Time-Efficient Execution of Bounded Jaro-Winkler Distances

Kevin Dreßler and Axel-Cyrille Ngonga Ngomo

University of Leipzig
AKSW Research Group
Augustusplatz 10, 04103 Leipzig
Germany

Abstract. Over the last years, time-efficient approaches for the discovery of links between knowledge bases have been regarded as a key requirement towards implementing the idea of a Data Web. A considerable portion of the information contained available as RDF on the Web pertains to persons. Thus, efficient and effective measures for comparing names are central to facilitate the integration of information about persons on the Web of Data. The Jaro-Winkler measure has been developed especially for the purpose of comparing person names. Hence, we present a novel approach for the efficient comparison of sets of strings using this measure. We evaluate our approach on several datasets derived from DBpedia 3.9 and containing up to 10^5 strings and show that it scales linearly with the size of the data for large thresholds. We also evaluate our approach against SILK and show that we outperform it even on small datasets.

1 Introduction

The Linked Open Data Cloud (LOD Cloud) has developed to a compendium of more than 2000 datasets over the last few years.[1] Currently, data sets pertaining to more than 14 million persons have already been made available on the Linked Data Web.[2] While this number is impressive on its own, it is well known that the population of the planet has surpassed 7 billion people. Hence, the Web of Data contains information on less that 1% of the overall population of the planet (counting both the living and the dead). The output of open-government movements,[3] scientific conferences,[4] health data[5] and similar endeavours yet promises to make massive amounts of data pertaining to persons available in the near future. Dealing with this upcoming increase of the number of person-related resources requires providing means to integrate these datasets with the aim to facilitate statistical analysis, data mining, personlization, etc. However, while the

[1] See `http://stats.lod2.eu` for an overview of the current state of the Cloud. Last access: July 11th, 2014.
[2] Data collected from `http://stats.lod2.eu`. Last access: July 11th, 2014.
[3] See for example `http://data.gov.uk/`.
[4] See for example `http://data.semanticweb.org/`
[5] `http://aksw.org/Projects/GHO`

number of datasets on the Linked Data Web grows drastically, the number of links between datasets still stagnates.[6] Addressing this lack of links requires solving two main problems: the quadratic time complexity of link discovery (efficiency) and the automatic support of the detection of link specifications (effectiveness). In this paper, we address the efficiency of the execution of bounded Jaro-Winkler measures,[7] which are known to be effective when comparing person names [10]. To this end, we derive equations that allow discarding a large number of computations while executing bounded Jaro-Winkler comparisons with high thresholds.

The contributions of this paper are as follows:

1. We derive length- and range-based filters that allow reducing the number of strings t that are compared with a string s .
2. We present a character-based filter that allows detecting whether two strings s and t share enough resemblance to be similar according to the Jaro-Winkler measure.
3. We evaluate our approach w.r.t. to its runtime and its scalability with several threshold settings and dataset sizes.

The rest of this paper is structured as follows: In Section 2, we present the problem we tackled as well as the formal notation necessary to understand this work. In the subsequent Section 3, we present the three approaches we developed to reduce the runtime of bounded Jaro-Winkler computations. We then evaluate our approach in Section 4. Related work is presented in Section 5, where we focus on approaches that aim to improve the time-efficiency of link discovery. We conclude in Section 6. The approach presented herein is now an integral part of LIMES.[8]

2 Preliminaries

In the following, we present some of the symbols and terms used within this work.

2.1 Link Discovery

In this work, we use *link discovery* as a hypernym for deduplication, record linkage, entity resolution and similar terms used across literature. The formal specification of link discovery adopted herein is tantamount to the definition proposed in [16]: Given a set S of source resources, a set T of target resources and a relation R, our goal is to find the set $M \subseteq S \times T$ of pairs (s, t) such that $R(s, t)$. If R is `owl:sameAs`, then we are faced with a *deduplication task*. Given that the explicit computation of M is usually a very complex endeavour, M is most commonly approximated by a set $M' = \{(s, t, \delta(s, t)) \in S \times T \times \mathbb{R}^+ : \sigma(s, t) \geq \theta\}$, where σ is a (potentially complex) similarity function and $\theta \in [0, 1]$ is a similarity threshold. Given that this problem is in $O(n^2)$, using naïve algorithms to compare large S and T is most commonly impracticable. Thus, time-efficient approaches for the computation of bounded measures

[6] `http://linklion.org`

[7] We use bounded measures in the same sense as [13], i.e., to mean that we are only interested in pairs of strings whose similarity is greater than or equal to a given lower bound.

