SiGMa: Simple Greedy Matching for Aligning Large Knowledge Bases

Simon Lacoste-Julien
INRIA - Sierra project-team
Paris, France

Konstantina Palla
University of Cambridge
Cambridge, UK

Alex Davies
University of Cambridge
Cambridge, UK

Gjergji Kasneci
Microsoft Research
Cambridge, UK

Thore Graepel
Microsoft Research
Cambridge, UK

Zoubin Ghahramani
University of Cambridge
Cambridge, UK

ABSTRACT

The Internet has enabled the creation of a growing number of large-scale knowledge bases in a variety of domains containing complementary information. Tools for automatically aligning these knowledge bases would make it possible to unify many sources of structured knowledge and answer complex queries. However, the efficient alignment of *large-scale* knowledge bases still poses a considerable challenge. Here, we present Simple Greedy Matching (SiGMa), a simple algorithm for aligning knowledge bases with millions of entities and facts. SiGMa is an iterative propagation algorithm that leverages both the structural information from the relationship graph and flexible similarity measures between entity properties in a greedy local search, which makes it scalable. Despite its greedy nature, our experiments indicate that SiGMa can efficiently match some of the world's largest knowledge bases with high accuracy. We provide additional experiments on benchmark datasets which demonstrate that SiGMa can outperform state-of-the-art approaches both in accuracy and efficiency.

Categories and Subject Descriptors

D.2.12 [**Software Engineering**]: Interoperability—*Data mapping*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Information networks*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Semantic networks*

Keywords

knowledge base; alignment; large-scale; entity; relationship; greedy algorithm

1. INTRODUCTION

In the last decade, a growing number of large-scale knowledge bases have been created online. Examples of domains include music, movies, publications, and biological data[1]. As these knowledge bases sometimes contain both overlapping and complementary information, there has been growing interest in attempting to merge them by *aligning* their common elements. This alignment could have important uses for information retrieval and question answering. For example, one could be interested in finding a scientist with expertise on certain related protein functions – information which could be obtained by aligning a biological database with a publication one. Unfortunately, this task is challenging to automate as different knowledge bases generally use different terms to represent their entities, and the space of possible matchings grows exponentially with the number of entities.

A significant amount of research has been done in this area – particularly under the umbrella term of *ontology matching* [7, 16, 12]. An ontology is a formal collection of world knowledge and can take different structured representations. In this paper, we will use the term *knowledge base* to emphasize that we assume very little structure about the ontology (to be specified in Section 2). Despite the large body of literature in this area, most of the work on ontology matching has been demonstrated only on fairly small datasets of the order of a few hundred entities. In particular, Shvaiko and Euzenat [26] identified *large-scale evaluation* as one of the ten challenges for the field of ontology matching.

In this paper, we consider the problem of aligning the *instances* in *large* knowledge bases, of the order of millions of entities and facts, where *aligning* means automatically identifying corresponding entities and interlinking them. Our starting point was the challenging task of aligning the movie database IMDb to the Wikipedia-based YAGO [28]. This provides another step towards the Semantic Web vision of interlinking different sources of knowledge as exemplified by the Linking Open Data Initiative[2] [4]. Initial attempts to match IMDb entities to YAGO entities by naively exploiting string and neighborhood information failed; we thus designed SiGMa (Simple Greedy Matching), a scalable greedy iterative algorithm that is able to exploit previous matching decisions as well as the relationship graph information between entities.

The design decisions behind SiGMa were both to be able to take advantage of the combinatorial structure of the match-

[1]Such as MusicBrainz, IMDb, DBLP, and UnitProt.
[2]http://linkeddata.org/

KDD'13, August 11–14, 2013, Chicago, Illinois, USA.

ACM 978-1-4503-2174-7/13/08 ...$15.00.

ing problem (in contrast to traditional scalable database record linkage approaches that make more independent decisions), as well as to focus on a simple approach that could be scalable. SiGMa works in two stages: it first starts with a small seed matching assumed to be of good quality. Then the algorithm incrementally augments the matching by using *both* structural information and properties of entities such as their string representation to define a modular score function. Some key aspects of the algorithm are that (1) it uses the current matching to obtain structural information, thereby harnessing information from previous decisions; (2) it proposes candidate matches in a *local* manner, from the structural information; and (3) it makes greedy decisions, enabling a scalable implementation. A surprising result is that we obtained accurate large-scale matchings in our experiments despite the greediness of the algorithm.

Contributions. The contributions of the present work are the following:

1. We present SiGMa, a knowledge base alignment algorithm which can handle millions of entities. The algorithm is easily extensible with tailored scoring functions to incorporate domain knowledge and has a simple implementation.[3] It also provides a natural tradeoff between precision and recall, as well as between computation and recall.
2. In the context of testing the algorithm, we constructed two large-scale partially labeled knowledge base alignment datasets with hundreds of thousands of ground truth mappings. We expect these to be a useful resource for the research community for developing and evaluating new knowledge base alignment algorithms.
3. We provide a detailed experimental comparison illustrating how SiGMa improves over the state-of-the-art. SiGMa is able to align knowledge bases with millions of entities with over 98% precision and 90% F-measure in less than two hours (a 50x speed-up over [27]). On standard benchmark datasets, SiGMa obtains solutions with higher F-measure than the best previously published results.

The remainder of the paper is organized as follows. Section 2 presents the knowledge base alignment problem with a real-world example as motivation for our assumptions. We describe the algorithm SiGMa in Section 3. We evaluate it on benchmark and on real-world datasets in Section 4, and situate it in the context of related work in Section 5.

2. ALIGNING LARGE-SCALE KNOWLEDGE BASES

2.1 Motivating example: YAGO and IMDb

Consider merging the information in the following two knowledge bases:

1. YAGO [28], a large semantic knowledge base derived from English Wikipedia, WordNet, and GeoNames.
2. IMDb, a large popular online database that stores information about movies.

The information in YAGO is available as a long list of triples (called *facts*) that we formalize as $\langle e, r, e' \rangle$, which means that the directed relationship r holds from entity e to entity e', such as $\langle$John_Travolta, ActedIn, Grease$\rangle$. The information from IMDb was originally available as several files that we merged into a similar list of triples. We call these two databases *knowledge bases* to emphasize that we are not assuming a richer representation like RDFS [29] (that distinguishes between classes and instances). In the language of ontology matching, our setup is the less studied *instance matching* problem, discussed by Castano et al. [6]. Here, the goal is to match concrete instantiations of concepts like specific actors and movies rather than the general actor or movie class. YAGO comes with an RDFS representation, but IMDb does not; therefore, we focus on methods that do not assume or require a class structure or rich hierarchy in order to find a *one-to-one matching* of instances between YAGO and IMDb. However, we assume that the relations between the two knowledge bases can be manually aligned. This is straightforward for these types of knowledge bases (e.g. column 1 and 3 of Table 2).

