LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data

Axel-Cyrille Ngonga Ngomo
AKSW/BIS, Institut für Informatik
Universität Leipzig
Postfach 100920, 04009 Leipzig, Germany
ngonga@informatik.uni-leipzig.de

Sören Auer
AKSW/BIS, Institut für Informatik
Universität Leipzig
Postfach 100920, 04009 Leipzig, Germany
auer@informatik.uni-leipzig.de

ABSTRACT

The Linked Data paradigm has evolved into a powerful enabler for the transition from the document-oriented Web into the Semantic Web. While the amount of data published as Linked Data grows steadily and has surpassed 25 billion triples, less than 5% of these triples are links between knowledge bases. Link discovery frameworks provide the functionality necessary to discover missing links between knowledge bases in a semi-automatic fashion. Yet, the task of linking knowledge bases requires a significant amount of time, especially when it is carried out on large data sets. This paper presents and evaluates LIMES - a novel time-efficient approach for link discovery in metric spaces. Our approach utilizes the mathematical characteristics of metric spaces to compute estimates of the similarity between instances. These estimates are then used to filter out a large amount of those instance pairs that do not suffice the mapping conditions. Thus, LIMES can reduce the number of comparisons needed during the mapping process by several orders of magnitude. We present the mathematical foundation and the core algorithms employed in the implementation. We evaluate LIMES with synthetic data to elucidate its behavior on small and large data sets with different configurations and show that our approach can significantly reduce the time complexity of a mapping task. In addition, we compare the runtime of our framework with a state-of-the-art link discovery tool. We show that LIMES is more than 60 times faster when mapping large knowledge bases.

Categories and Subject Descriptors

H.m [**Information Systems**]: Miscellaneous

General Terms

Web of Data, Link Data, Link Discovery

Keywords

Linked Data, Web of Data, Link Discovery, Record Linkage, Duplicate Detection, Instance-Based Matching

1. INTRODUCTION

The core idea behind the Linked Data paradigm is to facilitate the transition from the document-oriented Web to the Semantic Web by extending the Web with a data commons consisting of interlinked data sources [20, 6]. While the number of triples in data sources increases steadily and has surpassed 25 billions[1], links still constitute less than 5% of the total number of triples available on the Linked Data Web. In addition, while the number of tools for publishing Linked Data on the Web grows steadily, there is a significant lack of time-efficient solutions for discovering links between these data sets. Yet, links between knowledge bases play a key role in important tasks such as cross-ontology question answering [4, 14], large-scale inferences [19, 16] and data integration [15, 3].

Over the last years, link discovery frameworks have been developed to facilitate the establishment of links between instances in different knowledge bases. In general, these frameworks aim at discovering instances in knowledge bases that should be connected via a *typed link* such as `owl:sameAs`. The discovery of such links is often dubbed *matching*. The input for a matching task consists of a source knowledge base S, a target knowledge base T, a distance measure[2] and a threshold θ. Consequently, two key challenges arise when trying to discover links between S and T: the computational complexity of the matching task *per se* and the selection of an appropriate distance measure.

The first challenge is intrinsically related with the link discovery process. To carry out a matching task, the distance measure as defined by the user is usually applied to the value of some properties of instances from S and T so as to detect instances that should be linked. Instances whose distance is lower or equal to a given threshold are considered to be candidates for linkage. The a-priori complexity of a matching task is proportional to the product of the number of instances in the source and target data source, an unpractical proposition as soon as the source and target knowledge bases become large. For example, discovering duplicate cities in DBpedia [1] alone would necessitate approximately 0.15×10^9 distance computations. Hence, the provision of time-efficient approaches for the reduction of the time complexity of link discovery is a key challenge of the Linked Data.

[1] http://www4.wiwiss.fu-berlin.de/lodcloud/

[2] Note that a similarity measure can be used instead of a distance measure.

The second challenge of the link discovery process lies in the selection of an appropriate distance measure. This selection is usually carried out manually, in most cases simply by guessing. Yet, the choice of a suitable distance measure decides upon whether satisfactory links can be discovered. The large spectrum of measures available in literature underline the difficulty of the task of choosing the right function manually[3]. Supporting the user during the process of finding the appropriate distance measure for each mapping task is a problem that still needs to be addressed by the Linked Data community. Methods such as supervised and active learning can be used to guide the user in need of mapping to a suitable distance measure for his matching task. Yet, these methods could not be used so far because of the time complexity of link discovery, thus even further amplifying the need for time-efficient methods.

In this paper, we present LIMES (Link Discovery Framework for metric spaces) - a time-efficient approach for the discovery of links between Link Data sources. LIMES addresses the scalability problem of link discovery by utilizing the *triangle inequality* in metric spaces to compute pessimistic estimates of instance similarities. Based on these approximations, LIMES can filter out a large number of instance pairs that cannot suffice the matching condition set by the user. The real similarities of the remaining instances pairs are then computed and the matching instances are returned. We show that LIMES requires a significantly smaller number of *comparisons* than brute force approaches by using synthetic data. In addition, we show that our approach is superior to state-of-the-art link discovery frameworks by comparing their runtime in real-world use cases.

The contributions of this work are as follows:

- We present the lossless and time-efficient approach for the large-scale matching of instances in metric spaces dubbed LIMES. Especially, we introduce a general approach for the time-efficient discovery of links between Linked Data sources.
- We present two novel algorithms for the efficient approximation of distances within metric spaces based on the triangle inequality. These algorithms make use of pessimistic estimates of distances to reduce the number of comparisons necessary to complete a matching task.
- We evaluate LIMES on synthetic data by using the number of comparisons necessary to complete the given matching task. Furthermore, we evaluate our approach on real data against the SILK framework with respect to the runtime of both frameworks on the same hardware. We show that our framework outperforms SILK significantly and is more than 60 times faster when comparing large knowledge bases.

The remainder of this paper is structured as follows: after reviewing related work in Section 2 we develop the mathematical framework underlying LIMES in Section 3. We present the LIMES approach in Section 4 and report on the results of an experimental evaluation in Section 5. We conclude with a discussion and an outlook on future work in Section 6.

[3]The SimMetrics project (`http://simmetrics.sf.net`) provides an overview of strings similarity measures.

2. RELATED WORK

Using the triangle inequality for improving the runtime of algorithms is not a novel idea. This inequality has been used for tasks such as data clustering [8], spatial matching [23] and query processing [24]. Yet, to the best of our knowledge it has never been used previously for the discovery of links on the Web of Data.

