Populating Entity Name Systems for Big Data Integration

Mayank Kejriwal

University of Texas at Austin
kejriwal@cs.utexas.edu

Abstract. An Entity Name System (ENS) is a thesaurus for entities. An ENS is a fundamental component of *data integration* systems, serving instance matching needs across multiple data sources. Populating an ENS in support of co-referencing Linked Open Data (LOD) is a Big Data problem. *Viable* solutions to the long-standing *Entity Resolution* (ER) problem are required, meeting specific requirements of *heterogeneity*, *scalability* and *automation*. In this thesis, we propose to develop and implement algorithms for an ER system that address the three key criteria. Preliminary results demonstrate potential system feasibility.

Keywords: Entity Name System, Data Integration, Entity Resolution

1 Problem Statement

Linked Open Data (LOD) has grown exponentially since 2007 [2]. Many entities on the LOD cloud refer to the same logical entity and need to be resolved by linking them using *owl:sameAs* [10]. Given that LOD currently contains billions of triples in *independently* developed data sources, the *Entity Resolution* (ER) problem needs to be both *scalable* and to account for schema or *structural heterogeneity* between sources. Given that the Deep Web is about 500 times larger than the Surface Web [12], an ER solution that works with both RDF and relational (the dominant data model in the Deep Web) would need to also account for *data model heterogeneity*. The continuous growth of LOD mandates *automatic* linking of entities without excessive reliance on domain expertise.

In this thesis, we develop and implement algorithms for an ER system meeting certain requirements of heterogeneity, scalability and automation. We describe these requirements, and propose novel techniques to satisfy them in a *MapReduce* based framework for both RDF and relational data models.

2 Relevancy

The primary motivation of the proposed system is *data integration* [7]. A complete data integration system relies on the population of an ENS component, to resolve equivalent entities occurring in multiple data sources. The generic ER problem is pervase, however, both in the relational database community as

record linkage [8], and in the linked data community as *instance matching* or *link discovery* [10]. The importance of the problem is growing in concert with the Semantic Web, and with Web-scale efforts such as the Okkam ENS [3]. Thus, our solution would not be limited in potential application to data integration alone. Furthermore, the advantage of having a MapReduce [6] implementation is that the prototype can be deployed scalably as a *cloud* workflow on dynamically provisioned clusters. Scalability has emerged as an important issue in recent works on both data and ontology matching [9].

3 Related Work

The longevity of the ER problem has led to a considerable body of work on the subject. In this section, we selectively assess some recent literature from the framework of addressing automation, heterogeneity and scalability criteria. We first survey some techniques developed in the relational community, followed by related efforts in the Semantic Web community.

The ER problem has emerged in several different communities, with a profusion of terminology. Elmagarmid et al. surveys it broadly in the relational setting [8]. Typically, ER is conducted as a sequence of two steps. The first step, *blocking*, attempts to mitigate $O(n^2)$ complexity of pairwise comparison of n records by clustering 'similar' entities into overlapping *blocks* using an inexpensive *blocking scheme*. The second step takes a block as input and applies a similarity function only to pairs of entities *within* blocks. Christen provides a good survey of blocking methods [4]. Intuitively, blocking captures the *scalability* aspect.

Till quite recently, a domain expert had to *manually* specify a blocking scheme. In 2006, two independent *supervised* blocking scheme learners (BSLs) were proposed. These BSLs would learn a blocking scheme from *provided* training examples [1],[20]. Given that there was no unsupervised blocking, employing these methods implied choosing between scalability or automation, but not both. Recently however, we presented an *unsupervised* BSL [16]. In the context of this thesis, the solution addressed an impending *automation* issue.

The importance of scalability is further highlighted by an increased interest in parallel and distributed ER systems [19], with a good example of a MapReduce-based system being Dedoop [18]. Dedoop expects domain expertise or training examples, and cannot deduce blocking schemes on its own. We attempt to build our own MapReduce-based system for this thesis that integrates our automatic BSL. The ultimate goal is to ensure the system is both automated and scalable.