Relationships vs. properties. In this work, we decided to exploit the powerful assumption that the alignment is *injective* (1–1) – as we will see in our experiments, this covers most of the cases for the YAGO-IMDb setup, as well as other datasets. Given our 1–1 assumption, we need to distinguish between two types of objects present in the list of triples: *entities* vs. *literals*. By our definition, the *entities* are the only objects that we try to align – they are objects like specific actors or movies that have a clear identity. The *literals*, on the other hand, correspond to a value related to an entity through a special kind of relationship that we call *property*. The defining characteristic of literals is that it would not make sense to try to align them between the two knowledge bases in a 1–1 fashion – examples are numerical values, dates, strings, etc. We use the literals to define a similarity score between entities of the two knowledge bases. For example, the YAGO triple $\langle$`m1`, `wasCreatedOnDate`, `1999-12-11`$\rangle$ forms an entity-property-literal triple. Figure 1 provides a concrete example of information within the two knowledge bases that we will keep re-using in this paper. Table 2 gives the properties and relationships for our large-scale datasets. We now define formally the problem that we address.

Definition: A *knowledge base KB* is a tuple $(\mathcal{E}, \mathcal{L}, \mathcal{R}, \mathcal{P}, \mathcal{F}_R, \mathcal{F}_P)$ where $\mathcal{E}$, $\mathcal{L}$, $\mathcal{R}$, and $\mathcal{P}$ are sets of entities, literals, relationships, and properties (respectively); $\mathcal{F}_R \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a set of relationship-facts whereas $\mathcal{F}_P \subseteq \mathcal{E} \times \mathcal{P} \times \mathcal{L}$ is a set of property-facts (both can be represented as a simple list of triples). To simplify the notation, we assume that all inverse relations are added in $\mathcal{F}_R$ – that is, if $\langle e, r, e' \rangle$ is in $\mathcal{F}_R$, we also have $\langle e', r^{-1}, e \rangle$ in $\mathcal{F}_R$.[4]

Problem: one-to-one alignment of instances between two knowledge bases. Given two knowledge bases KB_1 and KB_2 as well as a partial mapping between their corresponding relationships and properties, we want to output a 1–1 partial mapping m from $\mathcal{E}_1$ to $\mathcal{E}_2$ that represents the semantically equivalent entities in the two knowledge bases (by partial mapping, we mean that the domain of m does not have to be the whole of $\mathcal{E}_1$).

2.2 Possible approaches

Standard approaches for the ontology matching problem, such as RiMOM [20], could be used to align small knowledge

[3]The code (in Python) and datasets can be downloaded from http://mlg.eng.cam.ac.uk/slacoste/sigma.

[4]This allows us to cover all possibilities while only mentioning one standard direction of facts (like in (3) for instance).

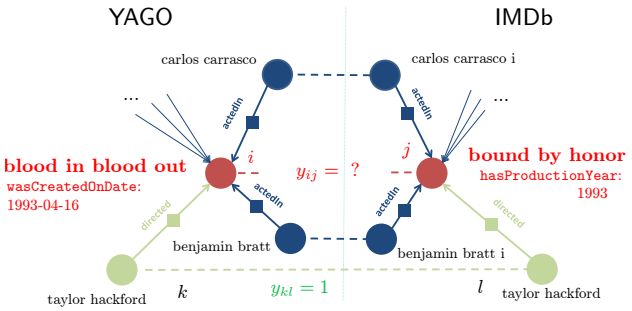

Figure 1: Example of neighborhood to match in YAGO and IMDb. Even though entities i and j have no words in common, the fact that several of their respective neighbors are matched together is a strong signal that i and j should be matched together. This is a real example from the dataset used in the experiments and SiGMa was able to correctly match all these pairs (i and j are actually the same movie despite their different stored titles in each KB).

bases. However, they do not scale to millions of entities (as needed for our task) given that they usually consider all pairs of entities and thus suffers from a quadratic scaling cost. On the other hand, the related problem of identifying duplicate entities known as *record linkage* or *duplicate detection* in the database field, and *co-reference resolution* in the natural language processing field, do have scalable solutions using *indexing* techniques [9, 13]. However, these do not typically exploit the 1–1 matching combinatorial structure present in our task, thereby reducing their accuracy. A notable exception is the work on *collective* entity resolution by Bhattacharya and Getoor [3], solved using a greedy agglomerative clustering algorithm. The algorithm SiGMa that we present in Section 3 can actually be seen as an efficient specialization of their work to the task of knowledge base alignment.

Another approach to alignment arises from the word alignment problem in natural language processing [23], which has been formulated as a quadratic assignment problem [18]. This formulation encourages neighboring entities in one graph to align with neighboring entities in the other graph while exploiting the 1–1 matching structure. This enables alignment decisions to depend on each other (see the caption of Figure 1 for an example of this in our setup). However, the quadratic assignment formulation [19], which can be solved as an integer linear program, is NP-hard in general, and these approaches were only used to align at most one hundred entities. In the algorithm SiGMa that we propose, we are interested in exploiting both the 1–1 matching constraint, as well as building on previous decisions, like these word alignment approaches, but in a scalable manner which would handle millions of entities. SiGMa does this by greedily optimizing the quadratic assignment objective, as we describe in Section 3.1. Finally, Suchanek et al. [27] recently proposed an ontology matching approach called PARIS that they have succeeded to apply on the alignment of YAGO to IMDb as well. But the scalability of their approach is not as clear, as we explain in Section 5. We provide a detailed comparison with PARIS in the experiments section.