Current frameworks for link discovery can be subdivided into two categories: *domain-specific* and *universal* frameworks. Domain-specific link discovery frameworks aim at discovering links between knowledge bases from a particular domain. For example, the RKB knowledge base (RKB-CRS) uses Universal Resource Identifier (URI) lists to compute links between universities and conferences [11]. For each new mapping task, a new program has to be written, wherein the source, target and mapping function must be declared. Another domain-specific tool is GNAT [17], which was developed for the music domain. GNAT uses similarity propagation on audio fingerprinting to discover links between music data sets.

Universal link discovery frameworks are designed to carry out mapping tasks independently from the domain of the source and target knowledge bases. For example, RDF-AI [18] implements a five-step approach that comprises the preprocessing, matching, fusion, interlink and post-processing of data sets. These modules can be configured by means of XML-files. In contrast to LIMES, RDF-AI does not comprise means for querying distributed data sets via SPARQL[4]. Like LIMES, SILK [20] is time-optimized for link discovery. Instead of utilizing the characteristics of metric spaces, SILK uses rough index pre-matching to reach a quasi-linear time-complexity. The drawback of the pre-matching approach is that the recall of this framework is not guaranteed to be 1. SILK allows the manual configuration of data blocks to minimize the runtime of the matching process. It can be configured by using the SILK-Link Specification Language, which is based on XML.

The task of discovering links between knowledge bases is closely related with record linkage [22, 9] and de-duplication [7]. The database community has produced a vast amount of literature on efficient algorithms for solving these problems. Different blocking techniques such as standard blocking, sorted-neighborhood, bigram indexing, canopy clustering and adaptive blocking [2, 5, 12] have been developed to address the problem of the quadratic time complexity of brute force comparison methods. The idea is to filter out obvious non-matches efficiently before executing the more detailed and time-consuming comparisons.

Several record linkage approaches have also used the triangle inequality to reduce the number of comparisons necessary to carry out a mapping task. For example, Ochard's algorithm has been used to develop anytime algorithms for

[4]`http://www.w3.org/TR/rdf-sparql-query/`

record linkage which trade the quality of the results for execution time [25]. This algorithm consists of two steps: First, all distances between all nodes of the target database are computed. Then the distances are used to reduce the number of distance computations necessary to discover similar records. In the worst case, the time complexity of Orchard's algorithm is $O((|T| + |S|)|T|)$.

The difference between the approaches described above and our approach is that LIMES uses the triangle inequality to portion the metric space. Each of these portions of the space is then represented by an exemplar [10] that allows to compute an accurate approximation of the distance between each instance in this region and others instances. By these means, we can discovery links between Linked Data sources efficiently.

3. MATHEMATICAL FRAMEWORK

In this section, we present the mathematical principles underlying the LIMES framework. We start by epitomizing the characteristics of metric spaces. Then, we present the formal definition of a matching task within metric spaces. We use this definition to infer upper and lower boundary conditions for distances based on the triangle inequality. Furthermore, we show how these boundary conditions can be used to reduce the number of comparisons necessary to complete a mapping.

3.1 Preliminaries

In the remainder of this paper, we use the following notations:

1. A is an affine space,
2. m, m_1, m_2, m_3 symbolize metrics on A,
3. x, y and z represent points from A and
4. α, β, γ and δ are scalars, i.e., elements of $\mathbb{R}$.

DEFINITION 1 (METRIC SPACE). *A* metric space *is a pair (A, m) such that A is a set of points and $m : A \times A \to \mathbb{R}$ is a function such that for all x, y and $z \in A$*

1. $m(x, y) \geq 0$ (M_1) *(non-negativity),*
2. $m(x, y) = 0 \Leftrightarrow x = y$ (M_2) *(identity of indiscernibles),*
3. $m(x, y) = m(y, x)$ (M_3) *(symmetry) and*
4. $m(x, z) \leq m(x, y) + m(y, z)$ (M_4) *(triangle inequality).*

An example of a string similarity measure representing a metric is the Levenshtein distance [13]. However, some popular measures such as JaroWinkler [21] do not satisfy the triangle inequality and are consequently not metrics.

DEFINITION 2 (MATCHING TASK). *Given two sets S (source) and T (target) of instances, a metric m and a threshold $\theta \in [0, \infty[$, the goal of a matching task is to compute the set M of triples (i.e., the matching) $(s, t, m(s, t))$ of all instances $s \in S$ and $t \in T$ such that $m(s, t) \leq \theta$.*

We call each computation of the distance $m(s, t)$ a *comparison*. The time complexity of a mapping task can be measured by the number of comparisons necessary to complete this task. A-priori, the completion of a matching task requires $O(|S||T|)$ comparisons. In this paper, we show how the number of comparisons necessary to map two knowledge bases can be reduced significantly by using the mathematical characteristics of metric spaces. For this purpose, we make particularly use of the fourth characteristic of metrics, the triangle inequality (TI).

3.2 Distance Approximation Based on the Triangle Inequality

The rationale behind our framework is to make use of the boundary conditions entailed by the TI to reduce the number of comparisons (and thus the time complexity) necessary to complete a matching task. In this section, we first present boundary conditions for distances entailed by the TI. Then, we use them to infer an approach to reduce the number of comparisons necessary to complete a matching task.

Given a metric space (A, m) and three points x, y and z in A, the TI entails that

$$m(x, y) \leq m(x, z) + m(z, y). \quad (1)$$

Without restriction of generality, the TI also entails that

$$m(x, z) \leq m(x, y) + m(y, z), \quad (2)$$

thus leading to the following boundary conditions in metric spaces:

$$m(x, y) - m(y, z) \leq m(x, z) \leq m(x, y) + m(y, z). \quad (3)$$

Inequality 3 has two major implications. The first is that the distance from a point x to any point z in a metric space can be approximated when knowing the distance from x to a reference point y and the distance from the reference point y to z. We call such a reference point an *exemplar* (following [10]). The role of an exemplar is to be used as a sample of a portion of the metric space A. Given an input point x, knowing the distance from x to an exemplar y allows to compute lower and upper bounds of the distance from x to any other point z at a known distance from y. An example of such an approximation is shown in Figure 1. In this figure, all the points on the circle are subject to the same distance approximation. The distance from x to z is close to the lower bound of inequality 3, while the distance from x to z' is close to the upper bound of the same inequality.