[8] `http://limes.sf.net`

have been developed over the last years for measures such as the Levenshtein distance, Minkowski distances, trigrams and many more [15].

In this paper, we thus study the following problem: Given a threshold $\theta \in [0,1]$ and two sets of strings S and T, compute the set $M' = \{(s,t,\delta(s,t)) \in S \times T \times \mathbb{R}^+ : \sigma(s,t) \geq \theta\}$. Two categories of approaches can be considered to improve the runtime of measures: Lossy approaches return a subset M'' of M' which can be calculated efficiently but for which there are no guarantees that $M'' = M'$. Lossless approaches on the other hand ensure that their result set M'' is exactly the same as M'. In this paper, we present a lossless approach. To the best of our knowledge, only one other link discovery framework implements a lossless approach that has been designed to exploit the bound defined by the threshold θ to ensure a more efficient computation of the Jaro-Winkler distance, i.e., the SILK framework with the approach MultiBlock [9]. We thus compare our approach with SILK 2.6.0 in the evaluation section of this paper.

2.2 The Jaro-Winkler Similarity

Let Σ be the set of all the strings that can be generated by using an alphabet A. The Jaro measure $d_j : \Sigma \times \Sigma \to [0,1]$ is a string similarity measure approach which was developed originally for name comparison in the U.S. Census. This measure takes into account the number of character matches m and the ratio of their transpositions t:

$$d_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & \text{otherwise} \end{cases} \tag{1}$$

Here two characters are considered to be a match if and only only if (1) they are the same and (2) they are at most at a distance $w = \lfloor \frac{\max(|s_1|,|s_2|)}{2} \rfloor$ from each other. For example, for $s_1 = "Spears"$ and $s_2 = "Pears"$, the second s of s_1 matches the s of s_2 while the first s of s_1 does not match the s of s_2.

The Jaro-Winkler measure [27] is an extension of the Jaro distance. This extension is based on Winkler's observation that typing errors occur most commonly in the middle or at the end of a word, but very rarely in the beginning. Hence, it is legitimate to put more emphasis on matching prefixes if the Jaro distance exceeds a certain "boost threshold" b_t, originally set to 0.7.

$$d_w = \begin{cases} d_j & \text{if } d_j < b_t \\ d_j + (\ell p(1-d_j)) & \text{otherwise} \end{cases} \tag{2}$$

Here, ℓ denotes the length of the common prefix and p is a weighting factor. Winkler uses $p = 0.1$ and $\ell \leq 4$. Note that ℓp must not be greater than 1.

For the strings $s_1 = "DEMOCRACY", s_2 = "DEMOGARPHY"$ (with s_2 being intentionally misspelled) we get the following output of the Jaro-Winkler measure.

- $|s_1| = 9, |s_2| = 10$
- $w = 4$
- $m = 7$

- $t = 1$
- $d_j = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) = \frac{1}{3}\left(\frac{7}{9} + \frac{7}{10} + \frac{6}{7}\right) = 0.778$
- $d_w = d_j + \ell p\,(1 - d_j) = 0.867$

3 Improving the Runtime of Bounded Jaro-Winkler

The main principle behind reducing the runtime of the computation of measures is to reduce their reduction ratio. Here, we use a sequence of filters that allow discarding similarity computations while being sure that they would have led to a similarity score which would have been less than our threshold θ. To this end, we regard the problem as that of finding filters that return an upper bound estimation $\theta_e(s_1, s_2) \geq d_w(s_1, s_2)$ for some properties of the input strings that can be computed in constant time. For a given threshold θ, if $\theta_e(s_1, s_2) \leq \theta$, then we can safely ignore the input (s_1, s_2).

3.1 Length-based filters

In the following, we denoted the length of a string s with $|s|$. Our first filter is based on the insight that large length differences are a guarantee for poor similarity. For example, the strings "a" and "$alpha$" cannot have a Jaro-Winkler similarity of 1 by virtue of their length difference. We can formalize this idea as follows: Let s_1 and s_2 be strings with respective lengths $|s_1|$ and $|s_2|$. Without loss of generality, we will assume that $|s_1| \leq |s_2|$. Moreover, let m be the number of matches across s_1 and s_2. Because $m \leq |s_1|$, we can substitute m with $|s_1|$ and gain the following upper bound estimation for $d_j(s_1, s_2)$:

$$d_j = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) \leq \frac{1}{3}\left(1 + \frac{|s_1|}{|s_2|} + \frac{|s_1| - t}{|s_1|}\right) \quad (3)$$