3. THE SIGMA ALGORITHM

3.1 Greedy optimization of a quadratic assignment objective

The SiGMa algorithm can be seen as the greedy optimization of an objective function that globally scores the suitability of a particular matching m for a pair of given KBs. This objective function uses two sources of information useful for choosing matches: a similarity function between pairs of entities defined from their properties; and a graph neighborhood contribution making use of neighbor pairs being matched (see Figure 1 for a motivation). Let us encode the matching $m : \mathcal{E}_1 \to \mathcal{E}_2$ by a matrix y with entries indexed by the entities in each KB, with $y_{ij} = 1$ if $m(i) = j$, meaning that $i \in \mathcal{E}_1$ is matched to $j \in \mathcal{E}_2$, and $y_{ij} = 0$ otherwise. The space of possible 1–1 partial mappings is thus represented by the following set of binary matrices: $\mathcal{M} \doteq \{y \in \{0,1\}^{\mathcal{E}_1 \times \mathcal{E}_2} : \sum_l y_{il} \leq 1 \ \forall i \in \mathcal{E}_1$ and $\sum_k y_{kj} \leq 1 \ \forall j \in \mathcal{E}_2\}$. We define the following quadratic objective function that globally scores the suitability of a matching y:

$$\texttt{obj}(y) \doteq \sum_{(i,j)\in\mathcal{E}_1\times\mathcal{E}_2} y_{ij}\left[(1-\alpha)s_{ij} + \alpha g_{ij}(y)\right], \quad \text{where} \quad g_{ij}(y) \doteq \sum_{(k,l)\in\mathcal{N}_{ij}} y_{kl}\, w_{ij,kl}. \tag{1}$$

The objective contains linear coefficients s_{ij} which encode a similarity between entity i and j, as well as quadratic coefficients $w_{ij,kl}$ which control the algorithm's tendency to match i with j given that k was matched to l[5]. $\mathcal{N}_{ij}$ is a local neighborhood around (i,j) that we define later in (3) and which depends on the graph information from the KBs. $g_{ij}(y)$ is basically counting (in a weighted fashion) the number of matched pairs (k,l) which are in the neighborhood of i and j. $\alpha \in [0,1]$ is a tradeoff parameter between the linear and quadratic contributions. Our approach is motivated by the maximization problem:

$$\max_y \quad \texttt{obj}(y) \quad \text{s.t. } y \in \mathcal{M}, \quad \|y\|_1 \leq R, \tag{2}$$

where the norm $\|y\|_1 \doteq \sum_{ij} y_{ij}$ represents the number of elements matched and R is an unknown upper-bound which represents the size of the best partial mapping which can be made from KB_1 to KB_2. We note that if the coefficients are all positive (as will be the case in our formulation – we are only encoding similarities and not repulsions between entities), then the maximizer y^* satisfies $\|y^*\|_1 = R$. Problem (2) is thus related to one of the variations of the quadratic assignment problem, a well-known NP-complete problem in operations research [19]. Even though one could approximate the solution to (2) using a linear program relaxation (see Lacoste-Julien et al. [18]), the number of variables is quadratic in the number of entities, so this is obviously not scalable. Our approach is instead to *greedily optimize* (2) by adding the match element $y_{ij} = 1$ at each iteration which increases the objective the most and selected amongst a small set of possibilities. In other words, the high-level operational definition of the SiGMa algorithm is as follows:

1. Start with an initial good quality partial match y_0.

[5] In the rest of this paper, we use the convention that i and k are always entities in KB_1; whereas j and l are in KB_2. e could be in either KB.

2. At each iteration t, augment the previous matching with a new matched pair by setting $y_{ij} = 1$ for the (i, j) which maximally increases `obj`, chosen amongst a small set $\mathcal{S}_t$ of reasonable candidates which preserve the feasibility of the new matching.
3. Stop when the bound $\|y\|_1 = R$ is reached (and never undo previous decisions).

Having outlined the general framework, we now describe our choice of neighborhood $\mathcal{N}_{ij}$, candidate set $\mathcal{S}_t$, and stopping criterion R. We describe methods for choosing the similarity coefficients s_{ij} and $w_{ij,kl}$ so that they guide the algorithm towards good matchings in Section 3.3. These choices influence both the speed and accuracy of the algorithm.

Compatible-neighbors. $\mathcal{N}_{ij}$ should be chosen so as to respect the graph structure defined by the KB facts. Its contribution in the objective crucially encodes the fact that a neighbor k of i being matched to a 'compatible' neighbor l of j should encourage i to be matched to j — see the caption of Figure 1 for an example. Here, compatibility means that they are related by the same relationship (they have the same color in Figure 1). Formally, we define:

$$\mathcal{N}_{ij} = \texttt{compatible-neighbors}(i, j) \doteq \{ (k, l) : \langle i, r, k\rangle \text{ is in } \mathcal{F}_{R1} \text{ and } \langle j, s, l\rangle \text{ is in } \mathcal{F}_{R2} \text{ and relationship } r \text{ is matched to } s\}. \quad (3)$$

Note that a property of this neighborhood is that $(k, l) \in \mathcal{N}_{ij}$ iff $(i, j) \in \mathcal{N}_{kl}$, as we have that the relationship r is matched to s iff r^{-1} is matched to s^{-1} as well. This means that the *increase* in the objective obtained by adding (i, j) to the current matching y defines the following *context dependent similarity score function* that is used to pick the next matched pair in the step 2 of the algorithm:

$$\texttt{score}(i, j; y) = (1 - \alpha)s_{ij} + \alpha\, \delta g_{ij}(y) \quad \text{where } \delta g_{ij}(y) \doteq \sum_{(k,l)\in\mathcal{N}_{ij}} y_{kl}\,(w_{ij,kl} + w_{kl,ij}). \quad (4)$$

Information propagation on the graph. The `compatible-neighbors` concept that we just defined is one of the most crucial characteristics of SiGMa. It allows the information of a new matched pair to propagate amongst its neighbors. It also defines a powerful heuristic to suggest new candidate pairs to include in a small set $\mathcal{S}_t$ of matches to choose from: after matching i to j, SiGMa adds all the pairs (k, l) from $\texttt{compatible-neighbors}(i, j)$ as new candidates. This yields a fire propagation analogy for the algorithm: starting from an initial matching (fire), the algorithm starts to match their neighbors, letting the fire propagate through the graph. If the graph in each KB is well-connected in a similar fashion, it can visit most nodes this way. This heuristic enables SiGMa to avoid the potential quadratic number of pairs to consider by only focussing its attention on the neighborhoods of current matches. In the language of the record linkage literature [8], this amounts to an *iterative blocking* technique.

Stopping criterion. R is implicitly chosen by the following heuristic motivated from standard optimization algorithms: SiGMa terminates when the variation in the objective value, $\texttt{score}(i, j; y)$, of the latest added match (i, j) falls below a threshold (or the queue becomes empty). The threshold in effect controls the precision/recall tradeoff of the algorithm. By ensuring that the s_{ij} and $g_{ij}(y)$ terms are normalized between 0 and 1, we can standardize the scale of the threshold for different score functions. We used a threshold of 0.25 in our experiments. It appeared to correlate well with a transition point at which the F-measure stops increasing and the precision starts to decrease significantly on a wide variety of datasets.