The second implication of inequality 3 is that the real distance from x to z can only be smaller than θ if the lower bound of the approximation of the distance from x to z via *any exemplar* y is also smaller than θ. Thus, if the lower bound of the approximation of the distance $m(x, z)$ is larger than θ, then $m(x, z)$ itself must be larger than θ. Formally,

$$m(x, y) - m(y, z) > \theta \Rightarrow m(x, z) > \theta. \quad (4)$$

Supposing that all distances from instances $t \in T$ to exemplars are known, reducing the number of comparisons simply consists of using inequality 4 to compute an approximation of the distance from all $s \in S$ to all $t \in T$ and computing the real distance only for the (s, t) pairs for which the first

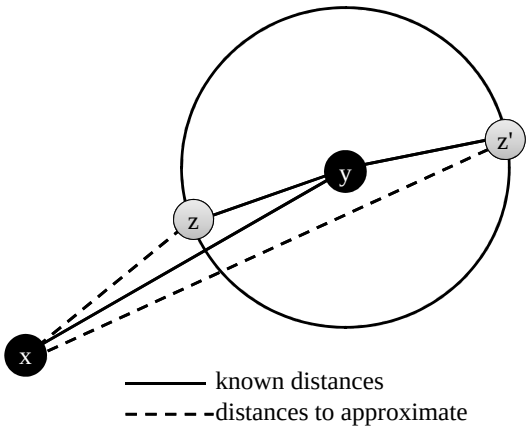


Figure 1: Approximation of distances via exemplars. The lower bound of the distance from x to z can be approximated by $m(x, y) - m(y, z)$.

term of inequality 4 does not hold. This is the core of the approach implemented by LIMES.

4. THE LIMES FRAMEWORK

In this section, we present the LIMES framework in more detail. First, we give an overview of the workflow it implements. Thereafter, we present the two core algorithms underlying our framework. Finally, we present the architecture of the current implementation.

4.1 Overview

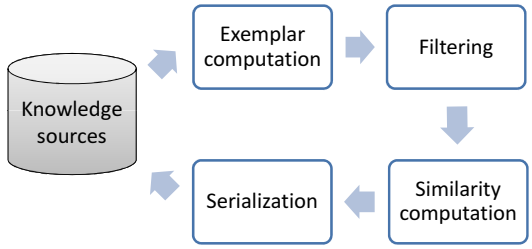


Figure 2: General Workflow of LIMES

The general workflow implemented by the LIMES framework comprises four steps (as depicted in Figure 2). Given the source S, the target T and the threshold θ, LIMES first computes a set E exemplars for T (step 1). This process is concluded by matching each point $t \in T$ to the exemplar closest to it. In step 2 and 3, the matching *per se* is carried out. For each $s \in S$ and each $e \in E$, the distance $m(s, e)$ is computed. This distance is used to approximate the distance from s to every $t \in T$ (step 2). We call this step *filtering*. The filtering step implements the central innovation of our approach. The main advantage here is that since pessimistic estimates are used, it is guaranteed to lead to exactly the same matching as a brute force approach while at the same time reducing the number of comparisons significantly. After the filtering, the real distance between the remaining $s \in S$ and the $t \in T$ for which the first term of inequality 4 did not hold are computed (step 3). Finally, the matchings $(s, t, m(s, t))$ such that $m(s, t) \leq \theta$ are serialized, i.e., written in a user-defined output stream according to a user-specified format, e.g. in an NTriples file[5] (step 4).

4.2 Computation of Exemplars

The role of exemplars is to represent a portion of a metric space. Accordingly, the best distribution of exemplars should achieve a homogeneous portioning of this metric space. The core idea underlying the computation of exemplars in LIMES is to select a set of exemplars in the metric space underlying the matching task in such a way that they are distributed in a uniform way in the metric space. One way of achieving this goal is by ensuring that the exemplars are as dissimilar as possible. The approach we use to generate exemplars with a maximal dissimilarity is shown in Algorithm 1.

Data: Number of exemplars n, target knowledge base T
Result: Set E of exemplars and their matching to the instances in T
1. Pick random point $e_1 \in T$;
2. Set $E = E \cup \{e_1\}$, $\eta = e_1$;
3. Compute the distance from e_1 to all $t \in T$;
while $|E| < n$ **do**
  4. Get a random point e' such that $e' \in argmax_t \sum_{t \in T} \sum_{e \in E} m(t, e)$;
  5. $E = E \cup \{e'\}$; 6. Compute the distance from e' to all $t \in T$;
end
7. Map each point in $t \in T$ to one of the exemplars $e \in E$ such that $m(t, e)$ is minimal;
8. Return E;

Algorithm 1: Computation of Exemplars

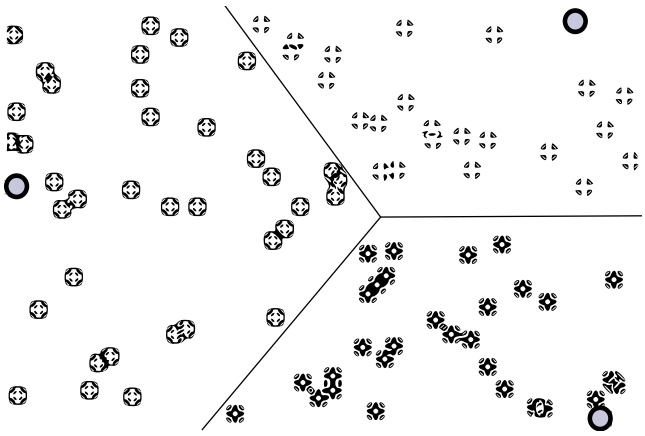


Figure 3: Mapping of points to three exemplars in a metric space. The exemplars are displayed as gray disks.

Let n be the desired number of exemplars and E the set of all exemplars. In step 1 and 2, we initialize E by picking a random point e_1 in the metric space (T, m) and setting $E = \{e_1\}$. Then, we compute the similarity from the exemplar e_1 to every other point in T (step 3). As long as the size of E has not reached n, we iterate steps 4 to 6: In step 4, we

[5]http://www.w3.org/2001/sw/RDFCore/ntriples/

pick a point $e' \in T$ such that the sum of the distances from e' to the exemplars $e \in E$ is maximal (there can be many of these points). This point is chosen as new exemplar and added to E (step 5). Then, we compute the distance from e' to all other points in T (step 6).