All works mentioned above (including our own) assumed the relational setting, and additionally, *structural homogeneity*. Elmagarmid et al. defines this as input tables restricted to having the *same* schemas [8]. Applying and extending these methods to structurally heterogeneous datasets is still relatively open, and addressed in this thesis as one *heterogeneity* contribution.

Because of the prevalence of RDF, the Semantic Web community has developed its own solutions to the ER problem. Research in this area was propelled in large part by the Silk system [13]. Ferraram et al. survey some recent ER

methods in the Semantic Web [10]. Two important systems that have gained recent attention are RDF-AI [23] and Knofuss [21].

From the lens of our three criteria, we note that no automatic MapReduce-based instance matching system currently exists in the Semantic Web. As for heterogeneity, while it is addressed *within* the community [22], linking instances of two *different* models is relatively unaddressed. In this thesis, we present techniques to link tables and RDF graphs in a *unified* framework. In current literature, data models are explicitly assumed to adhere either to RDF or relational. This discussion shows that, while ER research is continuing to converge, no system currently fulfills all three criteria in a combined framework.

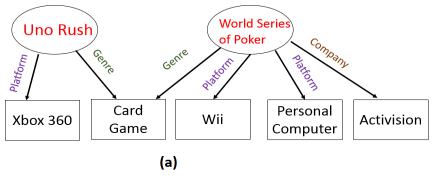
We also note that *DataSpace Support Platforms* (DSSPs) abstractly address some of the same three criteria as in this thesis [14]. To quote from an influential work on DSSPs, the framework assumes a '*large* number of *diverse*, interrelated sources' [11]. The authors also described how human attention should be judiciously used to continuously improve performance. Therefore, DSSPs embody scalability, heterogeneity and automation requirements that we address in this thesis. Traditionally, DSSPs were proposed in the context of *query answering* [11]. This thesis adapts similar principles to ER.

4 Research Questions

Based on the reviewed work, we identified three open research questions to motivate the thesis proposal and to respectively address heterogeneity, automation and scalability:

- In our context, a practical ER system must be designed to work with both RDF and relational models, given the growth of both LOD [2] and the Deep Web [12]. The first research question investigates whether existing ER techniques are even *well-defined* for input pairs where one input is RDF and the other, relational (*data model* heterogeneity). We also ask if current relational record linkage techniques can properly address *structural heterogeneity*.
- Assuming dataspace-like principles [14], a full ER pipeline must properly accommodate *uncertain* domain expertise. A research question is if we can anticipate the possible *forms* of domain expertise likely to be available, and incorporate the expertise in a judicious, performance-enchancing manner. A stronger research question is if we can eliminate supervision altogether.
- Given the dependencies among proposed methods, the last research question asks if a full ER pipeline can be implemented in a MapReduce-based prototype while still fulfilling the other two criteria.

The first two questions identify heterogeneity and automation issues. The third questions the possibility of scalably integrating these techniques in MapReduce. We choose MapReduce specifically because of its proven fault-tolerance advantages and the convenience of scaling and deploying MapReduce programs on the cloud [6].



Subject	Genre	Platform	Company
World Series of Poker	Card game	Wii; Personal Computer	Activision
Uno Rush	Card game	Xbox 360	*null*

(b)

Fig. 1. An example of the property table representation of an RDF fragment.The keyword *null* and delimiter *;* are reserved. Each field value in the property table has set semantics.

