# SiGMa: Simple Greedy Matching for Aligning Large Knowledge Bases

**Simon Lacoste-Julien**
Ecole Normale Superieure
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 which leverages both the structural information from the relationship graph as well as flexible similarity measures between entity properties in a greedy local search, thus making 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.

## 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* [1, 2, 3]. 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

[1] Such as MusicBrainz, IMDb, DBLP and UnitProt.

Euzenat [4] 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 [5], as another step towards the Semantic Web vision of interlinking different sources of knowledge which is exemplified by the Linking Open Data Initiative[2] [6]. Initial attempts to match IMDb entities to YAGO entities by naively exploiting string and neighborhood information failed, and so we designed SiGMa (Simple Greedy Matching), a scalable greedy iterative algorithm which 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 matching problem (by contrast with database record linkage approaches which make more independent decisions) as well as to focus on a simple approach which 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. 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 to develop and evaluate 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 95% precision in less than two hours (a 50x speed-up over [7]). 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 and evaluate it on benchmark and on real-world datasets in Section 4.

## 2 Aligning Large-Scale Knowledge Bases

### 2.1 Motivating example: YAGO and IMDb

Consider merging the information in the following two knowledge bases: YAGO, a large semantic knowledge base derived from English Wikipedia [5], WordNet [8] and GeoNames[3]; and IMDb, a large popular online database that stores information about movies.[4] 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 $\langle \text{John\_Travolta}, \text{ActedIn}, \text{Grease} \rangle$. The information from IMDb was originally available as several files which we merged into

[2] http://linkeddata.org/
[3] http://www.geonames.org/
[4] http://www.imdb.com/

a similar list of triples. We call these two databases *knowledge bases* to emphasize that we are not assuming a richer representation, such as RDFS [9], which would distinguish between classes and instances for example. In the language of ontology matching, our setup is the less studied *instance matching* problem, as pointed out by Castano et al. [10], for which the goal is to match concrete instantiations of concepts such as specific actors and specific movies rather than the general actor or movie class. YAGO comes with an RDFS representation, but not IMDb. We will 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. We will however assume that the relations between the two knowledge bases can be manually aligned, which is straightforward for these two knowledge bases (column 1 and 3 of Table 1a).

**Relationships vs. properties.** Given our assumption that the alignment is 1-1, it is important to distinguish between two types of objects which could be present in the list of triples: *entities* vs. *literals*. By our definition, the *entities* will be the only objects that we will try to align – they will be objects like specific actors or specific movies which have a clear identity. The *literals*, on the other hand, will correspond to a value related to an entity through a special kind of relationship that we will 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. For example, in the YAGO triple ⟨`m1`, `wasCreatedOnDate`, `1999-12-11`⟩, the object `1999-12-11` could be interpreted as a literal representing the value for the property `wasCreatedOnDate` for the entity `m1`. The corresponding property in our version of IMDb is `hasProductionYear` which has values only at the year granularity (`1999`). The 1-1 restriction would prevent us to align both `1999-12-11` and `1999-12-10` to `1999`. On the other hand, we can use these literals to define a similarity score between entities from the two knowledge bases (for example in this case, whether the year matches, or how close the dates are to each other). We will thus have two types of triples: entity-relationship-entity and entity-property-literal. We assume that the distinction between relationships and properties (which depends on the domain and the user's goals) is easy to make; for example, in the Freebase dataset that we also used in our experiments, the entities would have unique identifiers but not the literals. Figure 1 provides a concrete example of information presents in the two knowledge bases that we will keep re-using in this paper. We now define formally the problem that we address.

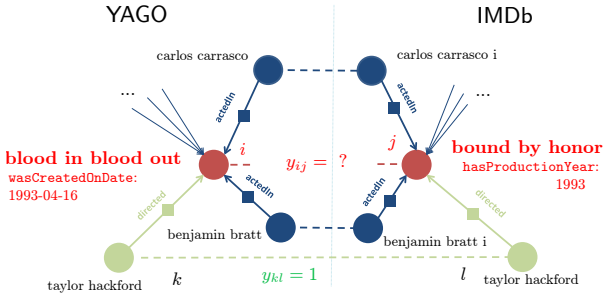


Figure 1: **Example of neighborhood to match in YAGO and IMDb.** Even though entities $i$ and $j$ have no words in common, the fact that several of their respective neighbors are matched together is a strong signal that $i$ and $j$ should be matched together. This is a real example from the dataset used 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).