Once E has reached the size n, we terminate the iteration. Finally, we map each point to the exemplar to which it is most similar (step 7) and terminate (step 8). This algorithm has a constant time complexity of $O(|E||T|)$.

An example of the results of the exemplar computation algorithm ($|E| = 3$) is shown in Figure 3. The initial exemplar was the leftmost exemplar in the figure.

4.3 Matching Based on Exemplars

The instances associated with an exemplar $e \in E$ in step 7 of Algorithm 1 are stored in a list L_e sorted in descending order with respect to their distance to e. Let $\lambda_1^e ... \lambda_m^e$ be the elements of the list L_e. The goal of matching an instance s from a source knowledge base to a target knowledge base w.r.t. a metric m is to find all instances t of the target knowledge source such that $m(s,t) \leq \theta$, where θ is a given threshold. The matching algorithm based on exemplars is shown in Algorithm 2.

Data: Set of exemplars E, point s, metric m, threshold θ
Result: matching M for s
1. $M = \emptyset$;
for $e \in |E|$ **do**
  if $m(s,e) \leq \theta$ **then**
    2. $M = M \cup \{e\}$;
  end
  for $i = 1...|L_e|$ **do**
    if $(m(s,e) - m(e,\lambda_i^e)) \leq \theta$ **then**
      if $m(s,\lambda_i^e) \leq \theta$ **then**
        3. $M = M \cup \{\lambda_i^e\}$
      end
    else
      break;
    end
  end
end
4. return M;

Algorithm 2: Comparison algorithm

We only carry out a comparison when the approximation of the distance is less than the threshold. We terminate the similarity computation for an exemplar e as soon as the first λ^e is found such that the lower bound of the distance is larger than θ. This break can be carried out because the list L_e is sorted, i.e., if $m(s,e) - m(e,\lambda_i^e) > \theta$, then the same inequality holds for all λ_j^e with $j > i$. In the worst case, our matching algorithm has the time complexity $O(|S||T|)$, leading to a total worst time complexity of $O((|E|+|S|)|T|)$, which is larger than that of brute force approaches. However, as our evaluation with both synthetic and real data shows, a correct parameterization of LIMES leads to significantly smaller numbers of comparisons and runtimes.

4.4 Implementation

The LIMES framework consists of seven main modules (see Figures 4 and 6) of which each can be extended to accommodate new or improved functionality[6].

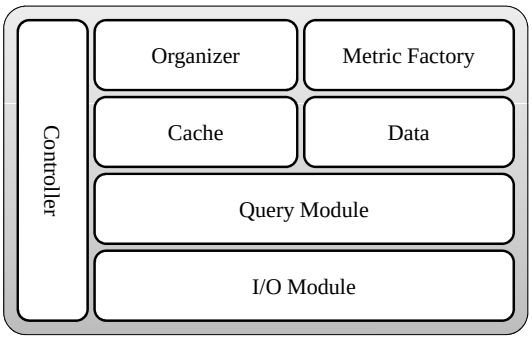


Figure 4: Architecture of LIMES

The central modules of LIMES are the *controller module*, which coordinates the matching process and the *data module*, which contains all the classes necessary to store data. The matching process is carried out as follows: First, the *controller* calls the *I/O-module*, which reads the configuration file and extracts all the information necessary to carry out the comparison of instances, including the URL of the SPARQL-endpoints of the knowledge bases, the restrictions on the instances to map (e.g., their type), the expression of the metric to be used and the threshold to be used. An example of such a configuration file is given in Figure 5. Note, that the LIMES configuration takes similarity (and not distance) thresholds as input. This is simply due to the fact that similarity are more intuitive for end-users than distances. Consequently, the experiments reported in the subsequent section also relate to similarity thresholds. Given that the configuration file is valid w.r.t. the LIMES Specification Language (LSL)[7], the *query module* is called. This module uses the configuration for the target and source knowledge bases to retrieve instances and properties from the SPARQL-endpoints of the source and target knowledge bases that adhere to the restrictions specified in the configuration file. The query module writes its output into a *cache*, which can either be a file (for large number of instances) or main memory. Once all instances have been stored in the cache, the controller calls the *organizer module*. This module carries out two tasks: first, it computes the exemplars of the source knowledge base. Then, it uses the exemplars to compute the matchings from the source to the target knowledge base. Finally, the *I/O-module* is called to serialize the results.

4.5 Supported Operations

Entity matching algorithms support a range of operations that allow users to combine a set of pre-defined measures to novel user-defined ones. The most common of these operations include the *linear combination* of metrics as well as building their *maximum*, *minimum* and *average*. The LIMES framework only supports operations that combine

