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

Axel-Cyrille Ngonga Ngomo, Sören Auer
AKSW/BIS, Institut für Informatik
Universität Leipzig
Postfach 100920, 04009 Leipzig, Germany
{ngonga|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. Yet, this task 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 during the mapping process to filter out a large number of those instance pairs that do not suffice the mapping conditions. We present the mathematical foundation and the core algorithms employed in LIMES. We evaluate our algorithms with synthetic data to elucidate their behavior on small and large data sets with different configurations and compare the runtime of LIMES with another state-of-the-art link discovery tool.

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 [Volz *et al.*, 2009]. While the number of triples in data sources increases steadily and has surpassed 25 billions, links still constitute less than 5% of the total number of triples available on the Linked Data Web[1]. 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 [Lopez *et al.*, 2009], large-scale inferences [Urbani *et al.*, 2010] and data integration [Ben-David *et al.*, 2010].

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

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 the source S and target 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 $|S||T|$, an unpractical proposition as soon as the source and target knowledge bases become large. For example, discovering duplicate cities in DBpedia [Auer *et al.*, 2008] 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.

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 instance 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. Our contributions are as follows:

- We present a lossless and time-efficient approach for the large-scale matching of instances in metric spaces.
- We present two novel algorithms for the efficient approximation of distances within metric spaces based on the triangle inequality.
- We evaluate LIMES on synthetic data by using the number of comparisons necessary to complete the given matching task and with real data against the SILK framework [Volz *et al.*, 2009] with respect to the runtime.

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.

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 [Cilibrasi and Vitanyi, 2005], spatial matching [Wong *et al.*, 2007] and query processing [Yao *et al.*, 2003]. Yet, to the best of our knowledge, it has never been used previously for link discovery.

Current frameworks for link discovery on the Web of Data 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) [Glaser *et al.*, 2009] computes links between universities and conferences while GNAT [Raimond *et al.*, 2008] discovers 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 [Scharffe *et al.*, 2009] implements a five-step approach that comprises the preprocessing, matching, fusion, interlink and post-processing of data sets. SILK [Volz *et al.*, 2009] (Version 2.0) is a time-optimized tool 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 is not guaranteed to be 1. In addition, SILK allows the manual configuration of data blocks to minimize the runtime of the matching process. Yet, this blocking approach is not lossless.

The task of discovering links between knowledge bases is closely related with record linkageand de-duplication [Bleiholder and Naumann, 2008]. 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 (see e.g. [Köpcke *et al.*, 2009]) have been developed to address the problem of the quadratic time complexity of brute force comparison methods. The rationale is to filter out obvious non-matches efficiently before executing the more detailed and time-consuming comparisons.

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 [Frey and Dueck, 2007] 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 without sacrificing precision.

3 Mathematical Framework

In this section, we present the mathematical principles underlying the LIMES framework. We present the formal definition of a matching task within metric spaces. Then, we use this definition to infer upper and lower boundary conditions for distances based on the triangle inequality. Finally, 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 notation. Let A be an affine space. m, m_1, m_2, m_3 symbolize metrics on A; x, y and z represent points from A and α, β, γ and δ are scalars, i.e., elements of $\mathbb{R}$. Furthermore, we assume that (A, m) is a metric space.

Definition 1 (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 triangle inequality (TI) that holds in metric spaces.

3.2 Distance Approximation Based on the Triangle Inequality

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. First, 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 [Frey and Dueck, 2007]). 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.

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 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

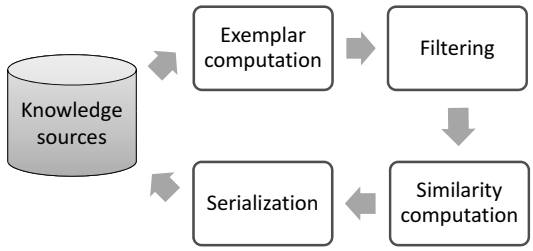


Figure 1: General Workflow of LIMES

The general workflow implemented by the LIMES framework comprises four steps (as depicted in Figure 1). Given the source S, the target T and the threshold θ, LIMES first computes a set E of 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 dramatically. 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 $(s, t, m(s, t))$ with $m(s, t) \leq \theta$ are stored in a user-specified format such as NTriples (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 rationale behind the computation of exemplars in LIMES is to select a set of instances 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 very dissimilar, i.e. very distant from each other. The approach we use to generate such exemplars is shown in Algorithm 1.

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 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). This algorithm has a constant time complexity of $O(|E||T|)$.

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

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$m(s, \lambda_i^e) \leq \theta$ 3. $M = M \cup \{\lambda_i^e\}$
    else
      break;
    end
  end
end
4. return M;

Algorithm 2: Comparison algorithm

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.

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 is possible since 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 the brute force approach. However, as our evaluation with both synthetic and real data shows, a correct parameterization of LIMES leads to dramatically reduced comparisons and runtime.