1: Initialize matching $m = m_0$.
2: Initialize priority queue $\mathcal{S}$ of suggested candidate pairs as $\mathcal{S}_0 \cup \left(\bigcup_{(i,j)\in m} \mathcal{N}_{ij}\right)$ – the `compatible-neighbors` of pairs in m, with $\texttt{score}(i, j; m)$ as their key.
3: **while** priority queue $\mathcal{S}$ is not empty **do**
4: Extract $\langle \texttt{score}, i, j\rangle$ from queue $\mathcal{S}$
5: **if** $\texttt{score} \leq \texttt{threshold}$ **then stop**
6: **if** i or j is already matched to some entity **then**
7: skip them and **continue** loop
8: **else**
9: Set $m(i) = j$.
{We update candidate lists and scores:}
10: **for** (k, l) in $\mathcal{N}_{ij}$ and not already matched **do**
11: Add $\langle \texttt{score}(k, l; m), k, l\rangle$ to queue $\mathcal{S}$.

Table 1: SiGMa algorithm.

3.2 Algorithm and implementation

We present the pseudo-code for SiGMa in Table 1. We now elaborate on the algorithm design as well as its implementation aspects. We note that the `score` defined in (4) to greedily select the next matched pair is composed of a static term s_{ij}, which does not depend on the evolving matching y, and a dynamic term $\delta g_{ij}(y)$, which depends on y, though only through the local neighborhood $\mathcal{N}_{ij}$. We call the δg_{ij} component of the score function the graph contribution; its local dependence means that it can be updated efficiently after a new match has been added. We describe the choice of similarity measures for these components in Section 3.3.

Initial match structure m_0. The algorithm can take any initial matching seed assumed of good quality. In our current implementation, this is done by looking for entities with the same string representation (with minimal standardization like removing capitalization and punctuation) with an *unambiguous 1–1 match* – that is, we do not include an exact matched pair when more than two entities have this same string representation, thereby increasing precision.

Optional static list of candidates $\mathcal{S}_0$. Optionally, we can initialize $\mathcal{S}$ with a static list $\mathcal{S}_0$ which only needs to be scored once as any score update will come from neighbors already covered by step 11 of the algorithm. $\mathcal{S}_0$ has the purpose to increase the possible exploration of the graph when another strong source of information (which is not from the graph) can be used, and corresponds to the standard *blocking* (or *indexing*) heuristics in record linkage [9]. In our implementation, we use an inverted index built on words to efficiently suggest entities which have at least two words in common in their string representation as potential candidates. More powerful indexing heuristics could also be used.

Data-structures. We use a binary heap for the priority queue implementation, allowing insertions in $O(\log n)$ where n is the size of the queue. Because the score function can only increase as we add new matches, we do not need to keep track of stale nodes in the priority queue in order to update their scores, yielding a significant speed-up.

3.3 Score functions

An important factor for any matching algorithm is the similarity function between pairs of elements to match. Designing good similarity functions has been the focus of much of the literature on record linkage, entity resolution, etc., and because SiGMa uses the score function in a modular fashion, SiGMa is free to use most of them for the term s_{ij} as long as they can be computed efficiently. We provide in this section our implementation choices (which were motivated by simplicity), but we note that the algorithm can easily handle more powerful similarity measures. The generic score function used by SiGMa was given in (4). In the current implementation, the static part s_{ij} is defined through the *properties* of entities only. The graph part $\delta g_{ij}(y)$ depends on the *relationships* between entities (as this is what determines the graph), as well as the previous matching y. We also make sure that s_{ij} and g_{ij} stay normalized so that the score of different pairs are on the same scale.

3.3.1 Static similarity measure

The static property similarity measure is further decomposed in two parts: we single out a contribution coming from the string representation property of entities (as it is such a strong signal for our datasets), and we consider the other properties together in a second term:

$$s_{ij} = (1-\beta)\texttt{string}(i,j) + \beta\texttt{prop}(i,j), \tag{5}$$

where $\beta \in [0,1]$ is a tradeoff coefficient between the two contributions set to 0.25 during the experiments.

String similarity measure. For the string similarity measure, we primarily consider the number of words that two strings have in common, albeit weighted by their information content. In order to handle the varying lengths of strings, we use the Jaccard similarity coefficient between the sets of words, with a smoothing term. To capture the information that some words are more informative than others, we use the IDF (inverse-document-frequency) weight for each word in a *weighted* Jaccard measure, a commonly used feature in information retrieval. The weight for word v in KB_o is $w_v^o \doteq \log_{10}|\mathcal{E}_o|/|E_v^o|$, where $E_v^o \doteq \{e \in \mathcal{E}_o : e \text{ has word } v \text{ in its string representation}\}$. Combining these elements, we get the following string similarity measure:

$$\texttt{string}(i,j) = \frac{\sum_{v\in(\mathcal{W}_i\cap\mathcal{W}_j)}(w_v^1 + w_v^2)}{\texttt{smoothing} + \sum_{v\in\mathcal{W}_i} w_v^1 + \sum_{v'\in\mathcal{W}_j} w_{v'}^2}, \tag{6}$$

where $\mathcal{W}_e$ is the set of words in the string representation of entity e and `smoothing` is the scalar smoothing constant (we try different values in the experiments). While this measure is robust to word re-ordering, it is not robust to variations of spelling for words. This problem could be addressed by using more involved string similarity measures as described in [10], though our current implementation only uses (6) for simplicity. We also explore the effect of different scoring functions in our experiments in Section 4.5.

Property similarity measure. We recall that we assume that the user provides a partial matching between properties of both databases. This enables us to use them in a property similarity measure. In order to elegantly handle missing values of properties, a varying number of property values present, etc., we also use a smoothed weighted Jaccard similarity measure between the sets of properties. The detailed formulation is given in Appendix A of the longer technical report [17] for completeness, but we note that it can make use of a similarity measure between literals like a normalized distance on numbers (for dates, years, and so on) or an edit distance on strings.