[6]The framework is open-source and can be found at `http://limes.sf.net`.
[7]The DTD for the LSL can be found in the LIMES distribution along with a user manual at: `http://limes.sf.net`.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE LIMES SYSTEM "limes.dtd">
<LIMES>
  <PREFIX> <NAMESPACE>http://www.w3.org/1999/02/22-rdf-syntax-ns#</NAMESPACE> <LABEL>rdf</LABEL>      </PREFIX>
  <PREFIX> <NAMESPACE>http://www.w3.org/2002/07/owl#</NAMESPACE>               <LABEL>owl</LABEL>      </PREFIX>
  <PREFIX> <NAMESPACE>http://data.linkedct.org/resource/linkedct/</NAMESPACE> <LABEL>linkedct</LABEL> </PREFIX>
  <PREFIX> <NAMESPACE>http://bio2rdf.org/ns/mesh#</NAMESPACE>                  <LABEL>meshr</LABEL>    </PREFIX>
  <SOURCE>     <ID> linkedct </ID>
               <ENDPOINT> http://data.linkedct.org/sparql </ENDPOINT>
               <VAR> ?x </VAR>
               <PAGESIZE> 5000 </PAGESIZE>
               <RESTRICTION> ?x rdf:type linkedct:condition </RESTRICTION>
               <PROPERTY> linkedct:condition_name </PROPERTY>                  </SOURCE>
  <TARGET>     <ID> mesh </ID>
               <ENDPOINT> http://mesh.bio2rdf.org/sparql </ENDPOINT>
               <VAR> ?y </VAR>
               <PAGESIZE> 5000 </PAGESIZE>
               <RESTRICTION> ?y rdf:type meshr:Concept </RESTRICTION>
               <PROPERTY> dc:title </PROPERTY>                                 </TARGET>
  <METRIC>     levenshtein(x.linkedct:condition_name, y.dc:title)              </METRIC>
  <EXEMPLARS>  70 </EXEMPLARS>
  <ACCEPTANCE> <THRESHOLD> 0.9 </THRESHOLD> <RELATION> owl:sameAs </RELATION>
               <FILE>diseases\_accepted.nt</FILE>                              </ACCEPTANCE>
  <REVIEW>     <THRESHOLD> 0.8 </THRESHOLD> <RELATION> owl:sameAs </RELATION>
               <FILE>diseases\_review.nt</FILE>                                </REVIEW>
</LIMES>
```

Figure 5: Example of a LIMES configuration file. The corresponding link discovery task maps diseases from LinkedCT and MESH by measuring the similarity of their labels.

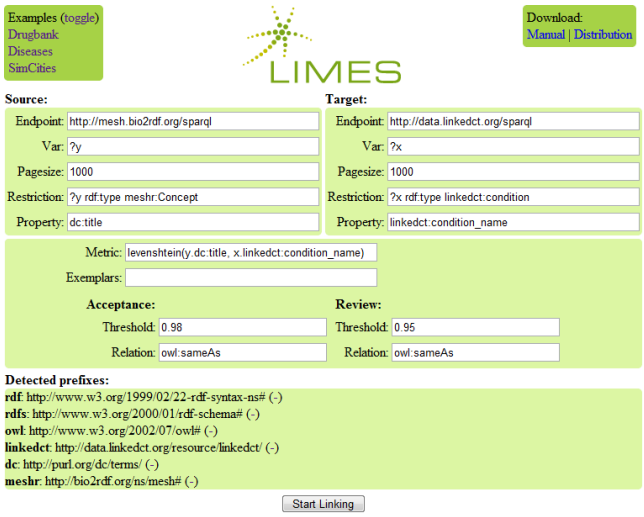


Figure 6: The LIMES Web dashboard. A publicly accessible version is available at: `http://limes.sf.net`.

metrics into new metrics. In the following, we show which of the standard operations typically used in matching frameworks can be used in LIMES.

Linear Combination. Let m_3 be the linear combination of two metrics m_1 and m_2 given by $\alpha m_1 + \beta m_2$. Such a combination can only be a metric if it obeys M_1 to M_4. One can easily show that obeying these conditions leads to $(\alpha \geq 0 \wedge \beta > 0) \vee (\alpha > 0 \wedge \beta \geq 0)$. Thus, the linear combination of metrics with positive scalars is a metric. In particular, the average function is a special case of a linear combination where $\alpha = \beta = 0.5$ and thus a metric.

Maximum. Let max be the maximum function and $m_3 = \max(m_1, m_2)$. Given two metrics m_1 and m_2 in the metric space A and three points x, y and z from A. One can easily show that m_3 fulfills (M_1) to (M_3):

1. $m_1(x, y) \geq 0 \wedge m_2(x, y) \geq 0 \Rightarrow m_3(x, y) \geq 0$ (M_1),
2. $(m_1(x, y) = 0 \Leftrightarrow x = y) \wedge (m_2(x, y) = 0 \Leftrightarrow x = y) \Rightarrow (m_3(x, y) = 0 \Leftrightarrow x = y)$ (M_2),
3. $m_1(x, y) = m_1(y, x) \wedge m_2(x, y) = m_2(y, x) \Rightarrow m_3(x, y) = m_3(y, x)$ (M_3).

To show that m_3 also fulfills (M_4), we use the following:

$$\forall x, y, z \in A : m_1(x, z) \leq m_1(x, y) + m_1(y, z) \tag{5}$$

and

$$\forall x, y, z \in A : m_2(x, z) \leq m_2(x, y) + m_2(y, z). \tag{6}$$

Thus,

$$\begin{aligned} \max(m_1(x, z), m_2(x, z)) \leq \\ \max(m_1(x, y) + m_1(y, z), m_2(x, y) + m_2(y, z)). \end{aligned} \tag{7}$$

Given that

$$\max(\alpha + \beta, \delta + \gamma) \leq \max(\alpha, \delta) + \max(\beta, \gamma) \tag{8}$$

we can infer that

$$\forall x, y, z \in A : m_3(x, z) \leq m_3(x, y) + m_3(y, z). \tag{9}$$

Thus, the combination of two metrics with max is also a metric.

Minimum. The minimum $m_3(x, y) = \min(m_1, m_2)$ of two metrics m_1 and m_2 is not a metric. For example, given $m_1(x, z) = 2$, $m_1(x, y) = 4$, $m_1(y, z) = 0$, $m_2(x, z) = 2$, $m_2(x, y) = 0$ and $m_1(y, z) = 4$, we get $m_3(x, z) = 2$ but $m_3(x, y) + m_3(y, z) = 0$, which does not satisfy the TI.

As a result, we can combine string metrics by using the summation, multiplication with positive scalars and the maximum. Negative scalars lead to non-positive values, thus violating (M_1). Furthermore, using the minimum operator leads to measures that violate the TI (M_4).

5. EVALUATION

In this section, we elucidate the following four central evaluation questions:

Q_1: *What is the best number of exemplars?* The core of the question is whether there is an obvious optimal value for $|E|$. A small number of exemplars leads to a smaller overhead for the computation of the exemplars per se. Yet, a small number of exemplars also leads to a coarser approximation of the distances and might consequently lead to a larger number of comparisons. On the other hand, a large number of exemplars leads to a considerable overhead for the exemplar computation. However, it also results in more fine-grained approximations of distances and thus less comparisons during the matching task per se.