Now the lower bound for the number t of transpositions is 0. Thus, we obtain the following equation.

$$d_j \leq \frac{1}{3}\left(1 + \frac{|s_1|}{|s_2|} + 1\right) \leq \frac{2}{3} + \frac{|s1|}{3|s2|} \quad (4)$$

The application of this approximation on Winkler's extension is trivial:

$$d_w = d_j + \ell \cdot p \cdot (1 - d_j) \leq \frac{2}{3} + \frac{|s1|}{3|s2|} + \ell \cdot p \cdot \left(\frac{1}{3} - \frac{|s1|}{3|s2|}\right) = \theta_e \quad (5)$$

Consider the pair s_1 = "$bike$" and s_2 = "$bicycle$" and a threshold $\theta = 0.9$. Applying the estimation for Jaro we get $d_j \leq \frac{2}{3} - \frac{4}{3 \cdot 7} = 0.857$. This exceeeds the boost threshold, so we use equation 5 to compute $\theta_e(s_1, s_2) = 0.885$. Now we do not have to actually compute $d_w(s_1, s_2)$, since $\theta_e(s_1, s_2) < \theta$.

By using this approach we can decide in $O(1)$[9] if a given pairs score is greater than a given threshold, which saves us the much more expensive score computation for a big number of pairs, provided that the input strings sufficiently vary in length.

[9] In most programming languages, especially Java (which we used for our implementation), the length of string is stored in a variable and can thus be accessed in constant time.

3.2 Filtering ranges by length

The approach described above can be reversed to limit the number of pairs that we are going to be iterated over. To this end, we can construct a $index : \mathbb{N} \rightarrow 2^{\Sigma}$ which maps strings lengths $l \in \mathbb{N}$ to all strings s with $|s| = l$. With the help of this index, we can now determine the set of strings t that should be compared with the subset $S(l)$ of S that only contains strings of length l. We go about using this insight by computing the upper and lower bound for the length of a string t that should be compared with a string s. This is basically equivalent to asking what is the minimum length difference $||s| - |t||$ so that $\theta \geq \theta_e(s,t)$ is satisfied. We transpose equation 5 to the following for our lower bound:

$$|t| \geq \left\lfloor 3|s| \frac{\theta - \ell p}{1 - \ell p} - 2|s| \right\rfloor \tag{6}$$

Analogously, we can derive the following upper bound:

$$|t| \leq \left\lceil \frac{|s|}{3\frac{\theta - \ell p}{1 - \ell p} - 2} \right\rceil \tag{7}$$

For example, consider a list of strings S with equally distributed, distinct string lengths $(4, 7, 11, 18)$. Using Equation 6 and Equation 7 we obtain Table 1. Taking into account the last column of the table, we will save a total of $\frac{3}{8}$ comparisons.

Table 1. Bounds for distinct string lengths ($\theta = 0.9$)

$\|t\|$	$\|s\|_{min}$	$\|s\|_{max}$	sizes in range
4	2	8	$(4, 7)$
7	3	14	$(4, 7, 11)$
11	5	22	$(7, 11, 18)$
18	9	36	$(11, 18)$

3.3 Filtering by character frequency

An even more fine-grained approach can be chosen to filter out computations. Let $e : \Sigma \times A \rightarrow \mathbb{N}$ be the function with returns the number of occurrences of a given character c in a string s. For the strings s_1 and$_2$, the number of maximum possible matches m_{max} can be expressed as

$$m_{max} = \sum\nolimits_{c \in s_1} min(e(s_1, c), e(s_2, c)) \geq m \tag{8}$$

Consequently, we can now substitute m for m_{max} in the Jaro distance computation:

$$d_j(s_1, s_2) = \frac{1}{3}\left(\frac{m_{max}}{|s_1|} + \frac{m_{max}}{|s_2|} + \frac{m_{max} - t}{m_{max}}\right) \leq \frac{1}{3}\left(\frac{m_{max}}{|s_1|} + \frac{m_{max}}{|s_2|} + 1\right) \tag{9}$$

We can thus derive that $d_j(s_1, s_2) \geq \theta$ iff

$$m_{max} \geq \frac{(3\theta - 1)|s_1||s_2|}{|s_1| + |s_2|}. \tag{10}$$

For instance, let $s_1 = "astronaut", s_2 = "astrochimp"$. The retrieval of m_{max} ist shown in Table 2.