3.3.2 Dynamic graph similarity measure

We now introduce the part of the score function which enables SiGMa to build on previous decisions and exploit the relationship graph information. We need to determine $w_{ij,kl}$, the weight of the contribution of a neighboring matched pair (k,l) for the score of the candidate pair (i,j). The general idea of the graph score function is to count the number of compatible neighbors which are currently matched together for a pair of candidates (this is the $g_{ij}(y)$ contribution in (1)). Going back to the example in Figure 1, there were three compatible matched pairs shown in the neighborhood of i and j. We would like to normalize this count by dividing by the number of possible neighbors, and we may want to weight each neighbor differently. We again use a smoothed weighted Jaccard measure to summarize this information, averaging the contribution from each KB. This can be obtained by defining $w_{ij,kl} = \gamma_i w_{ik} + \gamma_j w_{jl}$, where γ_i and γ_j are normalization factors specific to i and j in each database and w_{ik} is the weight of the contribution of k to i in KB_1 (and similarly for w_{jl} in KB_2). The graph contribution thus becomes:

$$g_{ij}(y) = \sum_{(k,l)\in\mathcal{N}_{ij}} y_{kl}(\gamma_i w_{ik} + \gamma_j w_{jl}). \tag{7}$$

Let $\mathcal{N}_i$ be the set of neighbors of entity i in KB_1, i.e. $\mathcal{N}_i \doteq \{k : \exists r \text{ s.t. } (i,r,k) \in \mathcal{F}_{R1}\}$ (and similarly for $\mathcal{N}_j$). Then, remembering that $\sum_k y_{kl} \leq 1$ for a valid partial matching $y \in \mathcal{M}$, the following normalizations γ_i and γ_j yield the average of two smoothed weighted Jaccard measures for $g_{ij}(y)$:

$$\gamma_i \doteq \frac{1}{2}\left(1 + \sum_{k\in\mathcal{N}_i} w_{ik}\right)^{-1} \quad \gamma_j \doteq \frac{1}{2}\left(1 + \sum_{l\in\mathcal{N}_j} w_{jl}\right)^{-1}. \tag{8}$$

We thus have $g_{ij}(y) \leq 1$ for $y \in \mathcal{M}$, keeping the contribution of each possible matched pair (i,j) on the same scale in `obj` in (1).

The graph part of the score in (4) then takes the form:

$$\delta g_{ij}(y) = \sum_{(k,l)\in\mathcal{N}_{ij}} y_{kl}\left(\gamma_i w_{ik} + \gamma_j w_{jl} + \gamma_k w_{ki} + \gamma_l w_{lj}\right). \tag{9}$$

The summation over the first two terms yields $g_{ij}(y)$ and so is bounded by 1, but the summation over the last two terms could be greater than 1 in the case that (i,j) is filling a 'hole' in the graph (thus increasing the contribution of many neighbors (k,l) in `obj` in (1)). Finally, we use unit weight for w_{ik}. See Section 4.5 and Appendix B of the longer technical report [17] for alternatives (which did not perform better in experiments).

4. EXPERIMENTS

4.1 Setup

We made a prototype implementation of SiGMa in Python[6] and compared its performance on benchmark datasets as

[6]The code and datasets can be downloaded from http://mlg.eng.cam.ac.uk/slacoste/sigma.

well as on large-scale knowledge bases. All experiments were run on a cluster node Hexacore Intel Xeon E5650 2.66GHz with 46GB of RAM running Linux. Each knowledge base is represented as two text files containing a list of triples of relationships-facts and property-facts. The input to SiGMa is a pair of such KBs as well as a partial mapping between the relationships and properties of each KB which is used in the computation of the score in (4), and the definition of `compatible-neighbors` (3). The output of SiGMa is a list of matched pairs (e_1, e_2) with their score information and the iteration number at which they were added to the solution. We evaluate the final alignment (after reaching the stopping threshold) by comparing it to a ground truth using the standard metrics of precision, recall and F-measure on the number of correctly matched *entities in the first KB with ground truth information*. The benchmark datasets are available together with corresponding ground truth data; for the large-scale knowledge bases, we built their ground truth using web URL information as described in Section 4.2.

We found reasonable values for the parameters of SiGMa by exploring its performance on the YAGO to IMDb pair (the methodology is described in Section 4.5), and then kept them fixed for all the other experimental comparisons (Section 4.3 and 4.4). This reflects the situation where one would like to apply SiGMa to a new dataset without ground truth or to minimize parameter adaptation. The standard parameters that we used in these experiments are given in Appendix D of [17] for reproducibility.

4.2 Datasets

Our experiments were done both on several large-scale datasets and on some standard benchmark datasets from the ontology alignment evaluation initiative (OAEI) (Table 3). We describe these datasets below.

Large-scale datasets. As mentioned throughout this paper so far, we used the dataset pair YAGO-IMDb as the main motivating example for developing and testing SiGMa. We also test SiGMa on the pair Freebase-IMDb, for which we could obtain a sizable ground truth. We describe here their construction. Both YAGO and Freebase are available as lists of triples from their respective websites.[7] IMDb, on the other hand, is given as a list of text files.[8] There are different files for different categories, e.g.: actors, producers, etc. We use these categories to construct a list of triples containing facts about movies and people. Because SiGMa ignores relationships and properties that are not matched between the KBs, we could reduce the size of YAGO and Freebase by keeping only those facts for which their relationship had a 1–1 mapping with IMDb as presented in Table 2, and the entities appearing in these facts. To facilitate the comparison of SiGMa with PARIS, the authors of PARIS kindly provided us their own version of IMDb that we refer to as IMDb_PARIS — this version actually has a richer structure in terms of properties. We also kept in YAGO the relationships and properties that were aligned with those of IMDb_PARIS (Table 2). Table 3 presents the number of unique entities and relationship-facts included in the relevant reduced datasets. We constructed the ground truth for YAGO-IMDb by scraping the relevant Wikipedia pages of entities to extract their link to the corresponding IMDb page, which often appears in the 'external links' section. We then obtained the entity name by scraping the corresponding IMDb page and matched it to our constructed database by using string matching (and some manual cleaning). We obtained 54k ground truth pairs this way. We used a similar process for Freebase-IMDb by accessing the IMDb URLs which were actually stored in the database. This yielded 293k pairs, probably one of the largest knowledge base alignment ground truth sets to date.

[7] YAGO2 core was downloaded from: http://www.mpi-inf.mpg.de/yago-naga/yago/downloads.html and Freebase from: http://wiki.freebase.com/wiki/Data_dumps.

[8] http://www.imdb.com/interfaces#plain

YAGO	IMDb_PARIS	IMDb	Freebase
`Relations`			
actedIn	actedIn	actedIn	actedIn
directed	directorOf	directed	directed
produced	producerOf	produced	produced
created	writerOf	composed	
wasBornIn	bornIn		
diedIn	deceasedIn		
capitalOf	locatedIn		
`Properties`			
hasLabel	hasLabel	hasLabel	hasLabel
wasCreatedOnDate		hasProductionYear	initialReleaseDate
wasBornOnDate	bornOn		
diedOnDate	deceasedOn		
hasGivenName	firstName		
hasFamilyName	lastName		
hasGender	gender		
hasHeight	hasHeight		

Table 2: Manually aligned movie related relationships and properties in large-scale KBs.