Q_2: *What is the relation between the threshold θ and the total number of comparisons?* The formalism underlying LIMES allows us to infer that the higher the value of θ, the smaller the number of required comparisons. The question here is whether one can predict the number of comparisons that LIMES will necessitate based on θ, $|S|$ and $|T|$.

Q_3: *Does the assignment of S and T matter?* Here, the question is whether the matching should be carried out by using the largest knowledge base as source or target. Given that the metrics are symmetric, the assignment of knowledge bases to source and target knowledge base would not change the value of the similarity. Consequently, the final matching will not change either.

Q_4: *How does LIMES compare to other approaches?* This is the key question for end-users.

To answer Q_1 to Q_3, we performed an evaluation on synthetic data as described in the subsequent section. Q_4 was elucidated by comparing the runtimes of LIMES and SILK on three different real-world matching tasks.

5.1 Evaluation with Synthetic Data

The general experimental setup for the evaluation on synthetic data was as follows: The source and target knowledge bases were filled with random strings having a maximal length of 10 characters. We used the Levenshtein metric [13] to measure string similarity. Each of the matching tasks was carried out five times and we report average values in the following.

Q_1 and Q_2. To address the first two questions, we considered six matching tasks on knowledge bases of sizes $1,000$, $2,000$, $3,000$, $5,000$, $7,500$ and $10,000$ respectively. We varied the thresholds between 0.95 and 0.75 and the number of exemplars between 10 and 300. We measured the average number of comparisons necessary to carry out each of the matching tasks. The results of this series of computations are shown in Figure 7.

Two main conclusions can be inferred from the results. First, the results clearly indicate that the best number of exemplars diminishes when the similarity threshold θ is increased. In general, the best value for $|E|$ lies around $\sqrt{|T|}$ for $\theta \geq 0.9$, which answers Q_1. The relation between θ and $|E|$ is a direct cause of our approach being based on the triangle inequality. Given a high threshold, even a rough approximation of the distances is sufficient to rule out a significant number of target instances as being similar to a source instance. However, a low threshold demands a high number of exemplars to be able to rule out a significant number of target instances.

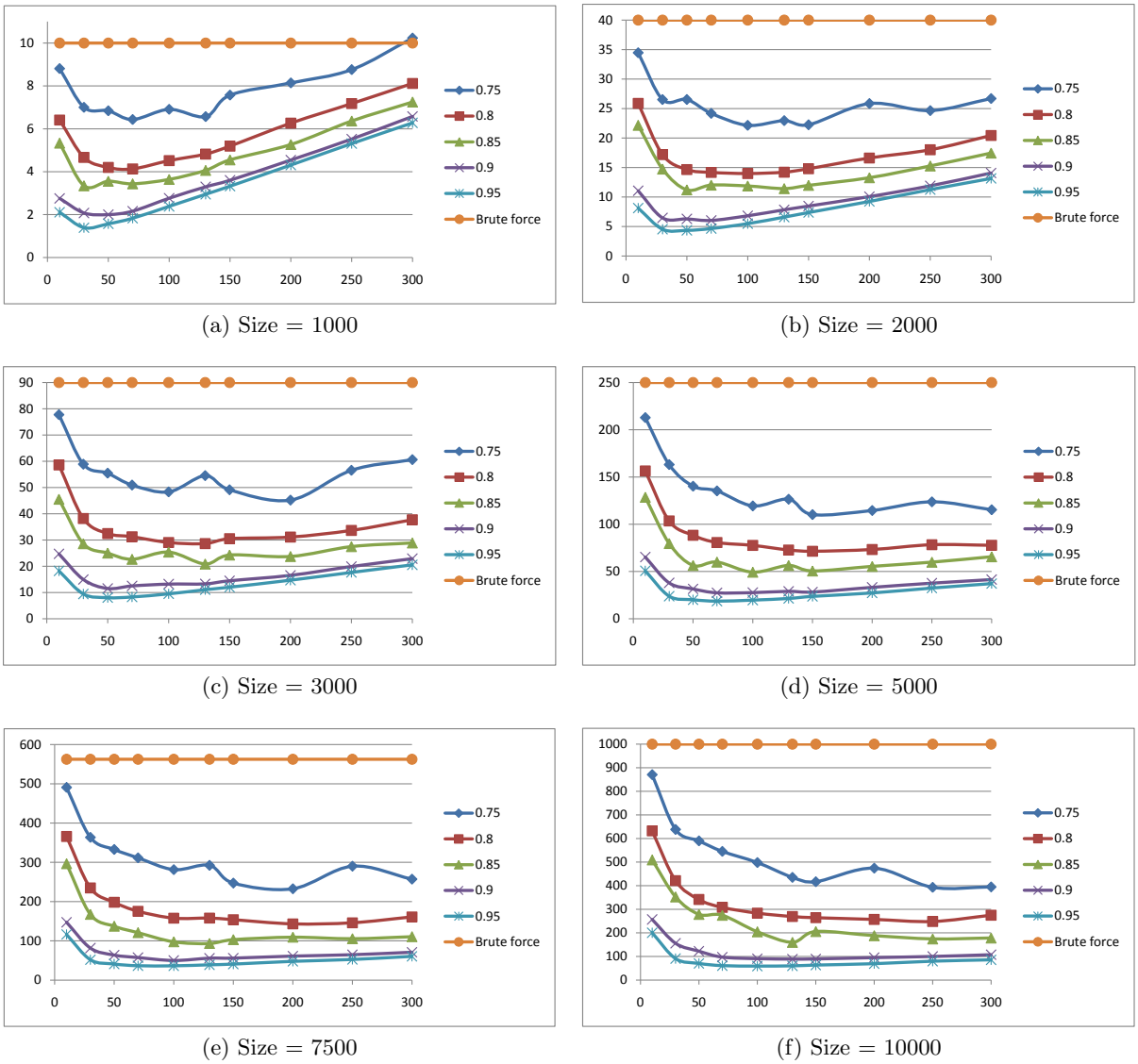
An analysis of the results displayed in Figure 7 also allows to answer Q_2. The higher the value of θ, the smaller the number of comparisons. This is due to the stricter filtering that results from the higher threshold and consequently leads to a smaller number of required comparisons. An important observation is that, the larger the size of the knowledge bases S and T, the higher the speedup obtained by using the LIMES approach. For example, while LIMES necessitates approximately 7 times less comparisons than a brute force approach for the knowledge bases of size $1,000$ and $\theta = 0.95$ in the best case, it requires approximately 17 times less comparisons for knowledge bases of size $10,000$ with the same threshold settings. It is also important to note that, even when 33% of the instances in T are used as exemplars (see Figure 7(a)), LIMES necessitates only 2% more comparisons than the brute force approach when θ is set to the low value of 0.75. In all other configurations, LIMES requires significantly less computations than brute force.

Q_3. To address Q_3, we measured the average number of comparisons required to map synthetic knowledge bases of sizes between 1,000 and 10,000 in all possible combinations of sizes for S and T. For this experiment, the number of exemplars was set to $\sqrt{|T|}$. θ was set to 0.9. The results of this experiment are summarized in Table 1.

Overall, the experiment shows that whether source or target knowledge base is larger does not affect the number of comparisons significantly. It appears that the results are slightly better when $|T| \leq |S|$. Yet, the difference between the number of comparisons lies below 5% in most cases and is thus not significant. Therefore, the link discovery can always be carried out by simply following the specification of the user with respect to which endpoint is source resp. target of the matching.

5.2 Evaluation with Real Data

To answer the question Q_4, we evaluated the performance of LIMES on real data by comparing its runtime with that of the time-optimized link discovery framework SILK. The total runtime of each of the frameworks consisted of three parts: (1) the time necessary to fetch data from the source



(a) Size = 1000

(b) Size = 2000

(c) Size = 3000

(d) Size = 5000

(e) Size = 7500

(f) Size = 10000

Figure 7: Comparisons required by LIMES for different numbers of exemplars on knowledge bases of different sizes. The x-axis shows the number of exemplars, the y-axis the number of comparisons in multiples of 10^5.

and target knowledge base, (2) the time necessary to compare the instances and (3) the time necessary to write the results. To ensure an objective comparison of the runtimes, we only considered the time necessary to carry out the comparisons in our evaluation, as the two other components of the total runtime depend on external factors such as network latency and the fragmentation of the hard drive. Each of the time measurements was carried out three times and only the best runtime was considered. Every time measurement experiment was carried out as a single thread on a 32-bit system with a 2.5GHz Intel Core Duo CPU and 4GB RAM. For our experiments, we used version 0.3.2 of LIMES and version 2.0 of SILK. The number of exemplars for LIMES was set to $\sqrt{|T|}$.

