Sampling Selection Strategy for Large Scale Deduplication in a Distributed System Using Apache Spark

Ritu Yadav,Neha Kumari, Samarth Varshney
Department of Computer Science and Engineering
National Institute of Technology, Warangal

Abstract— The generation of information from a wide range of sources has opened opportunities for the emergence of several new applications such as digital libraries, media streaming etc. that presuppose high quality data to provide reliable services. Data quality is degraded due to the presence of duplicate pairs. So, *data deduplication* task is necessary to detect and remove duplicates and provide efficient solutions to this problem. In very large datasets, it is very difficult to produce the labeled set from the information provided by the user. A *Two-stage sampling selection strategy (T3S)* that selects a reduced set of pairs to tune the deduplication process in large datasets has been proposed and is implemented in a distributed environment using Apache Spark. Thus, T3S reduces the labeling effort substantially while achieving superior matching quality when compared with state-of-the-art deduplication methods in large datasets. Also, performing the deduplication in a distributed environment offers a better performance over the centralized system in terms of speed and flexibility. So, in this work, a distributed approach is implemented for the above method using Apache Spark. Also, a comparison is done between T3S and FSDedup. It shows that T3S reduces the training set size by redundancy removal and hence offers better performance than FSDedup.

Index Terms—Apache Spark, Centralized Systems, Data Mining, Deduplication, Distributed Environment,FSDedup,Sampling Selection Strategy

1 Introduction

THERE has been a dramatic growth in the generation of information from a wide range of sources such as mobile devices, streaming media, and social networks. Data quality is also degraded due to the presence of duplicate pairswith misspellings, abbreviations, conflicting data, and redundant entities, among other problems. So, *data deduplication* task is necessary to detect and remove duplicates and provide efficient solutions to this problem. However, in the context of large datasets, it is a difficult task to produce a replica-free repository.

Data deduplication is a specializeddata compressiontechnique for eliminating duplicate copies of repeating data. This technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. In the deduplication process, unique chunks of data, or byte patterns, are identified and stored during a process of analysis. As the analysis continues, other chunks are compared to the stored copy and whenever a match occurs, the redundant chunk is replaced with a small reference that points to the stored chunk.

Deduplication may occur in-line, as data is flowing, or post-process, after it has been written. With post-process deduplication, new data is first stored on the storage device and then a process at a later time willanalyzethe data looking for duplication. This is the process where the deduplication hash calculations are created on the target device as the data enters the device in real time. If the device spots a block that it already stored on the system it does not store the new block, just references to the existing block.

A typical deduplication method is divided into three main phases: *Blocking* which aims at reducing the number of comparisons by grouping together pairs that share common features, *Comparison* which quantifies the degree of similarity between pairs belonging to the same block, by applying some type of similarity function (e.g. Jaccard, Levenshtein, Jaro) and *Classification* which identifies which pairs are matching or nonmatching. This phase can be carried out by selecting the most similar pairs by means of global thresholds, usually manually defined [1], [2], [3], [4] or learnt by using a classification model based on a training set. The blocking and classific*ation phases in the large scale* deduplication typically rely on the user to configure or tune the process. For instance, the classification phase usually requires a manually labeled training set. However, selecting and labeling a representative training set is a very costly task which is often restricted to expert users.

A number of solutions were formulated and proposed for the solutions of problem associated with data deduplication.Active learning approaches have been proposed to alleviate this problem by helping to select the most informative pairs [5], [6], [7]. *Active learning* can reduce considerably the number of pairs to be manually labeled when compared to randomselection in order to produce competitive effectiveness [7]. *Classifier committees* have been used in active learning to allow deduplication approaches to identify informative pairs based on divergences between the committee members [6], [7]. However, in the initial stages, these approaches still require a minimum training set (which is usually not small) and the definition of some thresholds to allow the classifiers to be learnt, thus still relying in considerable efforts from the ex-

perts.

Record deduplication studies have offered a wide range of solutions exploiting *supervised, semi supervised, and unsupervised strategies*. Supervised and unsupervised strategies rely on expert users to configure the deduplication process. The former assumes the presence of a large training set consisting of the most important patterns present in the dataset (e.g., [8], [9]). The latter relies on threshold values that are manually tuned to configure the deduplication process (e.g., [1], [2], [10], [4]). An alternative active learning method for deduplication was proposed in [5], where the objective is to maximize the recall under a precision constraint. The approach however, increases the manual effort [6].