5 Hypotheses

To answer each of these questions, we present the following hypotheses:

- Abstractly, RDF-tabular heterogeneity exists because there are two data models involved, and *also* because the RDF model may not have accompanying schema information (as RDFS or OWL). We hypothesize that this heterogeneity is partially reconciled by *logically* representing RDF in a *property table*, which was originally a *physical* data structure for efficient triple store implementations [25]. We detail the rationale (and potential issues) in Section 6. Figure 1 shows an example.
- We can employ principles from our unsupervised BSL [16] to generate our *own* (noisy) training samples using inexpensive heuristics. We hypothesize that robust machine learning classifiers (e.g. SVMs) will be able to learn from these noisy samples in a self-supervised fashion. We also propose a *knowledge base* (KB) to flexibly incorporate available domain expertise. The hypothesis is that after N ER tasks, the KB will be sophisticated enough that the $N + 1^{th}$ task requires no user corrections at all, for reasonable N.
- We design a full ER prototype with *two* MapReduce phases and *one* serial module, with the aim of bounding time and space complexity of each compute node by a sub-quadratic function. Moreover, we would ideally want the *number* of compute nodes to scale linearly with input dataset size.

6 Approach

We enumerate some approaches for testing the outlined hypotheses:

- Using a *property table* has several advantages that allow us to use it for ER. First, the property schema is built *dynamically* and populated at run-time, which makes it appropriate for non-static LOD datasets. Secondly, property tables are physical data structures that are already used in the Jena framework for implementing triple stores [25]. This gives the approach a *systems-level* advantage, since we can link RDF data already present in Jena triple stores, without processing, moving or re-storing the data. Note that some interesting *inverse mapping* issues arise when representing RDF as property

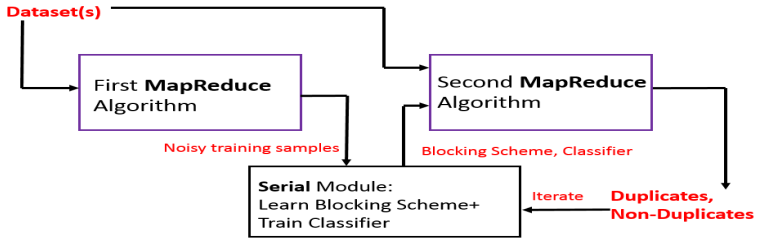


Fig. 2. An in-development prototype for unsupervised ER, with the KB excluded. We do not describe the full details of this schematic in this document; an implemented, evaluated version (with an integrated KB) will be a key deliverable of the proposed thesis

tabels. For example, R2RML explores mappings in the relational to RDF direction [5]. Property tables explore mappings in the inverse direction, with a view towards performing ER between two models. This is an important theoretical consequence that we will investigate during the course of the thesis. Another open issue is how to incorporate available schema information (as OWL/RDFS files) of the RDF datasets in the property table.

- We will adapt the noisy training set generator from our previous work [16], to bootstrap the system and make it self-supervised. We will implement the KB as an ontology, and try a variety of techniques, including rules, transfer learning and online algorithms to evaluate the best way of accommodating uncertain domain expertise.
- The schematic of the prototype is shown in Figure 2. Intuitively, a first MapReduce phase can be used to *generate* noisy training samples, while a second MapReduce phase is used to perform two-step ER, based on a blocking scheme and classifier trained in a *serial* module. If available, domain expertise can also be utilized in the serial module. In the first set of empirical evaluations, we will disallow domain expertise. After deriving initial insights, we will attempt integrating an adaptive KB into the unsupervised prototype.

7 Evaluation Plan

Both the ER and machine learning literature have well-defined criteria for evaluating success of our proposed methods. Henceforth, we assume RDF datasets are represented as property tables.

- The *blocking* phase of ER is traditionally evaluated separately using three metrics: Pairs Completeness (PC), Reduction Ratio (RR) and Pairs Quality (PQ). In our first set of evaluations, we measure the quality of our noisy training samples as well as unsupervised blocking. In essence, we are evaluating the impact of proposed *automation* and *heterogeneity* techniques on the blocking problem.

- Final results of ER are evaluated by plotting *precision* against *recall* of the obtained duplicates. We will follow this same methodology for the full ER system. The first round of full evaluations will be in a *serial* framework.
- The first[1] MapReduce-based prototype will be evaluated empirically through a Hadoop implementation. We will test individual components of the system, as well as the results of ER as a whole. Run-times will also be recorded and plotted to empirically evaluate the *scalability* of the system. Additionally, we will analyze and provide theoretical (time and space) bounds on the various phases. Ideally, we would want to prove near-linear scaling of the system.
- We will evaluate the KB through plotting *learning curves* showing precision-recall f-scores against *units* of domain expertise. The specific nature of these units will depend on the finalized KB approach. Because it is the most uncertain of the current thesis proposal, this evaluation is the last in our sequence.

