,279	523,068	9	2	1.12467	120,301	11.085%	4,818	5,129
File B	463,405	462,815	3	2	1.00127	1,177	0.127%	2,254	2,286
File C	585,409	528,714	8	2	1.10723	105,821	9.685%	5,094	4,981
File A & B	1,051,684	581,569	9	2	1.8084	893,750	44.701%	13,521	13,351
File A & C	1,173,688	654,135	12	2	1.7943	945,006	44.267%	17,193	17,238
File B & C	1,048,814	558,915	9	2	1.8765	930,615	46.710%	13,369	13,679
All Files	1,637,093	655,861	13	2	2.4961	1,477,165	59.937%	30,134	27,634

Table 3. Comparison of Regular and Enhanced R-Swoosh Performance

Running each file individually using R-Swoosh, reasonable consistent times when the duplication rate is fairly low. Duplication rate is calculated by dividing the clusters by the total records and subtracting from 1. But when the duplication rate increases, the run times also increase. This is due to the fact that R-Swoosh only matches one record at a time, and when it finds a match, puts the merged record back on the bottom of the set. The extended R-Swoosh, does much better on the higher duplication rates as can be seen in Table 3. There was a difference in run time by 2500 seconds a time savings of 8.3%. Interestingly, the Enhanced R-Swoosh under performs when the duplication is low actually taking longer to run. It is believed that the cost of the more complex coding is what is causing this increase. Two, other anomalies where noticed when Files AC and CB were processed it was found that the Enhanced R-Swoosh took longer to resolve these records. It is unknown what caused these two ER processes to run slightly longer. We are not running on a dedicated machine so there is the possibility that there was resource contention issue.

7 Conclusion and Future Work

In using the Enhanced version of R-Swoosh during the project we have seen several gains in processing speed while producing the same result. Based on the empirically results that we are seeing, Enhanced R-Swoosh seems to be a viable alternative to the standard R-Swoosh ER algorithm. But as can be seen in the results section there are some differences that are not completely understood. Future work includes determining in what circumstances Enhanced R-Swoosh should be used, determining the amount of duplication that is necessary for efficient ER with Enhanced R-Swoosh and refining the implementation to allow for a more efficient matching.

8 Acknowledgment

The research described in the paper was supported in part by a grant from the Arkansas Department of Education.

9 References

[1] Talburt, J. R. (2011). *Entity Resolution and Data Quality.* Morgan Kaufman.

[2] Benjelloun, O., Garcia-Molina, H., Gong, H., Kawai, H., Larson, T.E., Menestrina, D., Thavisomboon, S., (2007). "D-Swoosh: A Family of Algorithms for Generic, Distributed Entity Resolution", *Proceedings of the 27th International Conference on Distributed Computing Systems*; Retrieved from HYPERLINK "http://infolab.stanford.edu/serf/" http://infolab.stanford.edu/serf/

[3] Bhattacharya, I., Getoor, L., (2007). "Collective entity resolution in relational data", *ACM Transactions on Knowledge Discovery from Data (TKDD)*; 1(1)

[4] Bilgic, M., Licamele, L., Getoor, L., Shneiderman, B., (2006). "D-Dupe: An Interactive Tool for Entity Resolution in Social Networks", *IEEE Symposium on Visual Analytics Science and Technology*; Retrieved from HYPERLINK "http://infolab.stanford.edu/serf/" http://infolab.stanford.edu/serf/

[5] Garcia-Molina, H., (2006). "Pair-Wise entity resolution: overview and challenges", *Proceedings of the 15th ACM international conference on Information and knowledge management;* 1(1)

[6] Newcombe H. B., Kennedy J. M., Axford S. J. and James A. P., (1959). "Automatic Linkage of Vital Records", *Science New Series*; 130(3381):954-959

[7] Fellegi, I.P., & Sunter, A.B. (1969). A theory for record linkage. *Journal of the American Statistical Association*, 64(328), 1183-1210.

[8] Newcombe H.B., Fair M.E., Lalonde, P. (1992). "The Use of Names for Linking Personal Records", Journal of the American Statistical Association; 87(420)

[9] Benjelloun O., Garcia-Molina H., Kawai H., Larson T.E., Menestrina D., Su Q., Thavisomboon S., Widom J., (2006). "Generic Entity Resolution in the SERF Project", *IEEE Data Engineering Bulletin*; Retrieved from HYPERLINK "http://infolab.stanford.edu/serf/" http://infolab.stanford.edu/serf/

[10] Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S. E., & Widom, J. (2009). "Swoosh: A generic approach to entity resolution". *The VLDB Journal , 18* (1), 255-276.