One of the method for the problem of data deduplication is *FSDedup*. The FSDedup framework [11] has been designed to select the ―close-to-optimum configuration for large scale deduplication tasks with reduced user effort.FS-Dedup was demonstrated to be more effective than manually tuned methods, while still reducing labeling efforts. However, the resulting subsamples may still be composed of redundant pairs, with negative impacts in the labeling effort.

Each of the various methods formulated for solving the problem of data deduplication described above requires a lot of user effort. Thus, *in order to reduce the user effort* in the main deduplication steps (e.g., blocking and classification) as well to *increase the performance/speed* of the system, T3S method is designed to work in a *distributed environment using Apache Spark.*

Section 2 states the problem description in brieffollowed by section 3 in which proposed system design is explained, including the basic framework and all design modules and their constraints. Section 4 includes the flow diagram in distributed environment and the implementation details of all modules. Section 5 presents the results, analysis and comparison of the results with standard algorithms. Section 6 concludes the paper with future work.

2 Problem Description

Since data deduplication task is necessary to detect and remove duplicates and each of the methods proposed till now requires a lot of manual effort and relatively provides a poor performance,thus, a new method, T3S has been proposed which consists of two stages and the related algorithms are designed in a distributed environment using Apache Spark in order to overcome the performance issues faced in a centralized approach.

The new method introduced aims at reducing the redundancy in the subsamples.T3S is able to select a very small, non-redundant and informative set of examples with high effectiveness for large scale datasets. Then, a rule-based active sampling strategy, which requires no initial training set, is incrementally applied to the selected subsamples to reduce redundancy. The two steps of T3S are complementary. While the second stage helps to remove redundancy, the first stage allows the second one to concentrate on the ―most promising portions of the search space for the most informative pairs to be labeled. The final reduced training set produced by T3S is then integrated with FS-Dedup framework to efficiently identify the position of the fuzzy region and configure suitable strategies to classify the most ambiguous pairs.

FS-Dedup depends on subsamples that may include redundant information, representing a waste of manual effort. In contrast, T3S method applies the SSAR active incrementally in all the subsamples to produce the training set. T3S selects a small set composed of the most informative pairs and reduces the final training set size to a much greater extent than FS-Dedup.

To reduce the user effort in the main deduplication steps (e.g. blocking and classification), T3S [15] employs two steps. Given a very large set of records (each containing the same attributes) in a file. First, a strategy is employed to identify the blocking threshold, and thus produce the candidate pairs as introduced in [11]. In the second stage, the redundant information that is selected in the subsamples is removed by means of a rule-based active sampling [12]. These two steps work together to detect the boundaries of the fuzzy region. In [15], a centralized work has been proposed which is not flexible and *scalable* and also slow.

In this work, we will implement the above framework in the *distributed environment* since distributed system offers various advantages over centralized system as stated below:

- networked computing systems offer a better price/performance ratio than centralized systems
- redundancy increases availability when parts of a system fail
- applications that can easily be run simultaneously also offer benefits in terms of faster performance than centralized solutions
- distributed systems can be extended through the addition of components, thereby providing better scalability compared to centralized systems.

3 Proposed Model

3.1.Terminologies

Sig-Dedup has been used to efficiently handle large deduplication tasks. It maps the dataset strings into a set of signatures to ensure that similar substrings result in similar signatures. The signatures are computed by means of the inverted index method. To overcome the drawback of quadratic candidate generation [13], prefix filtering [14] is used.The prefix filtering is formally defined below:

Definition 1: Assume that all the tokens in each record are ordered by a global ordering. Let prefix of a record be the first p tokens of the record. If Jaccard(x,y) ≥ t, then the (p)-prefix of x and (p)-prefix of y must share at least one token. where, Jaccard(x,y) is defined as: $\frac{|x \cap y|}{|x \cup y|}$(x,y) =

Prefix length of each record u is calculated as $|u| - t \cdot \lceil |u| \rceil + 1$, where t= Jaccard similarity threshold.