The experiments on real data were carried out in three different settings as shown in Table 2. The goal of the first experiment, named Drugbank, was to map drugs in DBpedia[8] and Drugbank[9] by comparing their labels. The goal

Table 2: Overview of runtime experiments. |S| is the size of the source knowledge base, |T| is the size of the target knowledge base and |E| is the number of exemplars used by LIMES during the experiment.

	Drugbank	**SimCities**	**Diseases**
|S|	4,346	12,701	23,618
|T|	4,772	12,701	5,000
|E|	69	112	70
Source	DBpedia	DBpedia	MESH
Target	Drugbank	DBpedia	LinkedCT

[8]http://dbpedia.org/sparql
[9]http://www4.wiwiss.fu-berlin.de/drugbank/sparql

Table 1: Average number of comparisons (in millions) for matching knowledge bases of different sizes. The columns are the size of the source knowledge base, while the row are the size of the target knowledge base.

	1000	**2000**	**3000**	**4000**	**5000**	**6000**	**7000**	**8000**	**9000**	**10000**
1000	0.20	0.37	0.53	0.69	0.88	1.04	1.14	1.40	1.58	1.67
2000	0.36	0.64	0.88	1.24	1.37	1.63	1.97	2.25	2.50	2.70
3000	0.51	0.86	1.17	1.57	2.00	2.09	2.69	2.91	3.35	3.58
4000	0.70	1.11	1.59	2.00	2.45	2.88	3.10	3.61	3.94	4.50
5000	0.85	1.36	1.87	2.28	2.81	3.39	3.91	4.20	4.84	5.54
6000	1.02	1.60	2.14	2.81	3.29	3.93	4.44	4.96	5.39	6.08
7000	1.22	1.86	2.58	3.15	3.66	4.35	5.11	5.69	6.44	6.62
8000	1.41	2.04	2.78	3.43	4.06	4.98	5.51	6.55	7.14	7.53
9000	1.63	2.36	2.99	3.85	4.72	5.44	6.25	6.88	7.59	8.20
10000	1.80	2.62	3.51	4.25	4.97	6.01	6.33	7.81	8.31	9.15

of the second experiment, named SimCities, was to detect duplicate cities within DBpedia by comparing their labels. The purpose of the last experiments, named Diseases, was to map diseases from MESH[10] with the corresponding diseases in LinkedCT[11] by comparing their labels. The configuration files for all three experiments are available in the LIMES distribution.

Table 3: Absolute runtimes of LIMES and SILK. All times are given in seconds. The values in the second row of the table are the similarity thresholds.

	LIMES					**SILK**
	0.95	0.9	0.85	0.8	0.75	
DrugBank	86	120	175	211	252	1,732
SimCities	523	979	1,403	1,547	1,722	33,786
Diseases	546	949	1,327	1,784	1,882	17,451

Figure 8 shows a relative comparison of the runtimes of SILK and LIMES. The absolute runtimes are given in Table 3. LIMES outperforms SILK in all experimental settings. It is important to notice that the difference in performance grows with the (product of the) size of the source and target knowledge bases. While LIMES ($\theta = 0.75$) necessitates approximately 30% of SILK's computation time for the Drugbank experiment, it requires only roughly 5% of SILK's time for the SimCities experiments. The difference in performance is even more significant when the threshold is set higher. For example, $\theta = 0.95$ leads to LIMES necessitating only 1.6% of SILK's runtime in the SimCities experiment. The potential of our approach becomes even more obvious when one takes into consideration that we did not vary the number of exemplars in this experiment. Setting optimal values for the number of exemplars would have led to even smaller runtimes as shown by our experiments with synthetic data.

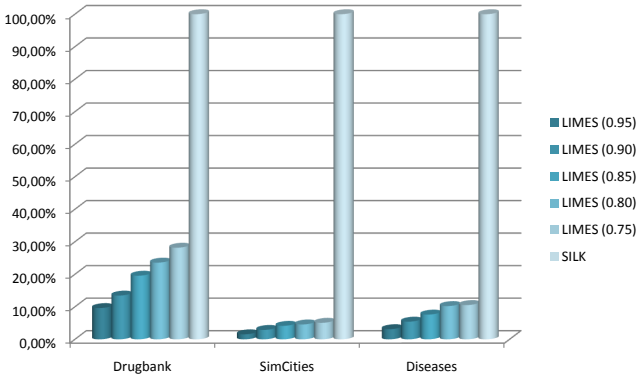


Figure 8: Comparison of the relative runtimes of SILK and LIMES. The number in brackets in the legend are the values of the θ threshold.

6. DISCUSSION AND FUTURE WORK

We presented the LIMES framework, which implements a very time-efficient approach for the discovery of links between knowledge bases on the Linked Data Web. We evaluated our approach both with synthetic and real data and showed that it outperforms state-of-the-art approaches with respect to the number of comparisons and runtime. In particular, we showed that the speedup of our approach grows with the a-priori time complexity of the mapping task, making our framework especially suitable for handling large-scale matching tasks (cf. results of the SimCities experiment).

In our experiments, we used $\lfloor\sqrt{|T|}\rfloor$ as number of exemplars. Finding a means for the automatic computation of an optimal value for the number of exemplars remains future work. So far, the results presented above seem to hint towards the relation between the number x and the number y of comparisons for a particular threshold θ to be of the form $y = \frac{1}{\alpha x} + \beta x + \gamma$. Providing an approach for computing the values α, β and γ as functions of θ automatically would allow to provide end-users with even better runtimes.

The main drawback of LIMES is that it is restricted to metric spaces. Thus, some popular semi-metrics such as JaroWinkler [21] can not be accelerated with LIMES. To ensure that our framework can be used even with these measures, we have implemented the brute force approach as a fall-back for comparing instances in such cases. In future work, we will take a closer look at semi-metrics and aim at finding a relaxed triangular inequality that applies to each of them. Based on these inequalities, our framework will also use semi-metrics to compute exemplar and render link discovery based on these measures more efficient.