**Definition:** A *knowledge base KB* is a tuple $(\mathscr{E}, \mathscr{L}, \mathscr{R}, \mathscr{P}, \mathscr{F}_R, \mathscr{F}_P)$ where $\mathscr{E}$, $\mathscr{L}$, $\mathscr{R}$ and $\mathscr{P}$ are sets of entities, literals, relationships and properties respectively; $\mathscr{F}_R \subseteq \mathscr{E} \times \mathscr{R} \times \mathscr{E}$ is a set of relationship-facts whereas $\mathscr{F}_P \subseteq \mathscr{E} \times \mathscr{P} \times \mathscr{L}$ 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 also present in $\mathscr{F}_R$ – that is, if $\langle e, r, e' \rangle$ is in $\mathscr{F}_R$, we also have $\langle e', r^{-1}, e \rangle$ in $\mathscr{F}_R$, effectively doubling the number of possible relations in the KB.[5]

**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 $\mathscr{E}_1$ to $\mathscr{E}_2$ which 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 $\mathscr{E}_1$).

[5]This allows us to look at only one standard direction of facts and cover all possibilities – see for example how it is used in the definition of `compatible-neigbhors`.

### 2.2 Possible approaches

Standard approaches for the ontology matching problem, such as RiMOM [11], could be used to align small knowledge bases. However, they do not scale to millions of entities as needed for our task given that they usually consider all pairs of entities, suffering 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 [12, 13], though these do not exploit the 1-1 matching combinatorial structure present in our task, which reduces their accuracy. More specifically, they usually make independent decisions for different entities using some kind of similarity function, rather than exploiting the competition between different assignments for entities. A notable exception is the work on *collective* entity resolution by Bhattacharya and Getoor [14], 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 [15], which has been formulated as a maximum weighted bipartite matching problem [16] (thus exploiting the 1-1 matching structure). It also has been formulated as a quadratic assignment problem in [17], which encourages neighbor entities in one graph to align to neighbor entities in the other graph, thus enabling alignment decisions to depend on each other — see the caption of Figure 1 for an example of this in our setup. The quadratic assignment formulation [18], which can be solved as an integer linear program, is NP-hard in general though, 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 will describe in Section 3.1. Finally, Suchanek et al. [7] recently proposed an ontology matching approach called PARIS that they have succeeded to apply on the alignment of YAGO to IMDb as well, though the scalability of their approach is not as clear, as we explain in the more detailed Related Work section of the longer version of this work [19]. We will 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 which globally scores the suitability of a particular matching $m$ for a pair of given KBs. This objective function will use two sources of information useful to choose 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 $m : \mathscr{E}_1 \to \mathscr{E}_2$ by a matrix $y$ with entries indexed by the entities in each KB, with $y_{ij} = 1$ if $m(i) = j$, meaning that $i \in \mathscr{E}_1$ is matched to $j \in \mathscr{E}_2$, and $y_{ij} = 0$ otherwise. The space of possible 1-1 partial mappings is thus represented by the set of binary matrices: $\mathscr{M} \doteq \{y \in \{0,1\}^{\mathscr{E}_1 \times \mathscr{E}_2} : \sum_l y_{il} \leq 1 \; \forall i \in \mathscr{E}_1$ and $\sum_k y_{kj} \leq 1 \; \forall j \in \mathscr{E}_2\}$. We define the following quadratic objective function which globally scores the suitability of a matching $y$:

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

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 $l$[6]. $\mathscr{N}_{ij}$ is a local neighborhood around $(i,j)$ that we define later and which will depend 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:

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

[6] In the rest of this paper, we will 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.

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 $y^*$ will have $\|y^*\|_1 = R$. Problem (2) is thus related to one of the variations of the quadratic assignment problems, a well-known NP-complete problem in operational research [18][7]. Even though one could approximate the solution to the combinatorial optimization (2) using a linear program relaxation (see Lacoste-Julien et al. [17]), the number of variables is quadratic in the number of entities, and so 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 $y_0$.
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 $\mathscr{S}_t$ 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, in the remainder of this section we will describe methods for choosing the similarity coefficients $s_{ij}$ and $w_{ij,kl}$ so that they guide the algorithm towards good matchings (Section 3.3), the choice of neighbors, $\mathscr{N}_{ij}$, the choice of a candidate set $\mathscr{S}_t$, and the stopping criterion, $R$. These choices influence both the speed and accuracy of the algorithm.