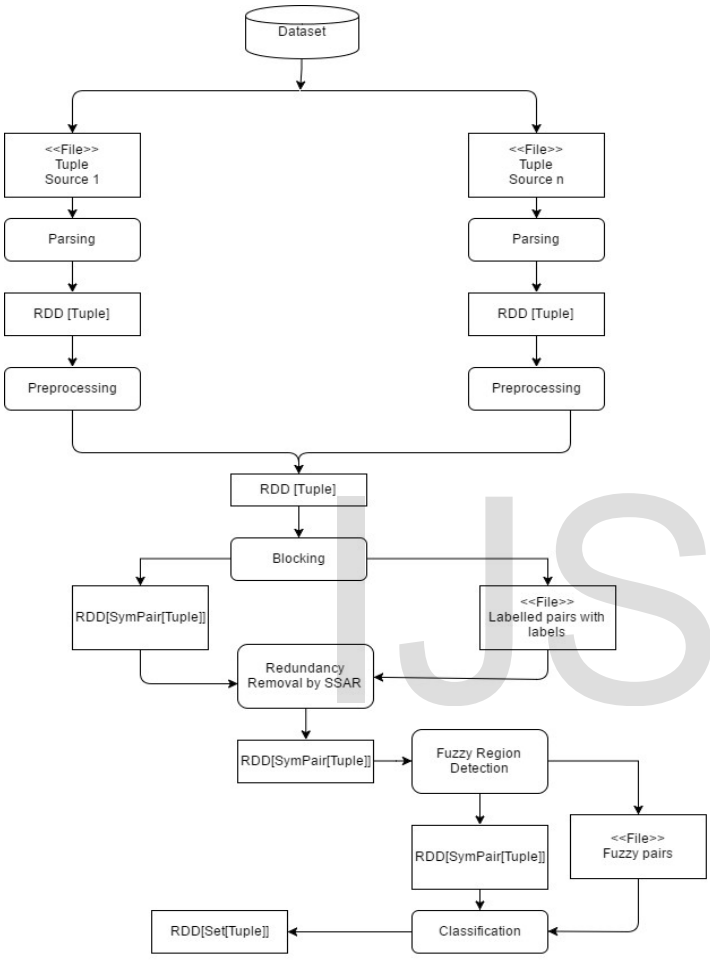

Fig 1: Data Deduplication in distributed System using T3S

3.2Framework

The framework for large scale deduplication using the two stage sampling selection strategy is illustrated in Figure 1. First, the candidate pairs are produced after identifying the blocking threshold. Next, T3S strategy is employed. In its first stage, T3S produces small balanced subsamples of candidate pairs. In the second stage, the redundant information that is selected in the subsamples is removed by means of a rule-based active sampling. These two steps work together to detect the boundaries of the fuzzy. Finally, the classification approach is introduced which is configured by using the pairs manually labeled in the two stages.

All these steps are implemented in the distributed environment using Apache Spark.

3.2.1.Determining Blocking Threshold

In large datasets, it is not feasible to run the Sig-Dedup filters with different thresholds due to the high computational costs. So, a stopping criterion is introduced. The method employed is defined as:

Definition 2: Consider a subset S, created from a randomly sampled dataset D and a range of thresholds with fixed step, th_j = 0.2, 0.3, ..., and 0.9. The subset S is matched using each threshold value th_j. The initial threshold will be the first th_j that results in a number of candidate pairs smaller than the number of records in S.

After finding the global initial threshold value for the blocking process, the entire dataset is matched to create the set of candidate pairs.

3.2.2. First stage of T3S: Sample Selection Strategy

The first stage of T3S adopts the concept of levels to allow each sample to have a similar diversity to that of the full set of pairs. The ranking, created by the blocking step, is fragmented into 10 levels (0.0-0.1,0.1-0.2, 0.2-0.3, ..., and 0.9-1.0), by using the similarity value of each candidate pair. The similarity value of each candidate pair is found using Jaccard similarity. This fragmentation produces levels composed of different matching patterns to prevent non-matching pairs dominating the sample.

3.2.3. Second Stage of T3S: Redundancy Removal

Several pairs selected inside each level are composed of redundant which does not help to increase the training set diversity. Selective Sampling using Association Rules is used to remove redundancy in the information randomly selected.

