2015 IEEE International Conference on Big Data (Big Data)

Holistic Entity Matching Across Knowledge Graphs

Maria Pershina[+]* Mohamed Yakout[^] Kaushik Chakrabarti[^]
[+]New York University
pershina@cs.nyu.edu
[^]Microsoft Research
{myakout,kaushik}@microsoft.com

***Abstract*—Entity matching is the problem of determining if two entities in a data set refer to the same real-world object. In the last decade a growing number of large-scale knowledge bases have been created online. Tools for automatically aligning these sources would make it possible to unify them in a structured knowledge and to answer complex queries. Here we present Holistic Entity Matching (HolisticEM), an algorithm based on Personalized Page Rank for aligning instances in large knowledge bases. It consists of two steps. First, a graph of potential matching pairs is constructed; second, local and global information from the relationship graph is propagated via Personalized Page Rank. We demonstrate that HolisticEM performs competitively and can efficiently handle databases with 110M and 203M entities accurately resolving 1.6M of matching entity pairs.**

I. Introduction

A common prerequisite for knowledge discovery is accurately combining data from multiple, heterogeneous sources into a unified, mineable knowledge graph. An important step in creating such a graph is entity matching. Entity matching is the problem of determining if two entities in a data set refer to the same real-world object. It is a complex and ubiquitous problem, that appears in numerous application domains including information extraction, data integration, language processing.

As an example consider two toy knowledge graphs in Figure 1ab. Nodes in these graphs are actor, movie, character, performance entities, and their attributes. Edges between nodes correspond to Resource Description Framework (RDF) triples (s, p, o) and are annotated with predicates p. For example, triple (p1, played_by, a1) represents an edge between performance p1 and actor a1, meaning that performance p1 was played by an actor a1.

A growing body of work has shown that incorporating global information can improve the performance. Some examples are simultaneous coreference in [18], [19], jointly modeled record and field coreference in [6], dirichlet process for modeling interactions between dataset entities in [2], [12], distribution of wrong entries in input datasets for data fusion in [7], probabilistic ontology alignment in [20]. Main limitations of the above techniques are requirements for prior domain knowledge for modeling, data for training, and/or probabilistic inference, which makes these methods computationally infeasible for large data sets.

Missing or incomplete information in the database is another challenge in the entity matching process, e.g. actors a1 and a3 in Figure 1 have same name "Douglas" but correspond to different people - actors Sam Douglas and Douglas Price. Only dissimilarity of neighboring pairs can help to properly resolve this pair, e.g. different names for characters (c1,c3) should impact the score for (a1, a3). The intuition behind our algorithm is that matching nodes should have similar nodes in their neighborhood, thus our approach ranks pair (a2,a4) higher than pair (a1,a3) in Figure 1.

Greedy iterative approaches [4], [8], [11], [14], [21] process nodes sequentially, e.g. by using priority queue: the highest scored node is resolved as match, triggering updates for other nodes, and so on. The drawback of this process is the propagation of erroneous decisions, accepted earlier. For example, different actors (a1,a3) would be mistakenly resolved as match by a greedy approach, since they have exactly same attributes. This decision would later boost the similarity score between movies (m1,m2) that are clearly different.

In the era of big data, sources to merge may comprise millions of nodes of tens of different types and will require scalable techniques to resolve matches. There are many approaches, such as blocking, clustering, bootstrapping [3], [5], [15], [16], [21], constrained deduplication [1], duplicate detection [13], proposed to avoid the quadratic number of comparisons between all pairs of entities to make it scalable. Many entity matching techniques strive for scalability and implicitly use the graph of potential matching candidate pairs to propagate similarity scores. In this paper we present a novel scenario to construct such a graph.