**Compatible-neighbors.** $\mathscr{N}_{ij}$ 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: $\mathscr{N}_{ij} = \texttt{compatible-neighbors}(i, j) \doteq \{(k,l)$: $\langle i, r, k\rangle$ is in $\mathscr{F}_{R1}$ and $\langle j, s, l\rangle$ is in $\mathscr{F}_{R2}$ and relationship $r$ is matched to $s\}$. Note that a property of this neighborhood is that $(k,l) \in \mathscr{N}_{ij}$ iff $(i, j) \in \mathscr{N}_{kl}$, 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* which is used to pick the next matched pair in the step 2 of the algorithm:

$$\texttt{score}(i, j; y) = (1-\alpha)s_{ij} + \alpha\,\delta g_{ij}(y) \qquad \text{where } \delta g_{ij}(y) \doteq \sum_{(k,l)\in\mathscr{N}_{ij}} y_{kl}\,(w_{ij,kl} + w_{kl,ij}). \tag{3}$$

**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 $\mathscr{S}_t$ 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) – it 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.

**Stopping criterion.** $R$ is implicitly chosen by the following heuristic: 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. In our experiments, a threshold of 0.25 is observed to correlate well with a point at which the F-measure stops increasing and the precision is significantly decreasing.

[7] See Appendix C of [19] for the traditional description of the quadratic assignment problem and its relationship to our problem.

### 3.2 Algorithm and implementation

**Algorithm 1:** SiGMa

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

We present the pseudo-code for SiGMa in Algorithm 1. More details on how we can implement it efficiently is given in Section 3.2 of [19]. We note that the `score` defined in (3) 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 $\mathscr{N}_{ij}$. We call the $\delta g_{ij}$ 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 explain in more details the choice of similarity measures for these components in Section 3.3. The initial match seed $m_0$ that we used in our experiments are pairs of entities which are the only ones to have the same string representation – that is, we do not include an exact matched pair when more than two entities have this same string representation, thereby increasing precision. The algorithm also takes an optional static list of candidates $\mathscr{S}_0$ which is built only once at the beginning and whose purpose is to increase exploration by using another source of information (which is not from the graph). 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.

### 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 (3). 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: $s_{ij} = (1-\beta)\texttt{string}(i,j) + \beta\texttt{prop}(i,j)$, 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, a metric often used in information retrieval and other data mining fields [20, 14]. To capture the information that some words are more informative than others, we use the IDF (inverse-document-frequency) weight for each word, a commonly used feature in information retrieval. The weight for word $v$ in $KB_o$ is $w_v^o \doteq \log_{10}|\mathscr{E}_o|/|E_v^o|$, where $E_v^o \doteq \{e \in \mathscr{E}_o : e \text{ has word } v \text{ in its string representation}\}$. Combining these elements, we get the following string similarity measure:

$$\texttt{string}(i,j) = \frac{\sum_{v \in (\mathscr{W}_i \cap \mathscr{W}_j)} (w_v^1 + w_v^2)}{\texttt{smoothing} \; + \sum_{v \in \mathscr{W}_i} w_v^1 + \sum_{v' \in \mathscr{W}_j} w_{v'}^2}, \tag{4}$$

where $\mathscr{W}_e$ is the set of words in the string representation of entity $e$ and `smoothing` is a scalar smoothing constant (we try different values in the experiments).

**Property similarity measure** We recall that we assume that the user provided 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, 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 [19] for completeness, but we note that it can make use of a similarity measure between literals such a normalized distance on numbers (for dates, years etc.) or a string-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 at 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 would possibly 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 $\gamma_i$ and $\gamma_j$ 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 $KB_2$). We use unit weights in our final experiments. A more detailed explanation for the graph contribution is given in Section 3.3.2 of [19].