3.2.3.1.SSAR Method

The second stage of T3S aims at incrementally removing the non-informative or redundant pairs inside each sample level by using the SSAR (Selective Sampling using Association Rules) active learning method [12]. In the beginning, when the training set D is empty, SSAR selects the pair that shares most feature values with all other unlabeled pairs to initially compose the

training set. SSAR selects an unlabeled pair u_i for labeling by using inferences about the number of association rules produced within a projected training set specific for u_i. The projected training set is produced by removing from the current training set D instances and features that do not share features

values with u_i. When compared with the current training set, the unlabeled pair with less classification rules over the projected training set represents the most informative pair. If this pair is not already present in the training set, it is labeled by the user and inserted into the training set. After this, a new round is performed and the training set must be re-projected for each remaining unlabeled pair to determine which one is most dissimilar when compared to the current training set. If the selected pair is already present in the training set, the algorithm converges.

ALGORITHM 1:SSAR METHOD

Require: Unlabeled set T and $\sigma_{min}(\approx 0)$

Ensure: The training set D

1: **while** true **do**

2: **for all** $u_i \in T$ **do**

3: $D_{u_i} \leftarrow$ projected according to u_i

4: $R_{u_i} \leftarrow$ extract useful rules from D_{u_i}

5: **end for**

6: **if** $D = \emptyset$ **then**

7: $\lambda_{u_i} \leftarrow u_i$ such that u_i is the most representative item of T.

8: **else**

9: $\lambda_{u_i} \leftarrow u_i$ such that $u_j \forall$: $|R_{u_i}| \leq |R_{u_j}|$

10: **end if**

11: **if** $\lambda_{u_i} \in D$ **then**

12: **break**

13: **else**

14: LabelPair (λ_{u_i})

15: $D \leftarrow D \cup \{\lambda_{u_i}\}$

16: **end if**

17: **end while**

3.2.3.2. Computational Complexity

The computational complexity of SSAR is $O(S * |U| * 2^m)$, where —S is the number of pairs selected to be labeled, —|U represents the total number of candidate pairs and —mis the number of features. —|U| pairs must be re-projected each time that a labeled pair is attached to the current training set, producing a computationally unfeasible time to process large datasets.

3.2.4. Fuzzy Region Detection

Definition 3: Let Minimum True Pair-(MTP) represent the matching pair with the lowest similarity value among the set of candidate pairs.

Definition 4: Let Maximum False Pair-(MFP) represent the non-matching pair with the highest similarity value among the set of non-matching pairs.

The fuzzy region is detected by using manually labeled pairs. The user is requested to manually label pairs that are selected incrementally by the SSAR from each level. First, SSAR is invoked to identify the informative pairs incrementally inside each level to produce a reduced training set. The pairs labeled within each level are used to identify the MFP and MTP pairs. MTP and MFP pairs define the fuzzy region boundaries.

The similarity value of the MTP and MFP pairs identifies α and β values. The pairs belonging to the fuzzy region are sent to the Classification Step.

3.2.5. Classification

The Classification step aims at categorizing the candidate pairs belonging to the fuzzy region as matching or non-matching. The classifier, T3S-NGram maps each record to a global sorted token set and then applies both the Sig-Dedup filtering and a defined similarity function (such as Jaccard) to the sets. The NGram Threshold is required to identify the matching pairs inside the fuzzy region using the NGram tokenization.

ALGORITHM 2: CLASSIFICATION

Input: File containing Labeled pairs between MTP and MFP

1: **Recompute** the similarity of each pair using Jaccard similarity alongwith **NGram tokenization**.

Sort the pairs based on similarity values.

for each pair p {

i. if (label(p)=='F' && label(p+1)=='F' && label(p+2)=='F'){
ii. sliding =p
iii. break }

for each pair p=sliding+3 to last {

iv. if(label(p)=='T')
v. NGramTh=similarity(p)
vi. break }

2. The candidate pairs that survive the filtering phase and meet the Ngram threshold value are considered as matching ones.

First, the similarity of each labeled pair is recomputed by means of a similarity function along with the NGram tokenization. After this, the labeled pairs are sorted incrementally by the similarity value and a sliding window with fixed-size N is applied to the sorted pairs. The sliding window is relocated in one position until it detects the last windows with only non-matching pairs. Finally, the similarity value of the first matching pair encountered after the last windows with only non-matching pairs, defines the NGram threshold value. The candidate pairs that survive the filtering phase and meet the Ngram threshold value are considered as matching ones.