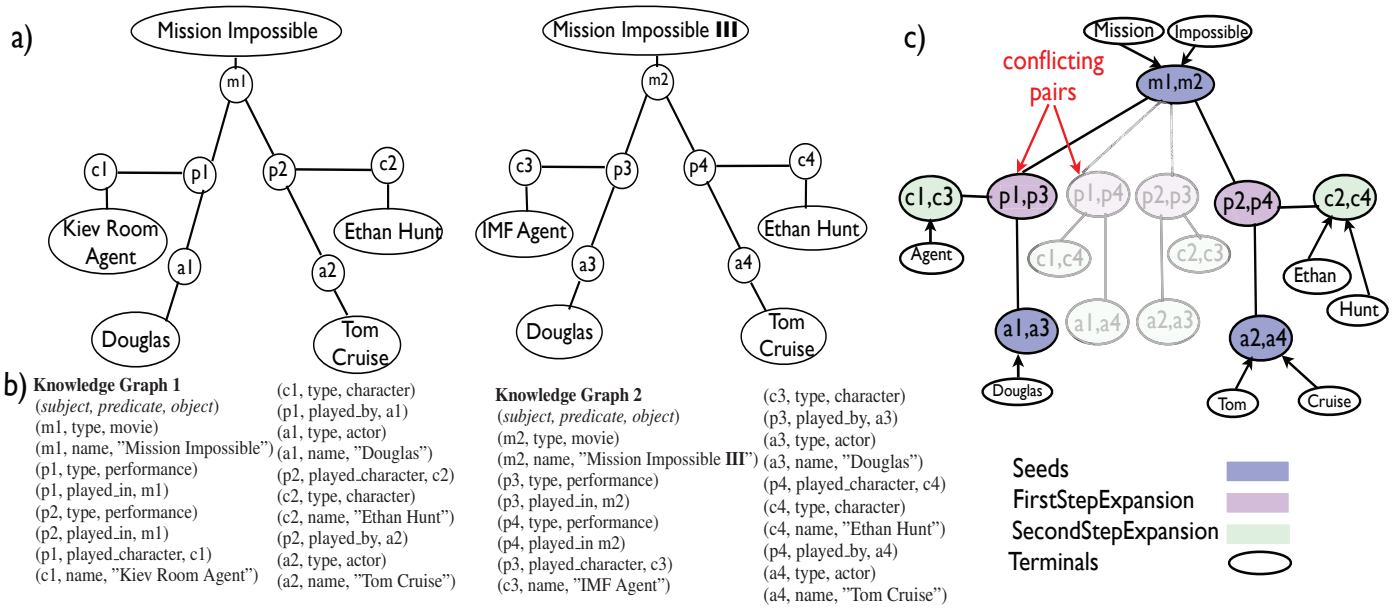
There are primary and relationship entities in the knowledge graphs. Primary entities can have non-reference attributes, such as $name$, etc (e.g. actor a2). Relationship entities serve to connect primary entities and to describe this connection, e.g. performance p1 in Figure 1ab shows that actor a1 played character c1 in movie m1. Relationship entities do not have any non-reference attributes making it very difficult to match them. Our approach achieves a high F-score of 98% when resolving relationship entity matches.

Main properties of our entity matching framework can be summarized as follows:

- Generic: domain independent, robust to incomplete data, accurate for primary and relationship entities;
- Scalable: both graph construction and score propagation scheme are scalable for large datasets;
- Efficient: does not require training, probabilistic inference, intermediate local models; does not propagate errors

*Work was done while visiting Microsoft Research.

978-1-4799-9926-2/15/$31.00 ©2015 IEEE



b) **Knowledge Graph 1**
(*subject, predicate, object*)
(m1, type, movie)
(m1, name, "Mission Impossible")
(p1, type, performance)
(p1, played_in, m1)
(p2, type, performance)
(p2, played_in, m1)
(p1, played_character, c1)
(c1, name, "Kiev Room Agent")
(c1, type, character)
(p1, played_by, a1)
(a1, type, actor)
(a1, name, "Douglas")
(p2, played_character, c2)
(c2, type, character)
(c2, name, "Ethan Hunt")
(p2, played_by, a2)
(a2, type, actor)
(a2, name, "Tom Cruise")

Knowledge Graph 2
(*subject, predicate, object*)
(m2, type, movie)
(m2, name, "Mission Impossible III")
(p3, type, performance)
(p3, played_in, m2)
(p4, type, performance)
(p4, played_in m2)
(p3, played_character, c3)
(c3, name, "IMF Agent")
(c3, type, character)
(p3, played_by, a3)
(a3, type, actor)
(a3, name, "Douglas")
(p4, played_character, c4)
(c4, type, character)
(c4, name, "Ethan Hunt")
(p4, played_by, a4)
(a4, type, actor)
(a4, name, "Tom Cruise")

Fig. 1: Toy knowledge databases represented as a) knowledge graphs; b) *(s,p,o)* triples; c) graph of entity pairs.

by doing *simultaneous* resolution for all nodes;

- Experiments on Microsoft Movie Knowledge Graphs validates the effectiveness and scalability of our approach by accurately resolving 1.6M matching pairs and outperforming baseline and state-of-art models.