Dataset	#facts	#entities
YAGO	442k	1.4M
IMDb_PARIS	20.9M	4.8M
IMDb	9.3M	3.1M
Freebase	1.5M	474k

(a) Large-scale datasets

Dataset	#facts	#entities
DBLP	2.5M	1.6M
Rexa	12.6k	14.7k
person11	500	1000
person12	500	1000
restaurant1	113	339
restaurant2	752	2256

(b) Benchmark datasets

Table 3: Datasets statistics

Benchmark datasets. We also tested SiGMa on three benchmark dataset pairs provided by the ontology alignment evaluation initiative (OAEI), which allowed us to compare the performance of SiGMa to some previously published methods [20, 14]. From the OAEI 2009 edition,[9] we use the Rexa-DBLP instance matching benchmark from the domain of scientific publications. Rexa contains publications and authors as entities extracted from the search results of the Rexa search server. DBLP is a version of the DBLP dataset listing publications from the computer science domain. The pair has one matched relationship, `author`, as well several matched properties such as `year`, `volume`, `journal name`, `pages`, etc. Our goal was to align publications and authors. The other two datasets come from the Person-Restaurants (PR) task from the OAEI 2010 edition,[10] containing data about people and restaurants. In particular, there are person11-person12 pairs where the second entity is a copy of the first with one property field corrupted, and restaurant1-restaurants2 pairs coming from two different online databases that were manually aligned. All datasets were downloaded from the corresponding OAEI webpages, with dataset sizes given in Table 3.

4.3 Exp. 1: Large-scale alignment

In this experiment, we test the performance of SiGMa on the three pairs of large-scale KBs and compare it with PARIS [27], which is described in more details in the re-

[9] http://oaei.ontologymatching.org/2009/instances/

[10] http://oaei.ontologymatching.org/2010/im/index.html

Dataset	System	Prec	Rec	F	GT size	# pred.	Time
Freebase-IMDb	SiGMa	98	95	97	255k	373k	50 min
	Exact-string	99	70	82		244k	5 min
YAGO-IMDb	SiGMa	98	93	95	54k	198k	45 min
	Exact-string	99	57	72		162k	5 min
YAGO-IMDb_PARIS (new ground truth)	SiGMa	98	96	97	57k	256k	70 min
	PARIS	97	96	97		702k	3100 min
	Exact-string	99	56	72		202k	10 min
YAGO-IMDb_PARIS (ground truth from [27])	SiGMa	98	85	91	11k	same as above	
	PARIS	94	90	92			
	Exact-string	99	61	75			

Table 4: Exp. 1: Results (precision, recall, F-measure) on large-scale datasets for SiGMa in comparison to a simple exact-matching phase on strings as well as PARIS [27]. The 'GT Size' column gives the number entities with ground truth information. Time is total running time, including loading the dataset (quoted from [27] for PARIS).

lated work Section 5. We also compare SiGMa and PARIS with the simple baseline of doing the unambiguous exact string matching step described in Section 3.2 which is used to obtain an initial match m_0 (called Exact-string). Table 4 presents the results. Despite its simple greedy nature which never goes back to correct a mistake, SiGMa obtains an impressive F-measure above 90% for all datasets, significantly improving over the Exact-string baseline. We tried running PARIS [27] on a smaller subset of YAGO-IMDb, using the code available from its author's website. It did not complete its first iteration after a week of computation and so we halted it (we did not have the SSD drive which seems crucial to reasonable running times). The results for PARIS in Table 4 are thus computed using the prediction files provided to us by its authors on the YAGO-IMDb_PARIS dataset. In order to better relate the YAGO-IMDb_PARIS results with the YAGO-IMDb ones, we also constructed a larger ground truth reference on YAGO-IMDb_PARIS by using the same process described in Section 4.2. On both ground truth evaluations, SiGMa obtains a similar F-measure to PARIS, but in 50x less time. On the other hand, we note that PARIS is solving the more general problem of instance and schema alignment, and was not provided any manual alignment between relationships. The large difference of recall between PARIS and SiGMa on the ground truth from [27] can be explained by the fact that more than a third of its entities had no neighbor, whereas the process used to construct the new larger ground truth included only entities participating in movie facts and thus having at least one neighbor. The recall of SiGMa actually increases for entities with increasing number of neighbors (going from 70% for entities in the ground truth from [27] with 0 neighbors to 95% for entities with 4+ neighbors).

About 2% of the predicted matched pairs from SiGMa on YAGO-IMDb have no word in common and thus zero string similarity – difficult pairs to match without any graph information. Examples of these pairs came from spelling variations of names, movie titles in different languages, foreign characters in names which are not handled uniformly, or multiple titles for movies (such as the 'Blood In, Blood Out' example of Figure 1).

Error analysis. Examining the few errors made by SiGMa, we observed the following types of matching errors: 1) errors in the ground truth (either coming from the scraping scheme used; or from Wikipedia (YAGO) which had incorrect information); 2) errors with multiple very similar entities (e.g. mistaking the 'making of' the movie vs. the movie itself); 3) errors with pairs of entities that shared exactly the same neighbors (e.g. two different movies with exactly the same actors) but without other discriminating information. Finally, we note that going through the predictions of SiGMa that had a low property score revealed a significant number of errors in the databases (e.g. wildly inconsistent birth dates for people), indicating that SiGMa could be used to highlight data inconsistencies between databases.

Dataset	System	Prec	Rec	F	GT size
Person	SiGMa	100	100	100	500
	PARIS	100	100	100	
Restaurant	SiGMa-linear	100	100	100	89
	SiGMa	99	94	97	
	PARIS	95	88	91	
	Exact-string	100	75	86	
Rexa-DBLP	SiGMa	99	94	96	8656
	SiGMa-linear	97	95	96	
	Exact-string	99	89	94	
	RiMOM	97	74	84	

Table 5: Exp. 2: Results on the benchmark datasets for SiGMa, compared with PARIS [27] and RiMOM [20]. SiGMa-linear and Exact-string are also included on the interesting datasets as further comparison points.

4.4 Exp. 2: Benchmark comparisons

In this experiment, we test the performance of SiGMa on the three benchmark datasets and compare them with the best published results so far that we are aware of: PARIS [27] for the Person-Restaurants datasets (which compared favorably over ObjectCoref [14]); and RiMoM [20] for Rexa-DBPL. Table 5 presents the results. We also include the results for Exact-string as a simple baseline as well as SiGMa-linear, which is the SiGMa algorithm without using the graph information at all,[11] to give an idea of how important the graph information is in these cases.

Interestingly, SiGMa significantly improved the previous results without needing any parameter tweaking. The Person-Restaurants datasets did not have a rich relationship structure to exploit: each entity (a person or a restaurant) was linked to exactly one another in a 1–1 bipartite fashion (their address). This is perhaps why SiGMa-linear is surprisingly able to *perfectly* match both these datasets.