4 Experimental Results

In the paper the modules which were discussed in the proposed design model are implemented. The focus of this implementation is on efficiency and time.

4.1 DataSets

We have used both synthetic as well as real datasets to evaluate our framework. The real datasets used are film data, camera specifications and student survey. In the synthetic dataset, the records are synthetically built with 7 attributes: first name, last name, age, gender, state, job and phone number. The camera data contains 500 records with 13 attributes in each record: Model, release date, max resolution, low resolution, effective pixels, zoom wide (W), zoom tele (T), normal focus range, macro focus range, storage included, weight, dimensions and price.

4.2 Experimental SetUp

All the experiments are carried out on the Operating System: Ubuntu 14.04 using Scala Language and the development environment used is Apache Spark 1.6.0.

Observations: For the real dataset: Camera Specifications

The results for Camera data are:Number of records taken in the input file are 500 (approx.) and the number of attributes in each record: 13, then we run our algorithms.

Output of the various steps:

1. Identifying Blocking Threshold Output: After performing this step: the blocking threshold obtained is 0.375

2. Sampling Selection Output: Two files each containing 827 pairs of records, first containing the pairs and their levels and the second containing the pairs, their levels in sorted order and their labels. Table 1. highlights the number of pairs falling in every level.

3. SSAR Output:Output is a file containing less number of pairs (711 pairs) after redundancy removal. Table 2 shows the number of pairs in the levels.

Table 1: Sampling Selection: Number of pairs falling in each level

Level	Number of Pairs
1	0
2	0
3	521
4	167
5	81
6	51
7	0
8	7
9	0
10	0

Table 2: SSAR: number of pairs in the levels

Level	Number of Pairs
3	458
4	142
5	67
6	39
7	5

4. Fuzzy Region Detection: Algorithm applied for fuzzy region detection outputs the values as: MTP = (1,288),MFP = (335,336), alpha = 0.375,beta = 0.692 and thefile containing the number of fuzzy pairs = 698 pairs

5. Classification: For the classification step, the sliding window size is taken as 2. The value of N in finding NGrams is taken as 3. Thus, output we observe as:NGram threshold= 0.467, Number of matching pairs after classification= 32

4.3 T3S vs FSDedup

As compared to FSDedup, T3S reduces the training set size considerably by removing the redundancies using the sampling selection strategy. Table 3. highlights the differences in

the number of pairs generated after each step in using both of these methods. The input file is same in both the cases and contains 30 records.

Table 3: T3S vs FSDedup

	T3S	FSDedup
Level Size	10	10
Number of pairs after first stage	827	827
Number of pairs after SSAR in T3S	711	827
Number of fuzzy pairs	698	811
NGram Threshold	0.467	0.375
Number of matching pairs after classification	32	89