5 Evaluation

Our evaluation is based on the implementation of the LIMES framework consisting of seven main modules (including a dashboard) of which each can be extended to accommodate new or improved functionality[2]

We elucidate the following four central evaluation questions:

Q_1: *What is the best number of exemplars?*

Q_2: *What is the relation between the threshold θ and the total number of comparisons?*

Q_3: *Does the assignment of S and T matter?*

Q_4: *How does LIMES compare to other approaches?*

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 to measure string similarity. Each of the matching tasks was carried out five times and we report average values in the following.

To address the first Q_1 and Q_2, we considered four matching tasks on knowledge bases of sizes between $2,000$ and $10,000$. 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 (see Figure 2). 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.

[2]LIMES is available as an open-source framework at `http://limes.sf.net`.

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

	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2	0.6	0.9	1.2	1.4	1.6	2.0	2.3	2.5	2.7
3	0.9	1.2	1.6	2.0	2.1	2.7	2.9	3.4	3.6
4	1.1	1.6	2.0	2.5	2.9	3.1	3.6	3.9	4.5
5	1.4	1.9	2.3	2.8	3.4	3.9	4.2	4.8	5.5
6	1.6	2.1	2.8	3.3	3.9	4.4	5.0	5.4	6.1
7	1.9	2.6	3.2	3.7	4.4	5.1	5.7	6.4	6.6
8	2.0	2.8	3.4	4.1	5.0	5.5	6.6	7.1	7.5
9	2.4	3.0	3.9	4.7	5.4	6.3	6.9	7.6	8.2
10	2.6	3.5	4.3	5.0	6.0	6.3	7.8	8.3	9.2

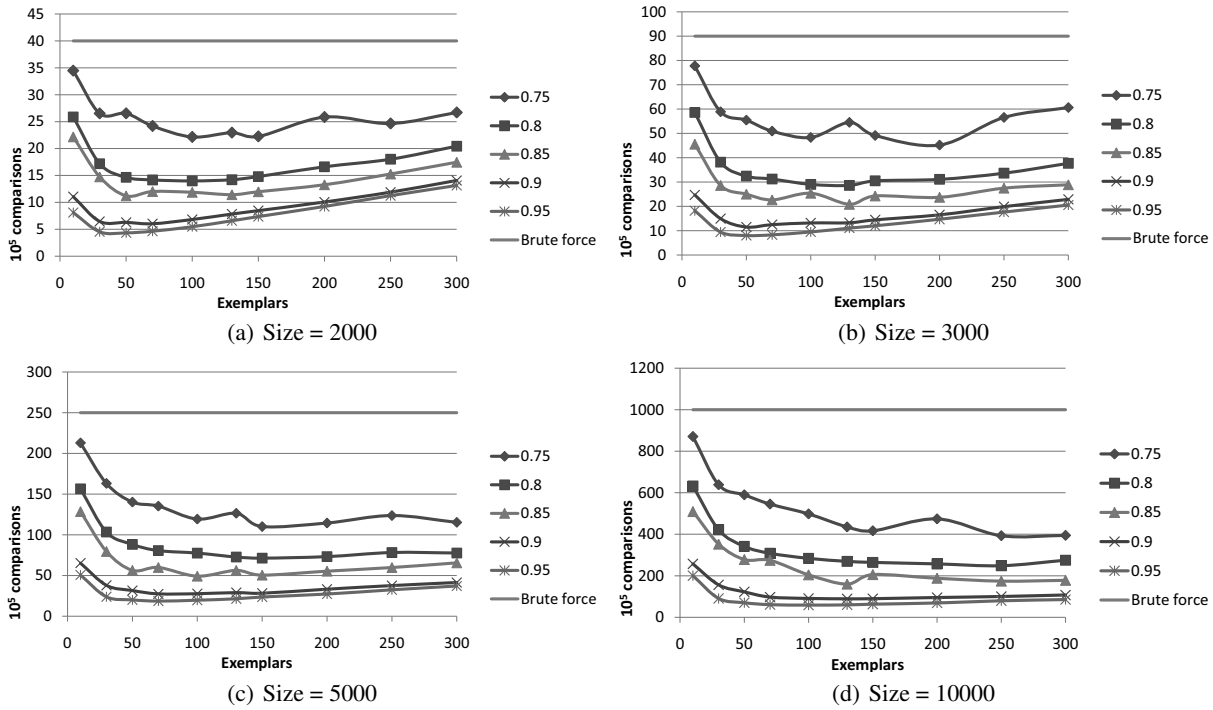
An analysis of the results displayed in Figure 2 also allows to answer Q_2. The higher the value of θ, the smaller the number of comparisons. This is due to the stricter filtering that results from the higher threshold and consequently leads to a smaller number of required comparisons. An important observation 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 $2,000$ 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.

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 (to the best of our knowledge) only time-optimized link discovery framework SILK. Non-optimized frameworks would perform like a brute-force approach, which is clearly inferior to LIMES. To ensure an objective comparison of the runtimes, we only considered the time necessary for both frameworks to carry out the comparisons in our evaluation . Each of the time measurements was carried out three times and only the best runtime was considered. Note that there was no significant different between the different runtimes of LIMES. Every time measurement experiment was carried out as a sin-



(a) Size = 2000 (b) Size = 3000

(c) Size = 5000 (d) Size = 10000

Figure 2: 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.

gle 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. We did not use SILK's blocking feature because it loses some links and we were interested in lossless approaches. The number of exemplars for LIMES was set to $\sqrt{|T|}$.