With LIMES we have laid the groundwork for a substan-

[10] http://mesh.bio2rdf.org/sparql
[11] http://data.linkedct.org/sparql

tial flexibilization of link discovery on the Web of Data. By providing a novel framework that addresses the performance challenge, we have also developed the backbone necessary to address other substantial challenges of link discovery on the Web of Data. The main quality criteria of matching tasks are performance, precision/recall and ease-of-use. These quality criteria are not unconnected. Rather, they influence each other strongly. In particular, we can trade performance for precision/recall and substantially increase the ease-of-use for users with faster matching algorithms by using supervised matching. On the other hand, increasing precision/recall with faster matching algorithms can be carried out by involving the user into an interactive feedback loop, where she feeds hints on the quality of matching results back into the system. These hints can be used subsequently for automatically adjusting thresholds and/or matcher configurations. Such semi-automatic adjustments of thresholds and matcher configurations can in turn dramatically increase the ease-of-use for the users of a matching framework, since they are released from the burden of creating an comprehensive a-priori configuration. In the light of these considerations we see this work as the initial stage in a longer-term agenda.

We aim to explore the combination of LIMES with active learning strategies in a way, that a manual configuration of the tool becomes unnecessary. Instead, matching results will be computed quickly by using the exemplars in both the source and target knowledge bases. Subsequently, they will be presented to the user who will give feedback to the system by rating the quality of found matches. This feedback in turn will be employed for improving the matching configuration and to generate a revised list of matching suggestions to the user. This iterative process will be continued until a sufficiently high quality (in terms of precision and recall) of matches is reached.

7. REFERENCES

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A nucleus for a web of open data. In *ISWC2008*, pages 722–735. Springer, 2008.

[2] R. Baxter, P. Christen, and T. Churches. A comparison of fast blocking methods for record linkage. In *KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, 2003.

[3] D. Ben-David, T. Domany, and A. Tarem. Enterprise data classification using semantic web technologies. In *ISWC2010*, 2010.

[4] R. Bhagdev, S. Chapman, F. Ciravegna, V. Lanfranchi, and D. Petrelli. Hybrid search: Effectively combining keywords and semantic searches. In *ESWC 2008*, pages 554–568, 2008.

[5] M. Bilenko, B. Kamath, and R. J. Mooney. Adaptive blocking: Learning to scale up record linkage. In *ICDM '06*, pages 87–96. IEEE, 2006.

[6] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, 2009.

[7] J. Bleiholder and F. Naumann. Data fusion. *ACM Comput. Surv.*, 41(1):1–41, 2008.

[8] R. Cilibrasi and P. Vitanyi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, April 2005.

[9] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19:1–16, 2007.

[10] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007.

[11] H. Glaser, I. C. Millard, W.-K. Sung, S. Lee, P. Kim, and B.-J. You. Research on linked data and co-reference resolution. Technical report, University of Southampton, 2009.

[12] H. Köpcke, A. Thor, and E. Rahm. Comparative evaluation of entity resolution approaches with fever. *Proc. VLDB Endow.*, 2(2):1574–1577, 2009.

[13] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8, 1966.

[14] V. Lopez, V. Uren, M. R. Sabou, and E. Motta. Cross ontology query answering on the semantic web: an initial evaluation. In *K-CAP '09*, pages 17–24, New York, NY, USA, 2009. ACM.

[15] L. Ma, X. Sun, F. Cao, C. Wang, and X. Wang. Semantic enhancement for enterprise data management. In *ISWC2009*, 2009.

[16] J. McCusker and D. McGuinness. Towards identity in linked data. In *Proceedings of OWL Experiences and Directions Seventh Annual Workshop*, 2010.

[17] Y. Raimond, C. Sutton, and M. Sandler. Automatic interlinking of music datasets on the semantic web. In *1st Workshop about Linked Data on the Web*, 2008.

[18] F. Scharffe, Y. Liu, and C. Zhou. Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. In *Proc. IJCAI 2009 IR-KR Workshop*.

[19] J. Urbani, S. Kotoulas, J. Maassen, F. van Harmelen, and H. Bal. Owl reasoning with webpie: calculating the closure of 100 billion triples. In *ESWC2010*, 2010.

[20] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Discovering and maintaining links on the web of data. In *ISWC 2009*, pages 650–665. Springer, 2009.

[21] W. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Bureau of the Census, 1999.

[22] W. Winkler. Overview of record linkage and current research directions. Technical report, Bureau of the Census - Research Report Series, 2006.

[23] R. C.-W. Wong, Y. Tao, A. W.-C. Fu, and X. Xiao. On efficient spatial matching. In *VLDB '07*, pages 579–590. VLDB Endowment, 2007.

[24] Z. Yao, L. Gao, and X. S. Wang. Using triangle inequality to efficiently process continuous queries on high-dimensional streaming time series. In *SSDBM*, pages 233–236. IEEE, 2003.

[25] L. Ye, X. Wang, D. Yankov, and E. J. Keogh. The asymmetric approximate anytime join: A new primitive with applications to data mining. In *SIAM Int. Conf. on Data Mining*, pages 363–374, 2008.