## 4 Experiments

We made a prototype implementation of SiGMa in Python[8] and compared its performance on benchmark datasets as well as on large-scale knowledge bases. 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 *entities* correctly matched. 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 [19]. We found reasonable values for the parameters of SiGMa by exploring its performance on the YAGO to IMDb pair, and then kept them fixed for all the other experimental comparisons. 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 [19] for reproducibility. Additional experiments exploring the role of different parameter configurations for SiGMa as well as choosing the stopping threshold are given in section 4.5 of [19], but overall, the results were fairly robust to the parameter choice.

### 4.1 Experiment 1: Large-scale alignment

In this experiment, we test the performance of SiGMa on three pairs of large-scale KBs and compare it with PARIS [7]. The first dataset pair is YAGO-IMDb (the main motivating example for developing and testing SiGMa). The second pair is Freebase-IMDb, for which we could obtain a

[8] The code and datasets will be made available at http://mlg.eng.cam.ac.uk/slacoste/sigma.

| 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 | | |

(a) 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 |
| DBLP | 2.5M | 1.6M |
| Rexa | 12.6K | 14.7K |
| person11 | 500 | 1000 |
| person12 | 500 | 1000 |
| restaurant1 | 113 | 339 |
| restaurant2 | 752 | 2256 |

(b) Datasets statistics

Table 1: Information about datasets.

| Dataset | System | Prec | Rec | F | GT size | # pred. | Time |
|---|---|---|---|---|---|---|---|
| Freebase-IMdb | SiGMa | 99 | 95 | 97 | 255k | 366k | 90 min |
| | Exact-string | 99 | 70 | 82 | | 244k | 1 min |
| YAGO-IMDb | SiGMa | 98 | 93 | 95 | 54k | 188k | 50 min |
| | Exact-string | 99 | 57 | 72 | | 162k | 1 min |
| YAGO-IMDb_PARIS (new ground truth) | SiGMa | 98 | 96 | 97 | 57k | 237k | 70 min |
| | PARIS | 97 | 96 | 97 | | 702k | 3100 min |
| | Exact-string | 99 | 56 | 72 | | 202k | 1 min |
| YAGO-IMDb_PARIS (ground truth from [7]) | SiGMa | 98 | 84 | 91 | 11k | 237k | 70 min |
| | PARIS | 94 | 90 | 92 | | 702k | 3100 min |
| | Exact-string | 99 | 61 | 75 | | 202k | 1 min |

(a) Large-scale alignment results

| 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 | 98 | 96 | 97 | |
| | PARIS | 95 | 88 | 91 | |
| | Exact-string | 100 | 75 | 86 | |
| Rexa-DBLP | SiGMa | 97 | 90 | 94 | 1464 |
| | SiGMa-linear | 96 | 86 | 91 | |
| | Exact-string | 98 | 81 | 89 | |
| | RiMOM | 80 | 72 | 76 | |

(b) Benchmark comparison results

Table 2: (a) 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 [7]. The 'GT Size' column gives the number entities with ground truth information. Time is total running time, including loading the dataset (quoted from [7] for PARIS). (b) Results on the benchmark datasets for SiGMa, compared with PARIS [7] and RiMOM [11]. SiGMa-linear and Exact-string are also included on the interesting datasets as further comparison points.

sizable ground truth. We describe their construction in Section 4.2 of [19]. Finally, to facilitate the comparison of SiGMa with PARIS, the authors of PARIS gave us their own version of IMDb that we will refer from now on as IMDb_PARIS – this version has actually a richer structure in terms of properties. We present the aligned relationships and properties in Table 1a, and the number of unique entities and relationship-facts included in Table 1b. 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 2a 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 [7] 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 2a 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 as described in [19]. On both ground truth evaluations, SiGMa obtains a similar F-measure as PARIS, but in 50x less time.

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) having multiple very similar entities (e.g. mistaking the 'making of' of the movie vs. the movie itself); 3) pair of entities which 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.

### 4.2 Experiment 2: Benchmark comparisons

In this experiment, we test the performance of 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 [11, 21]. From the OAEI 2009 edition,[9] we use the Rexa-DBLP instance matching benchmark from the domain of scientific publications, where the goal is 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 1b.

We compare SiGMa with the best published results so far that we are aware of: PARIS [7] for the Person-Restaurants datasets (which compared favorably over ObjectCoref [21]); and RiMoM [11] for Rexa-DBPL. Table 2b 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 the Person and Restaurants datasets. Analyzing the errors made by SiGMa, we noticed that they were due to a violation of the assumption that each entity is unique in each KB: the same address is represented as different entities in Restaurant2, and SiGMa greedily matched the one which was not linked to another restaurant in Restaurant2, thus reducing the graph score for the correct match. SiGMa-linear couldn't suffer from this problem, and thus obtained a perfect matching.