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.

	Drugs	**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

The experiments on real data were carried out in three different settings as shown in Table 2. The goal of the first experiment, named Drugs, was to map drugs in DBpedia[3] and Drugbank[4] by comparing their labels. The goal of the second experiment, named SimCities, was to detect duplicate cities within DBpedia by comparing their labels. The purpose of the last experiment, named Diseases, was to map diseases from MESH[5] with the corresponding diseases in LinkedCT[6] 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	
Drugs	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 3 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 (θ = 0.75) necessitates approximately 30% of SILK's computation time for the Drugs 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, θ = 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

[3]http://dbpedia.org/sparql

[4]http://www4.wiwiss.fu-berlin.de/drugbank/sparql

[5]http://mesh.bio2rdf.org/sparql

[6]http://data.linkedct.org/sparql

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.

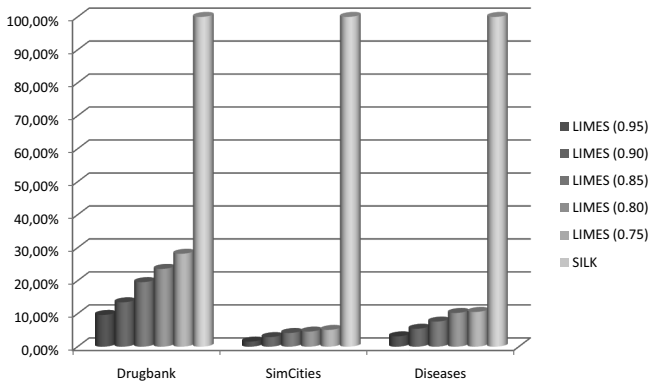


Figure 3: Comparison of the relative runtimes of SILK and LIMES. The number in brackets in the legend are the values of the θ threshold.

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).

The current approach to the computation of exemplars does not take the distribution of data in the metric into consideration. In future work, we will integrate this feature. The main drawback of LIMES is that it is restricted to metric spaces. Thus, some popular semi-metrics such as JaroWinkler [Winkler, 1999] 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. One can easily show that our approach can be extended to semi-metrics. 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. We also aim to explore the combination of LIMES with active learning strategies in a way, that a manual configuration of the tool becomes unnecessary.

Acknowledgement

This work was supported by the Eurostars grant SCMS E!4604 and the EU FP7 grant LOD2 (GA no. 257943).

References

[Auer *et al.*, 2008] Sören Auer, Chris Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A nucleus for a web of open data. In *ISWC2008*, pages 722–735. Springer, 2008.

[Ben-David *et al.*, 2010] David Ben-David, Tamar Domany, and Abigail Tarem. Enterprise data classification using semantic web technologies. In *ISWC2010*, 2010.

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

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

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

[Glaser *et al.*, 2009] Hugh Glaser, Ian C. Millard, Won-Kyung Sung, Seungwoo Lee, Pyung Kim, and Beom-Jong You. Research on linked data and co-reference resolution. Technical report, University of Southampton, 2009.

[Köpcke *et al.*, 2009] Hanna Köpcke, Andreas Thor, and Erhard Rahm. Comparative evaluation of entity resolution approaches with fever. *Proc. VLDB Endow.*, 2(2):1574–1577, 2009.

[Lopez *et al.*, 2009] Vanessa Lopez, Victoria Uren, Marta Reka Sabou, and Enrico Motta. Cross ontology query answering on the semantic web: an initial evaluation. In *K-CAP '09*, pages 17–24, 2009.

[Raimond *et al.*, 2008] Yves Raimond, Christopher Sutton, and Mark Sandler. Automatic interlinking of music datasets on the semantic web. In *1st Workshop about Linked Data on the Web*, 2008.

[Scharffe *et al.*, 2009] Franois Scharffe, Yanbin Liu, and Chuguang Zhou. Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. In *Proc. IJCAI 2009 IR-KR Workshop*, 2009.

[Urbani *et al.*, 2010] Jacopo Urbani, Spyros Kotoulas, Jason Maassen, Frank van Harmelen, and Henri Bal. Owl reasoning with webpie: calculating the closure of 100 billion triples. In *ESWC2010*, 2010.

[Volz *et al.*, 2009] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Discovering and maintaining links on the web of data. In *ISWC 2009*, pages 650–665. Springer, 2009.

[Winkler, 1999] William Winkler. The state of record linkage and current research problems. Technical report, U.S. Bureau of the Census, 1999.

[Wong *et al.*, 2007] Raymond Chi-Wing Wong, Yufei Tao, Ada Wai-Chee Fu, and Xiaokui Xiao. On efficient spatial matching. In *VLDB*, pages 579–590, 2007.

[Yao *et al.*, 2003] Zhengrong Yao, Like Gao, and X. Sean Wang. Using triangle inequality to efficiently process continuous queries on high-dimensional streaming time series. In *SSDBM*, pages 233–236. IEEE, 2003.