II. Pairs Graph Construction

Constructing the Pairs Graph with all possible pairs of entities is unnecessary and often not feasible when we integrate very large knowledge graphs. However, we want the Pairs Graph to have at least all the matching entities. A pair of entities is a potential match if: (1) their attribute values overlap (or their corresponding bag-of-word encodings overlap); and/or (2) they are connected to entities in their corresponding graph, that are likely to match.

First, we construct potentially matching Seed Pairs based on the direct attributes similarity. Second, we expand Seed Pairs, using their corresponding connected entities, and add the necessary new entity pairs and edges to the graph.

A. Seed Pairs Generation

In the generation of Seed Pairs we rely only on the attribute values of the entities, encoded into a bag of words. We compute IDF scores for words with respect to the source graph and calculate cosine similarity between entities.

To compute it efficiently with MapReduce we organize encoded entities e for each input graph into schema $\langle Word, e, idf \rangle$. Then we combine it on a $Word$ to produce a table with scores for individual words w that are common for e_1 and e_2:

$$\langle\ e_1, e_2,\ score(w) = \frac{1}{||e_1|| \cdot ||e_2||} idf_1(w) \times idf_2(w) \rangle$$

and finally reduce on e_1, e_2 to obtain their initial similarity:

$$\langle e_1, e_2, sim(\langle e_1, e_2 \rangle) = \frac{1}{||e_1|| \cdot ||e_2||} \sum_{w \in e_1 \cap e_2} idf_1(w) \times idf_2(w) \rangle \quad (1)$$

In practice, we generate Seed Pairs separately for each entity type and do additional optimization and pruning, e.g. we discard words with very low IDF score (stop words), and keep only top $k = 100$ potential matches per entity.

B. Expanding the Pairs Graph

We want to construct the pairs graph, which captures the interaction between pairs in terms of similarity influence. An edge between entity pairs describes the dependency between their corresponding similarities. We generate additional pairs and edges to add connectivity to the graph and to include relationship entities, that do not have any atomic attribute values to be considered at the seed generation step.

We do two-step expansion for seed pairs $(a_s, b_s) \in$ Seed Pairs, generated for graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$.

First Step. For triples $(a_s, p, a) \in E_1$ and $(b_s, p, b) \in E_2$ we add an entity pair node (a, b), if both a and b are entities of the same type. Then we add an undirected edge connecting it to (a_s, b_s). Similar step is performed for triples $(a, p, a_s) \in E_1$ and $(b, p, b_s) \in E_2$. For example, we add a new pair (p1,p3) and an edge connecting it to the seed (m1,m2) (Figure 1c).

Second Step. At this step we expand relationship entities (a, b) that were added at the first step. Thus, we expand a pair of performances (p1,p3) by adding a pair of characters (c1,c3) and a corresponding edge between these two pairs. This two-step process is illustrated in Figure 1c. There can be slightly different definitions of relationship entities. We use entities that do not have *name* attribute.

There are many conflicting pairs produced during expansion process, e.g. a pair of movies (m1,m2) generates four performance pairs (p1,p3), (p1,p4), (p2,p3), (p2,p4). We assume, that there are no duplicates within original knowledge graphs and thus pairs (p1,p3) and (p1,p4) cannot coexist together with respect to their parent (m1,m2). To resolve conflicts between nodes we use local information, such as shared terminal attributes, and *stable marriage* heuristics [10] to keep only relevant pairs. For example, performance node p1 can "marry" either p3 or p4, same is true about node p2. Node p1 would "prefer" p3 since their immediate neighbor pairs (c1,c3) and (a1,a3) have some shared terminals. Similarly, p2 would "prefer" p4 over p3. As a result, the stable marriage algorithm maps p1 to p3 and p2 to p4. Thus nodes (p1,p4) and (p2,p3) and their further extensions, (c1,c4), (a1,a4), (c2,c3), (a2,a3), are removed from the graph (shaded nodes in Figure 1c).