For the Rexa-DBLP dataset, we note that the organizers of the OAEI 2009 edition built the augmented ground truth set that we have used by inspection of the predictions of the submitted systems (and probably by using exact string matching heuristics as well). SiGMa discovered about a thousand new matches that were not present in this ground truth but that appeared to be mostly correct. The recall for Exact-string and RiMOM is thus probably overestimated compared to the one of SiGMa. This benchmark which has a medium size also highlights the nice scalability of SiGMa: despite using the interpreted language Python, our implementation runs in less than 10 minutes on this dataset. By comparison, RiMOM took 36 hours on a 8-core server in 2009 [20].

4.5 Parameter experiments

In this section, we explore the role of different configurations for SiGMa on the YAGO-IMDb pair, as well as deter-

[11] SiGMa-linear is not using the graph score component (α is set to 0) and is not using the neighbors in $\mathcal{N}_{ij}$ to suggest candidates (it only uses the inverted index $\mathcal{S}_0$).

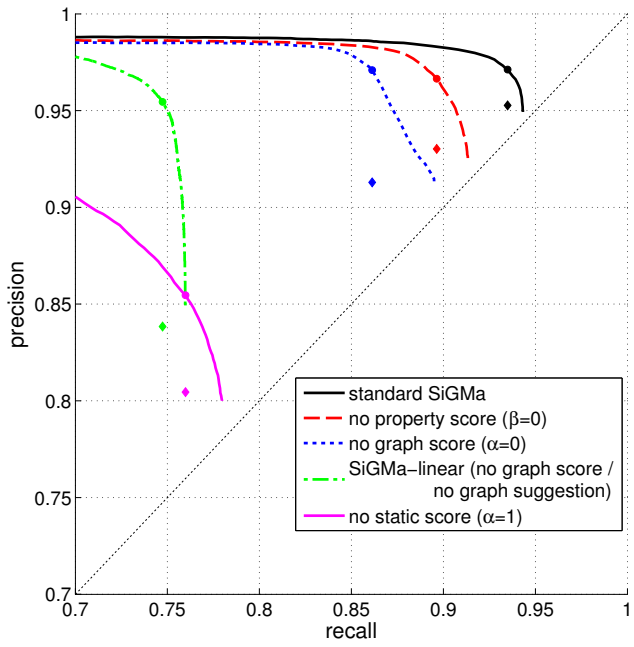

Figure 2: Exp. 3: Precision/recall curves for SiGMa on YAGO-IMDb with different scoring configurations. The filled circles indicate the maximum F-measure position on each curve, with the corresponding diamond giving the F-measure value on the y-axis at this recall point.

mine which parameters to use for the other experiments. We recall that SiGMa with the final parameters yields a 95% F-measure on this dataset (second section of Table 4). Experiments 5 and 6, which explore the optimal weighting schemes as well as the correct stopping threshold, are described for completeness in Appendix E of [17].

4.5.1 Exp. 3: Score components

In this experiment, we explore the importance of each part of the score function by running SiGMa with some parts turned off (this can be done by setting the α and β tradeoffs to 0 or 1). The resulting precision/recall curves are plotted in Figure 2. By comparing SiGMa with SiGMa-linear, we see that including the graph information moves the F-measure from a bit below 85% to over 95%, a significant gain, indicating that the graph structure is more important on this challenging dataset than the easier OAEI benchmark datasets and it was crucial to exploit it.

4.5.2 Exp. 4: Matching seed

In this experiment, we test how important the size of the matching seed m_0 is for the performance of SiGMa. We report the following notable results. We ran SiGMa with no exact seed matching at all: we initialized it with a random exact match pair and let it explore the graph greedily (with the inverted index still making suggestions). This obtained an even better score than the standard setup: 99% of precision, 94% recall and 96% F-measure, demonstrating that *a good initial seed is actually not needed for this setup*, and illustrating the power of the graph information for this dataset. We observed a similar improvement for the Freebase and YAGO-IMDb_PARIS datasets (with the new ground truth).

5. RELATED WORK

We contrast here SiGMa with the work already mentioned in Section 2.2 and provide further links. In the ontology matching literature, the only approach which was applied to datasets of the size that we considered in this paper is the recently proposed PARIS [27], which solves the more general problem of matching instances, relationships and classes. The PARIS framework defines a normalized score between pairs of instances that represents how likely they are to be matched,[12] and that depends on the matching scores of their compatible neighbors. The final scores are obtained by first initializing (and fixing) the scores on pairs of literals, and then propagating the updates through the relationship graph using a fixed point iteration, yielding an analogous fire propagation of information as SiGMa, though it works with soft [0-1]-valued assignments whereas SiGMa works with hard {0,1}-valued ones. The authors handle the scalability issue of maintaining scores for all pairs by using a sparse representation with various pruning heuristics (in particular, keeping only the maximal assignment for each entity at each step, *thus making the same 1–1 assumption that we did*). An advantage of PARIS over SiGMa is that it is able to include property values in its neighborhood graph (it uses soft-assignments between them) whereas SiGMa only uses relationships given that a 1–1 matching of property values is not appropriate. We conjecture that this could explain the higher recall that PARIS obtained on entities that had no relationship neighbors on the YAGO-PARIS_IMDB dataset. On the other hand, PARIS was limited to use a 0-1 similarity measure between property values for the large-scale experiments in [27], as it is unclear how one could apply the same sparsity optimization in a scalable fashion with more involved similarity measures (such as the IDF one that SiGMa is using). The use of a 0-1 similarity measure on strings could explain the lower performance of PARIS on the Restaurants dataset in comparison to SiGMa. We stress that SiGMa is able in contrast to use sophisticated similarity measures in a scalable fashion, had a 50x speed improvement over PARIS on the large-scale datasets, and yields a significantly simpler implementation.

The SiGMa algorithm is related to the collective entity resolution approach of Bhattacharya and Getoor [3], who proposed a greedy agglomerative clustering algorithm to cluster entities based on previous decisions. Their approach could handle constraints on the clustering, including a 1–1 matching constraint in theory, though it was not implemented. We think a contribution of our work is to demonstrate the effectiveness of using the 1–1 matching constraint for knowledge base alignment. Some scalable solutions for collective entity resolution were proposed recently [1, 25], though they did not implement a 1–1 matching constraint, and their implementation can be a complex software engineering endeavor in contrast to the simplicity of our approach.