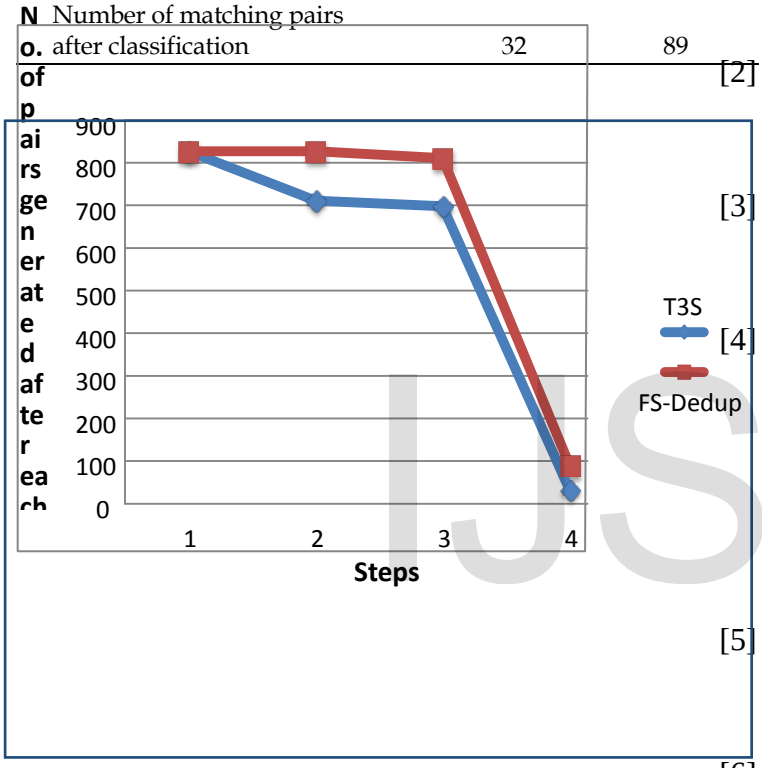

Figure 2: T3S Vs FSDedup

Also, we observed that our designed distributed framework took comparatively less time to process the steps as compared to the centralized framework proposed in [15]. Distributed system framework also offers scalability and can be extended.

5 Conclusion

In this paper, we have proposed a distributed algorithm for large scale deduplication using sampling selection strategy which produces the same result as the centralized system but speeds up the process by a considerable amount. The strategy proposed for the data deduplication in the context of large datasets is a simple approach which reduces the user effort as well as since the algorithms for the proposed method are devised for a distributed system, it increases the performance too.As followed from our experiments, our distributed approach performs the same processes in a lesser time with a greater flexibility and scalability. We have also compared the T3S approach with the FSDedup. T3S reduces the user effort by reducing the training set size and results in a lesser number of matching pairs.

6 References

[1] R. J. Bayardo, Y. Ma, and R. Srikant, —Scaling up all pairs similarity search,in Proc. 16th Int. Conf. World Wide Web, pp. 131–140, 2007.

[2] S. Chaudhuri, V. Ganti, and R. Kaushik, —A primitive operator for similarity joins in data cleaning, in Proc. 22nd Int. Conf. Data Eng., p. 5, Apr. 2006.

[3] J. Wang, G. Li, and J. Fe, —Fast-join: An efficient method for fuzzy token matching based string similarity join, in Proc. IEEE 27th Int. Conf. Data Eng., 2011, pp. 458–469.

[4] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang, —Efficient similarity joins for nearduplicate detection, ACM Trans. Database Syst., vol. 36, no. 3, pp. 15:1–15:41, 2011.

[5] A. Arasu, M. Gotz, and R. Kaushik, —On active learning of record matching packages, in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2010, pp. 783–794.

[6] K. Bellare, S. Iyengar, A. G. Parameswaran, and V. Rastogi, —Active sampling for entity matching, in Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2012, pp. 1131–1139.

[7] S. Sarawagi and A. Bhamidipaty, —Interactive deduplication using active learning, in Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2002, pp. 269–278.

[8] M. G. de Carvalho, A. H. Laender, M. A. Goncalves, and A. S. da Silva, —A genetic programming approach to record deduplication, IEEE Trans. Knowl. Data Eng., vol. 24, no. 3, pp. 399–412, Mar. 2012.

[9] J. Wang, G. Li, J. X. Yu, and J. Feng, —Entity matching: How similar is similar, Proc. VLDB Endow., vol. 4, no. 10, pp. 622–633, Jul. 2011.

[10]R. Vernica, M. J. Carey, and C. Li, —Efficient parallel set-similarity joins using mapreduce, in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2010, pp. 495–506.

[11] G. Dal Bianco, R. Galante, C. A. Heuser, and M. A. Gonal ves, —Tuning large scale deduplication with reduced effort, in Proc. 25th Int. Conf. Scientific Statist. Database Manage., 2013, pp. 1–12.

[12] R. M. Silva, M. A. Gonc¸alves, and A. Veloso, —A two-stage active learning method for learning to rank, J. Assoc. In form. Sci. Technol., vol. 65, no. 1, pp. 109–128, 2014.

[13] M. Bilenko and R. J. Mooney, —On evaluation and training-set construction for duplicate detection, in Proc. Workshop KDD, 2003, pp. 7–12.

[14] A. Arasu, C. R_e, and D. Suciu, —Large-scale deduplication with constraints using dedupalog, in Proc. IEEE Int. Conf. Data Eng., 2009, pp. 952–963.

[15] Guilherme Dal Bianco, Renata Galante, Marcos Andr_e Gonc¸alves, Sergio Canuto, and Carlos A. Heuser, —A Practical and Effective Sampling Selection Strategy for Large Scale Deduplication,IEEE Transactions On Knowledge And Data Engineering, Vol. 27, No. 9, September 2015