$Terminals$. Lastly, we add shared terminals to all nodes. Thus, string $Douglas$ is attached to the pair (a1,a3), strings Tom and $Cruise$ are attached to the pair (a2,a4), etc.

Individual seed expansion step is independent from other seeds expansion, so it is parallelizable and is done efficiently with MapReduce. The graph construction process is summarized in Algorithm 1.

Algorithm 1 : Pairs Graph Construction

```
Input: graphs G1(V1, E1(s,p,o)), G2(V2, E2(s,p,o))
Phase 1: generate Seeds = {(s1, s2) | s1 ∈ V1, s2 ∈ V2}
Phase 2: expand Seeds

table Edges(pair(s1, s2), pair(o1, o2));
// One step expansion for all seeds.
for (as, bs) ∈ Seeds do
    Edges=Edges ∪ SINGLESTEP(as, bs);
end for
table newNodes = select pairs (s1, s2), (o1, o2)
                 from Edges;
// Second step expansion for relationship entities.
for {(a, b) ∈ newNodes & isRelationshipEnt(a, b)} do
    Edges=Edges ∪ SINGLESTEP(a, b);
end for

makeBidirectionalEdges(Edges);
table AllNodes(s1, s2) = select (s1, s2), (o1, o2)
                         from Edges, Seeds;
// Find shared terminal attributes of the same type p1 = p2.
table TerminalEdges(terminal o, pair(s1, s2)) =
  AllNodes combine E1 on AllNodes.s1 = E1.s1
           combine E2 on AllNodes.s2 = E2.s2
           where E1.p1 = E2.p2
           and E1.o1 = E2.o2 = o is terminal;
return Edges ∪ TerminalEdges;

function SINGLESTEP(pair (a, b))
    // case 1: find edges to object pairs
    table Expand_O(pair(a, b), pair(o1, o2)) =
        E1 combine E2 on p1 = p2, s1 = a, s2 = b;
    // case 2: find edges from subject pairs
    table Expand_S(pair(s1, s2), pair(a, b)) =
        E1 combine E2 on p1 = p2, o1 = a, o2 = b;
    // Remove conflicting nodes via StableMarriage algo
    NewEdges = StableMarriage(Expand_O ∪ Expand_S);
    return NewEdges;
end function
```

III. Random Surfer Model

The graph of pairs captures the influence of each pair of entities on the neighborhood entities (Figure 1). We need a holistic approach to quantify the influence of terminal values similarity towards entity pairs similarity and the influence of one entity pair to another.

A. Personalized PageRank

Consider a directed weighted graph $G(V, E)$. PageRank is the stationary distribution of a random walk on G, where at each step with probability ϵ (teleport probability) we jump to a randomly selected node on a graph, and with probability $1-\epsilon$ we follow a random outgoing edge for the current node. Personalized PageRank (PPR) is the same as PageRank but all teleports are made to the same source node, for which we personalize the PageRank. Thus, for every source node v and a landing node u there is an associated PPR weight denoted as $PPR(v \to u)$.

B. Holistic Similarity In a Knowledge Graph

The contribution of each word to the similarity of a pair of entities should consider the words popularity. Popular words across entities will less likely influence the matching decision. Therefore each word will have a weight. The well established measure for this purpose is the IDF (or Inverse Document Frequency) which is well known in the Information Retrieval area. The lower the IDF score of a word is the more entities this word is shared between. Hence, the less distinguishing such a word is to its entities.

The contribution of a word w to the similarity of a pair p is conversely proportion to the degree of the node w in the graph (i.e. the number of node pairs connected to node w). This is similar to the popularity notion, which the IDF weight is trying to capture.

Another way to explain words contributions to the pairs similarity is by considering a random surfer, walking in the graph. Let us consider a random surfer that continuously starts its trip at the word node w. Then the landing probability of the surfer at node $\langle e_1, e_2 \rangle$ is essentially the amount of contribution of word w to the similarity of pair $\langle e_1, e_2 \rangle$. Summing these contributions over all words w we obtain

$$sim(\langle e_1, e_2 \rangle) = \sum_{\forall w} PPR(w \to \langle e_1, e_2 \rangle) \qquad (2)$$

The above reasoning about the contribution of the words to entities similarity using a random surfer can be extended further to the contribution of entity pairs towards entity pairs.