We will carry out all these experiments on a range of at least three real-world, representative test suites. At least one of these has already been identified and published by us in a related work [24]. We have also identified good sources of government data.

8 Preliminary Results

We observed in Section 3 that, although unsupervised techniques have been developed for several ER phases, learning a blocking scheme remained supervised till quite recently. In a 2013 work, we devised an approach for unsupervised learning of blocking schemes on structurally homogeneous datasets [16]. To deal with lack of training data, we devised an inexpensive token-based algorithm to generate our own *noisy* training set. We showed that for a *few* retrieved examples, the noise levels are low. Figure 3 shows some selected results from this work. On all three benchmarks, precision was well over 90%; on *Cora*, it was almost perfect. In our blocking scheme learner [16], we used feature-selection based techniques to compensate for these noise levels. Final blocking results (not shown here) were competitive compared to a *supervised* set-covering baseline[1]. This shows that noisy training set generation is a viable procedure as long as noise levels are low, and the algorithmic design takes the noise into account.

We have since extended this work to account for structural heterogeneity and link discovery [17], and are also in the process of using it for a full self-supervised ER pipeline. Additionally, we have also developed some theoretical results on heterogeneous blocking in a recent report, and are continuing to update it [15].

9 Reflections

In this thesis, we propose solutions to the ER problem that address three key criteria of automation, heterogeneity and scalability in a unified framework. To

[1] The first version is unsupervised, and does not employ the KB (Figure 2)

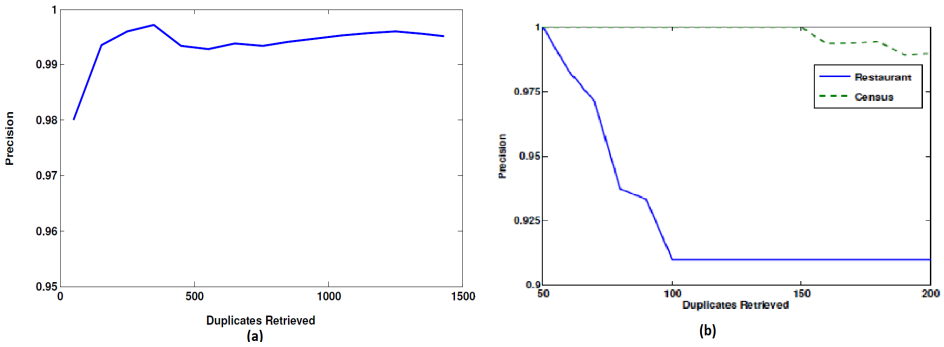


Fig. 3. Precision of duplicates automatically retrieved from three benchmark datasets ((a) shows *Cora* and (b) *Restaurant* and *Census*), using an inexpensive heuristic. Precision of non-duplicates (up to 5000 samples) was 100% for all benchmarks

address automation, we generate noisy training samples in an unsupervised procedure, and incorporate limited domain expertise by building and updating a flexible *knowledge base*. We extend methods to explicitly account for *structural* heterogeneity, and use the physical property table representation of RDF *logically* to address *data model* heterogeneity. Proposed methods will be implemented in a MapReduce-based prototype to address *scalability* and cloud deployment.

One potential area of concern is the *noise* in the generated training samples. The system will therefore have to be *robust*. Another threat could be scaling the property table for highly heterogeneous RDF datasets, and incorporating OWL schema information in the property schema. We continue to investigate these issues, both empirically and also in a theoretical framework.

Acknowledgments. I want to thank my advisor, Dr. Daniel P. Miranker, for his support. I am also grateful to the reviewers for their constructive feedback.