The Rexa-DBLP dataset has a more interesting relationship structure which is not just 1-1: papers have multiple authors and authors have written multiple papers, enabling the fire propagation algorithm to explore more possibilities. However, it appears that a purely string based algorithm can already do quite well on this dataset — Exact-string obtains a 89% F-measure, already significantly improving the previously best published results (RiMOM at 76% F-measure), thus nuancing the difficulty of this standard benchmark. SiGMa-linear improves this to 91%, and finally using the graph structure helps to improve this to 94%. 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, which can be compared to RiMOM taking 36 hours on a 8-core server in 2009.

## 5 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 obtained a higher F-measure than the previously best published results on the OAEI benchmark datasets, and matched 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.

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

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

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

Some apparent limitations of SiGMa are a) that it 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. We could also envision using it to align social networks, as they have a rich graph information as well as multiple entity properties.

**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.

## References

[1] N. Choi, I.-Y. Song, and H. Han, "A survey on ontology mapping," *SIGMOD Rec.*, vol. 35, pp. 34–41, 2006.

[2] Y. Kalfoglou and M. Schorlemmer, "Ontology mapping: the state of the art," *Knowl. Eng. Rev.*, vol. 18, pp. 1–31, 2003.

[3] J. Euzenat and P. Shvaiko, *Ontology matching*. Springer-Verlag, 2007.

[4] P. Shvaiko and J. Euzenat, "Ten challenges for ontology matching," in *Proc. ODBASE*, 2008.

[5] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A Core of Semantic Knowledge," in *Proc. WWW*, 2007.

[6] C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee, "Linked data on the web (LDOW2008)," in *Proc. WWW*, 2008.

[7] F. M. Suchanek, S. Abiteboul, and P. Senellart, "PARIS: Probabilistic alignment of relations, instances, and schema," *PVLDB*, vol. 5, no. 3, pp. 157–168, 2011.

[8] C. Fellbaum, *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[9] W3C, "RDF Primer (W3C Recommendation 2004-02-10)."

[10] S. Castano, A. Ferrara, D. Lorusso, and S. Montanelli, "On the ontology instance matching problem," in *Proc. DEXA*, 2008.

[11] J. Li, J. Tang, Y. Li, and Q. Luo, "Rimom: A dynamic multistrategy ontology alignment framework," *IEEE Trans. on Knowl. and Data Eng.*, vol. 21, pp. 1218–1232, 2009.

[12] A. Arasu, C. Ré, and D. Suciu, "Large-scale deduplication with constraints using dedupalog," in *Proc. ICDE*, 2009.

[13] J. Gracia, M. d'Aquin, and E. Mena, "Large scale integration of senses for the semantic web," in *Proc. WWW*, 2009.

[14] I. Bhattacharya and L. Getoor, "Collective entity resolution in relational data," *ACM TKDD*, vol. 1, no. 1, 2007.

[15] F. J. Och and H. Ney, "A systematic comparison of various statistical alignment models," *Comput. Linguist.*, vol. 29, pp. 19–51, 2003.

[16] B. Taskar, S. Lacoste-Julien, and D. Klein, "A discriminative matching approach to word alignment," in *Proc. EMNLP*, 2005.

[17] S. Lacoste-Julien, B. Taskar, D. Klein, and M. I. Jordan, "Word alignment via quadratic assignment," in *Proc. HLT-NAACL*, 2006.

[18] E. L. Lawler, "The quadratic assignment problem," *Management Science*, vol. 9, no. 4, pp. 586–599, 1963.

[19] S. Lacoste-Julien *et al.* arXiv:1207.4525v1 [cs.AI], 2012.

[20] L. Hamers, Y. Hemeryck, *et al.*, "Similarity measures in scientometric research: The Jaccard index versus Salton's cosine formula," *Information Processing & Management*, vol. 25, no. 3, 1989.

[21] W. Hu, J. Chen, and Y. Qu, "A self-training approach for resolving object coreference on the semantic web," in *Proc. WWW*, 2011.