C. Optimization

Equation (2) requires computing PPR weights for all primitive words w in the graph to calculate $sim(\langle e_1, e_2\rangle)$. Thus, to compute the contribution of a pair of actors (a2,a4) from Figure 1c) to the similarity of a pair of movies (m1,m2) we would have to start a random surfer at each shared primitive value for (a2,a4): names *Tom* and *Cruise*. One can optimize this by computing summary of words at the pair level as pair's initial similarity, and then propagate it using PPR weights. We use two strategies to compute this initial similarity.

Simple Initial Similarity. Pairs Graph construction filters out irrelevant pairs, entities, and their attributes. It induces two subgraphs - one in each source. These are the subgraphs that were used to build Pairs Graph. We can compute initial similarity scores $iSim(\langle e_1, e_2\rangle)$ for every pair node $\langle e_1, e_2\rangle$ with respect to these two subgraphs in a similar fashion as in Section II-A, Equation (1): first, compute $idf(w)$ for words w with respect to the source subgraphs, then calculate cosine similarity between entities in each pair to produce $iSim(\langle e_1, e_2\rangle)$.

GraphBased Initial Similarity. Let us denote as V_w all pairs of entities in the graph $G(V, E)$ that share primitive value w. These pairs are immediate neighbors of w, so $|V_w| = degree(w)$. Random surfer, started at node w, is either teleported with probability ϵ or makes one step in a random direction with probability $1-\epsilon$. There are $|degree(w)|$ possible directions, so after one iteration random surfer lands at any of the nodes $v \in V_w$ with probability $P(v) = \frac{1-\epsilon}{degree(w)}$, and then the process resumes. Thus

$$PPR(w \to \langle e_1, e_2\rangle) = \sum_{v \in V_w} P(v) \cdot PPR(v \to \langle e_1, e_2\rangle) \sim \sum_{v \in V_w} \frac{1}{deg(w)} \cdot PPR(v \to \langle e_1, e_2\rangle)$$

Then similarity of a pair $\langle e_1, e_2\rangle$ from Equation (2) can be rewritten as following

$$sim(\langle e_1, e_2\rangle) = \sum_w PPR(w \to \langle e_1, e_2\rangle) \sim \sum_w \frac{1}{deg(w)} \sum_{v \in V_w} PPR(v \to \langle e_1, e_2\rangle)$$

Every vertex $v = \langle e_1', e_2'\rangle$ appears in this sum as many times as many primitive values w pair $\langle e_1', e_2'\rangle$ has. Let us denote this set of shared primitive values for vertex v as W_v. Changing the order of summation and combining terms for the same vertices we get

$$sim(\langle e_1, e_2\rangle) \sim \sum_v PPR(v \to \langle e_1, e_2\rangle) \sum_{w \in W_v} \frac{1}{deg(w)} = \sum_v PPR(v \to \langle e_1, e_2\rangle) \cdot iSim(v), \quad (3)$$

$$\text{where } iSim(v) = \sum_{w \in W_v} \frac{1}{deg(w)}$$

denotes initial similarity of node v and is equal to the sum of degree reciprocals of primitive values for node v.

This computation justifies the idf logic, described in Section III-B. Namely, the contribution of a word $w \in W_v$ into initial similarity of a pair v is conversely proportion to the degree of the node w in the graph (i.e. the number of reference pairs sharing word w).

The optimization step (3) subsumes all shared primitive values $w \in W_v$ of every pair v into initial similarity $iSim(v)$.

These strategies for summarizing primitive values allow us to remove all terminal nodes from the pairs graph and thus drastically reduce its size. Moreover, this step significantly simplifies further computation by reducing starting points for random surfer to only non-terminal (reference) nodes:

$$sim(\langle e_1, e_2\rangle) = \sum_{\substack{v \in non\text{-}terminal \\ nodes}} PPR(v \to \langle e_1, e_2\rangle) \cdot iSim(v) \quad (4)$$

IV. Pipeline

Our pipeline for Holistic Entity Matching is in Figure 2. Its input consists of two knowledge graphs. The goal is to identify duplicate entries in these graphs. There are following steps in this pipeline: (1) generate Seeds; (2) construct Pairs Graph; (3) compute initial similarity for pairs in the Pairs Graph; (4) propagate initial scores via PPR; (5) resolve final scores.