References

1. M. Bilenko, B. Kamath, and R. J. Mooney. Adaptive blocking: Learning to scale up record linkage. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 87–96. IEEE, 2006.
2. C. Bizer, T. Heath, and T. Berners-Lee. Linked data-the story so far. *International journal on semantic web and information systems*, 5(3):1–22, 2009.
3. P. Bouquet and A. Molinari. A global entity name system (ens) for data ecosystems. *Proceedings of the VLDB Endowment*, 6(11):1182–1183, 2013.
4. P. Christen. A survey of indexing techniques for scalable record linkage and deduplication. *Knowledge and Data Engineering, IEEE Transactions on*, 24(9):1537–1555, 2012.
5. S. Das, S. Sundara, and R. Cyganiak. {R2RML: RDB to RDF Mapping Language}. 2012.

6. J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.
7. A. Doan, A. Halevy, and Z. Ives. *Principles of data integration*. Access Online via Elsevier, 2012.
8. A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, 19(1):1–16, 2007.
9. J. Euzenat, P. Shvaiko, et al. *Ontology matching*, volume 18. Springer, 2007.
10. A. Ferraram, A. Nikolov, and F. Scharffe. Data linking for the semantic web. *Semantic Web: Ontology and Knowledge Base Enabled Tools, Services, and Applications*, page 169, 2013.
11. A. Halevy, M. Franklin, and D. Maier. Principles of dataspace systems. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–9. ACM, 2006.
12. B. He, M. Patel, Z. Zhang, and K. C.-C. Chang. Accessing the deep web. *Communications of the ACM*, 50(5):94–101, 2007.
13. R. Isele and C. Bizer. Learning expressive linkage rules using genetic programming. *Proceedings of the VLDB Endowment*, 5(11):1638–1649, 2012.
14. S. R. Jeffery, M. J. Franklin, and A. Y. Halevy. Pay-as-you-go user feedback for dataspace systems. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 847–860. ACM, 2008.
15. M. Kejriwal and D. P. Miranker. N-way heterogeneous blocking. In *TR-14-06*, 2013.
16. M. Kejriwal and D. P. Miranker. An unsupervised algorithm for learning blocking schemes. In *Data Mining, 2013. ICDM'13. Thirteenth International Conference on*. IEEE, 2013.
17. M. Kejriwal and D. P. Miranker. A two-step blocking scheme learner for scalable link discovery. In *Under review at ISWC OM Workshop*, 2014.
18. L. Kolb, A. Thor, and E. Rahm. Dedoop: efficient deduplication with hadoop. *Proceedings of the VLDB Endowment*, 5(12):1878–1881, 2012.
19. A.-A. Mamun, T. Mi, R. Aseltine, and S. Rajasekaran. Efficient sequential and parallel algorithms for record linkage. *Journal of the American Medical Informatics Association*, pages amiajnl–2013, 2013.
20. M. Michelson and C. A. Knoblock. Learning blocking schemes for record linkage. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 440. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.
21. A. Nikolov, V. Uren, E. Motta, and A. De Roeck. Handling instance coreferencing in the knofuss architecture. 2008.
22. A. Nikolov, V. Uren, E. Motta, and A. De Roeck. Overcoming schema heterogeneity between linked semantic repositories to improve coreference resolution. In *The Semantic Web*, pages 332–346. Springer, 2009.
23. F. Scharffe, Y. Liu, and C. Zhou. Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. In *Proc. IJCAI 2009 workshop on Identity, reference, and knowledge representation (IR-KR), Pasadena (CA US)*, 2009.
24. A. Tian, M. Kejriwal, and D. P. Miranker. Schema matching over relations, attributes, and data values. In *Proceedings of the 26th International Conference on Scientific and Statistical Database Management*, page 28. ACM, 2014.
25. K. Wilkinson, C. Sayers, H. A. Kuno, D. Reynolds, et al. Efficient rdf storage and retrieval in jena2. In *SWDB*, volume 3, pages 131–150, 2003.