The idea to propagate information on a relationship graph has been used in several other approaches for ontology matching [15, 21], though none were scalable for the size of knowledge bases that we considered. An analogous 'fire propagation' algorithm has been used to align social network graphs in [22], though with a very different objective function (they define weights in each graph and want to align edges which

[12]The authors call these 'marginal probabilities' as they were motivated from probabilistic arguments, but these do not sum to one.

have similar weights). The heuristic of propagating information on a relationship graph is related to a well-known heuristic for solving constraint satisfaction problems known as constraint propagation [2]. Ehrig and Staab [11] mentioned several heuristics to reduce the number of candidates to consider in ontology alignment, including a similar one to `compatible-neighbors`, though they tested their approach only on a few hundred instances. Finally, we mention that Peralta [24] aligned the movie database MovieLens to IMDb through a combination of steps of manual cleaning with some automation. SiGMa could be considered as an alternative which does not require manual intervention apart from specifying the score function to use.

Böhm et al. have recently proposed a similar algorithm to SiGMA called LINDA [5], independently to our prior technical report [17]. LINDA is also an iterative greedy algorithm for a quadratic assignment objective applied to instance matching, but with slightly different scoring functions. We think that their work complements ours: they have demonstrated how to scale a SiGMa-like algorithm to billions of triples using the MapReduce framework. On the other hand, we provide a significantly simpler implementation for single machines with experiments demonstrating that higher accuracy can be achieved (they reported only 80% F-measure on the Restaurants dataset, as well as 66% sampled precision for their large-scale experiments, whereas the sampled precision we measured was actually always above 90%).

6. CONCLUSION

We have presented SiGMa, a simple and scalable algorithm for the alignment of large-scale knowledge bases. Despite making greedy decisions and never backtracking to correct decisions, SiGMa obtains a higher F-measure than the previously best published results on the OAEI benchmark datasets, and matches the performance of the more involved algorithm PARIS while being 50x faster on large-scale knowledge bases of millions of entities. Our experiments indicate that SiGMa can obtain good performance over a range of datasets with the same parameter setting. On the other hand, SiGMa is easily extensible to more powerful scoring functions between entities, as long as they can be efficiently computed.

Some apparent limitations of SiGMa are that it a) cannot correct previous mistakes and b) cannot handle alignments other than 1–1. Addressing these in a scalable fashion which preserves high accuracy are open questions for future work. We note though that the non-corrective nature of the algorithm didn't seem to be an issue in our experiments. Moreover, pre-processing each knowledge base with a de-duplication method can help make the 1–1 assumption, which is a powerful feature to exploit in an alignment algorithm, more reasonable. Another interesting direction for future work would be to use machine learning methods to learn the parameters of more powerful scoring function. In particular, the 'learning to rank' model seems suitable to learn a score function which would rank the correctly labeled matched pairs above the other ones. The current level of performance of SiGMa already makes it suitable though as a powerful generic alignment tool for knowledge bases and hence takes us closer to the vision of Linked Open Data and the Semantic Web.

Acknowledgments: We thank Fabian Suchanek and Pierre Senellart for sharing their code and answering our questions about PARIS. We thank Guillaume Obozinski for helpful discussions. This research was supported by a grant from Microsoft Research Ltd. and a Research in Paris fellowship.

7. REFERENCES

[1] A. Arasu, C. Ré, and D. Suciu. Large-scale deduplication with constraints using dedupalog. In *ICDE*, 2009.
[2] C. Bessiere. Constraint propagation. *Foundations of Artificial Intelligence*, 2:29–83, 2006.
[3] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *ACM TKDD*, 1(1), 2007.
[4] C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee. Linked data on the web (LDOW2008). In *WWW*, 2008.
[5] C. Böhm, G. de Melo, et al. LINDA: distributed web-of-data-scale entity matching. In *CIKM*, 2012.
[6] S. Castano, A. Ferrara, D. Lorusso, and S. Montanelli. On the ontology instance matching problem. In *DEXA*, 2008.
[7] N. Choi, I.-Y. Song, and H. Han. A survey on ontology mapping. *SIGMOD Rec.*, 35:34–41, 2006.
[8] P. Christen. *Data Matching*. 2012.
[9] P. Christen. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE TKDE*, 24(9), 2012.
[10] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proc. of IJCAI-03 Work. on Inf. Integ. on the Web*, 2003.
[11] M. Ehrig and S. Staab. QOM – quick ontology mapping. In *ISWC*, 2004.
[12] J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag, 2007.
[13] J. Gracia, M. d'Aquin, and E. Mena. Large scale integration of senses for the semantic web. In *WWW*, 2009.
[14] W. Hu, J. Chen, and Y. Qu. A self-training approach for resolving object coreference on the semantic web. In *WWW*, 2011.
[15] W. Hu, N. Jian, Y. Qu, and Y. Wang. GMO: A graph matching for ontologies. In *K-Cap Workshop in Integrating Ontologies*, 2005.
[16] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *Knowl. Eng. Rev.*, 18:1–31, 2003.
[17] S. Lacoste-Julien et al. SiGMa: Simple Greedy Matching for Aligning Large Knowledge Bases. arXiv:1207.4525, 2012.
[18] S. Lacoste-Julien, B. Taskar, D. Klein, and M. I. Jordan. Word alignment via quadratic assignment. In *NAACL*, 2006.
[19] E. L. Lawler. The quadratic assignment problem. *Management Science*, 9(4):586–599, 1963.
[20] J. Li, J. Tang, Y. Li, and Q. Luo. Rimom: A dynamic multistrategy ontology alignment framework. *IEEE Trans. on Knowl. and Data Eng.*, 21:1218–1232, 2009.
[21] M. Mao. Ontology mapping: An information retrieval and interactive activation network based approach. In *ISWC/ASWC*, 2007.
[22] A. Narayanan, E. Shi, and B. I. P. Rubinstein. Link prediction by de-anonymization: How we won the Kaggle social network challenge. In *IJCNN*, 2011.
[23] F. J. Och and H. Ney. A systematic comparison of various statistical alignment models. *Comput. Linguist.*, 29, 2003.
[24] V. Peralta. Matching of MovieLens and IMDb movie titles. Technical report, Université de Versailles, 2007.
[25] V. Rastogi, N. Dalvi, and M. Garofalakis. Large-scale collective entity matching. *VLDB*, 4:208–218, 2011.
[26] P. Shvaiko and J. Euzenat. Ten challenges for ontology matching. In *ODBASE*, 2008.
[27] F. M. Suchanek, S. Abiteboul, and P. Senellart. PARIS: Probabilistic alignment of relations, instances, and schema. *PVLDB*, 5(3):157–168, 2011.
[28] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge. In *WWW*, 2007.
[29] W3C. RDF Primer (W3C Recommendation 2004-02-10).