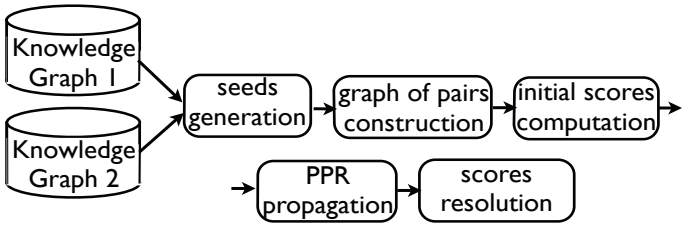


Fig. 2: Pipeline for HolisticEM framework.

V. Experiments

Data. For our experiments we use two datasets: Freebase[1] and IMDB dataset from an internal data warehouse[2]. Given two sources with more than 110M and 203M entities correspondingly (Table I), we focus on several entity types of interest to avoid space and time limitations. Namely, we pick actors and movies, as they are closely related and thus may benefit from each other. During graph construction step these entities will introduce new nodes, such as performance pairs, allowing us to validate our technique for relationship entities.

Graph construction and score propagation. We generated 5M seeds, 3.2M actor pairs and 1.8M movie pairs, and built graph of pairs as described in Section II. We adopt the Monte Carlo approach [9] for computing Personalized PageRank. It performs a number of independent random walks for every source node and takes an empirical distribution

[1] www.freebase.com

[2] Internal Microsoft IMDB dataset is obtained from imdb pages.

Dataset	Freebase	IMDB
Total Entities	110.6M	203.9M
Actors	425.9K	2.7M
Movies	240.8K	2.5M
Performances	1.2M	5.8M

TABLE I: Freebase and IMDB datasets statistics.

of ending nodes to obtain PPR weights with respect to the source. We initialized 4,000 random walks for every source node, performed 5 steps of PPR at each node with teleport probability $\epsilon = 0.2$, and computed final scores according to (4).

Models and Evaluation. Seed scores from Section II-A, and initial scores from Section III-B, are two our baselines; we compare its precision and recall with HolisticEM in Table II. In addition to the ground truth matches, available from internal data warehouse, that covers about 95% of all matches between Freebase and IMDB datasets, we perform additional manual evaluation of uniformly sampled 1000 unmatched entities for each type. We use user-defined thresholds for every entity type to resolve HolisticEM scores.

Models	Movies P/R/F	Actors P/R/F	Performances P/R/F
Seeds	68.2 / 82.4 / 74.6	66.4 / 79.1 / 72.2	N/A
Simple	92.0 / 76.1 / 83.3	83.2 / 78.2 / 80.6	N/A
GraphBased	88.2 / 86.4 / 87.3	86.9 / 83.2 / 85.0	N/A
Holistic+S	93.1 / 92.5 / 92.8	95.8 / 93.5 / 94.7	93.1 / 95.9 / 94.4
Holistic+GB	99.3 / 96.9 / **98.1**	98.9 / 97.9 / **98.4**	98.6 / 97.4 / **98.0**

TABLE II: Precision, Recall and F-score for (1) Seed Pairs; (2) pairs in Pairs Graph with Simple and GraphBased initial scores; (3) pairs in Pairs Graph with PPR-propagated Simple (HolisticEM+S) and GraphBased (HolisticEM+GB) scores.

Results. Holistic Entity Matching improves F-score over Seeds and both initial scores - Simple and GraphBased. It is interesting that pairs generated in Pairs Graph and scored with Simple or GraphBased routines are already a good baselines achieving on movies an F-score of 83% and 87% correspondingly. Propagating initial scores with PPR further improves the performance achieving a very competitive results with F-score of 98.1% on movies and F-score of 98.4% on actors. The GraphBased initial scores perform better than Simple initial scores proving that optimization (3) properly captures contribution of terminal values with respect to the graph structure.

This result compares favorably with the most recent state-of-the-art greedy approach SiGMa [14] that achieves an F-score of 97% on movies when merging smaller datasets, also derived from IMDB and Freebase, with 3.1M and 474K entities correspondingly and with ground truth of 255K movies. Using similar technical specifications our HolisticEM requires 4.5 hours of running time while SiGMa runs for 1.5 hours. The gain in running time of SiGMa over HolisticEM is related to the sizes of the retrieved ground truth and sizes of original databases: 255K of ground truth from 3.1M and 0.45M entities vs 1.6M of ground truth from 203M and 110M entities. Thus 3x times longer running time of HolisticEM enables the extraction of 6 times bigger ground truth from 60 times bigger datasets. This justifies the efficiency of HolisticEM that is able to handle probably the largest knowledge bases to date.

In addition to primary entities HolisticEM efficiently resolves relationship entities matches achieving high F-score of 98.0% on performances (Table II).

VI. Conclusion and Future Work

We propose a novel scalable framework for collective entity matching across knowledge graphs. We describe a new way of constructing graph of potential matching pairs and propose a new scheme to propagate similarity between pairs in this graph. Building the subgraph from seeds, adding necessary connections and controlling its expansion can be a very efficient graph sampling technique for dense graphs, where direct computations are infeasible. By propagating scores via Personalized Page Rank we significantly simplified the entity matching routine. Our PPR-based framework does not require any prior domain knowledge, training, probabilistic inference; it scales to large datasets and has a competitive performance on both primary and relationship entities. This approach can be implemented on MapReduce to efficiently handle industrial size datasets.

Our future work will be focused on different scenarios for constructing pairs graph and on a more sophisticated score propagation schemes. We plan to explore different strategies for filtering out noise when aggregating contributions from different nodes in order to further improve the score propagation.

References

[1] A. Arasu, C. Re, and D. Suciu. Large-scale deduplication with constraints using dedupalog. *ICDE*, pages 952–963, 2009.

[2] I. Bhattacharya and L. Getoor. A latent dirichlet model for unsupervised entity resolution. *SDM*, pages 47–58, 2006.

[3] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *TKDD*, 1, 2007.

[4] C. Bohm, G. de Marco, F. Naumann, and G. Weikum. Linda: Distributed web-of-data-scale entity matching. *CIKM*, pages 2104–2108, 2012.

[5] W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. *KDD*, pages 475–480, 2002.

[6] A. Culotta and A. McCallum. Joint deduplication of multiple record types in relational data. *CIKM*, pages 31–48, 2005.

[7] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, K. Murphy, S. Sun, and W. Zhang. From data fusion to knowledge fusion. *VLDB*, pages 881–892, 2014.

[8] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. *SIGMOD*, pages 85–96, 2005.

[9] D. Fogaras and B. Racz. Towards scaling fully personalized pagerank. *WAW*, pages 105–117, 2004.

[10] D. Gale and L. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[11] Z. Guo and D. Barbosa. Robust entity linking via random walks. *CIKM*, pages 499–508, 2014.

[12] R. Hall, C. Sutton, and A. McCallum. Unsupervised deduplication using cross-field dependencies. *KDD*, pages 310–317, 2008.

[13] M. Herschel and F. Naumann. Scaling up duplicate detection in graph data. *CIKM*, pages 1325–1326, 2008.

[14] S. Lacoste-Julien, K. Palla, A. Davies, G. Kasneci, T. Graepel and Z. Ghahramani. SiGMa: Simple Greedy Matching for Aligning Large Knowledge Bases. *KDD*, pages 572–580, 2013.

[15] A. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. *KDD*, pages 169–178, 2000.

[16] V. Rastogi, N. Dalvi, and M. Garofalakis. Large-scale collective entity matching. *VLDB*, pages 208–218, 2011.
[17] V. G. S. Chaudhuri and R. Motwani. Robust identification of fuzzy duplicates. *ICDE*, pages 865 – 876, 2005.
[18] P. Singla and P. Domingos. Multi-relational record linkage. *KDD*, pages 31–48, 2004.
[19] P. Singla and P. Domingos. Entity resolution with markov logic. *ICDM*, pages 572–582, 2006.
[20] F. Suchanek, S. Abiteboul and P. Senellart. PARIS: Probabilistic Alignment of Relations, Instances, and Schema *VLDB*, pages 157–168, 2011.
[21] S. Whang, D. Menestrina, G. Koutrika, M. Theobald, and H. Garcia-Molina. Entity resolution with iterative blocking. *SIGMOD*, pages 219–232, 2009.