Table 2. Calculation of m_{max}

c	$e(s_1, c)$	$e(s_2, c)$	$min(e(s_1, c), e(s_2, c))$	m_{max}
a	2	1	1	1
c	0	1	0	1
h	0	1	0	1
i	0	1	0	1
m	0	1	0	1
n	1	0	0	1
o	1	1	1	2
p	0	1	0	2
r	1	1	1	3
s	1	1	1	4
t	2	1	1	5
u	1	0	0	5

The question that remains to answer is how well do these filters perform on real person data. We answer this question empirically in the subsequent section.

4 Evaluation

The aim of our evaluation was to study how well our approach performs on real data. We chose DBpedia 3.9 as a source of data for our experiments as it contains data pertaining to 1.1 million persons and thus allows for both fine-grained evaluations and scalability evaluations. All experiments where deduplication experiments, i.e., $S = T$. We considered the list of all `rdfs:label` in DBpedia in our runtime evaluation and scalability experiments. We also computed the runtime of our approach on up to 10^5 labels for our scalability experiments. All experiments were performed on a 2.5 GHz Intel Core i5 machine with 16GB RAM running OS X 10.9.3.

4.1 Runtime Evaluation

In our first series of experiments, we evaluated the runtime of all filter combinations against the naïve approach on a small dataset containing 1000 labels from DBpedia. The results of our evaluation are shown in Figure 4.1. This evaluation suggests that all filters outperform the naïve approach. Moreover, the combination of all filtersl lead to the best overall runtime in most cases. Interestingly, the character-based filter leads to a significant reduction of the number of comparisons (see Figure 2) by more than 2 orders of magnitude. However, the runtime improvement is not as substantial. This result seems to indicate that the lookup in the character indexes is very time-demanding. We will thus aim to improve our character indexing in future work. Overall, the results on this dataset already shows that we outperform the naïve approach by more than an order of magnitude when θ is high. The runtimes on a larger sample of size 10^4 show an even better improvement (see Figure 3). This suggests that the relative improvement of our approach improves with the size of the problem.

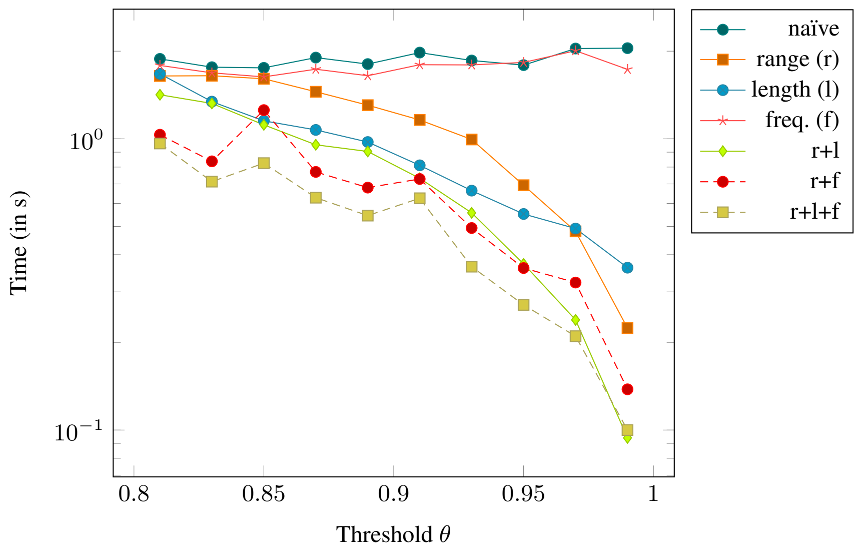

Fig. 1. Runtime comparison on input size 1000, scaling threshold

4.2 Scalability Evaluation

The aim of the scalability evaluation was to measure how well our approach deals with datasets of growing size datasets. In our first set of experiments, we looked at the growth of the runtime of our approach on datasets of growing sizes. Our results suggest that our approach grows linearly with the number of labels contained in S and T (see Figure 5). This suggests that the runtime of our approach can be easily predicted for large datasets, which of importance when asking users to wait for the results of the computation. The second series of scalability experiments looked at the runtime behaviour of our approach on a large dataset with 10^5 labels. Our results suggest that the runtime of our approach falls superlinearly with an increase of the threshold θ (see Figure 4). This behaviour suggest that our approach is especially useful on clean datasets, where high thresholds can be used for link discovery.

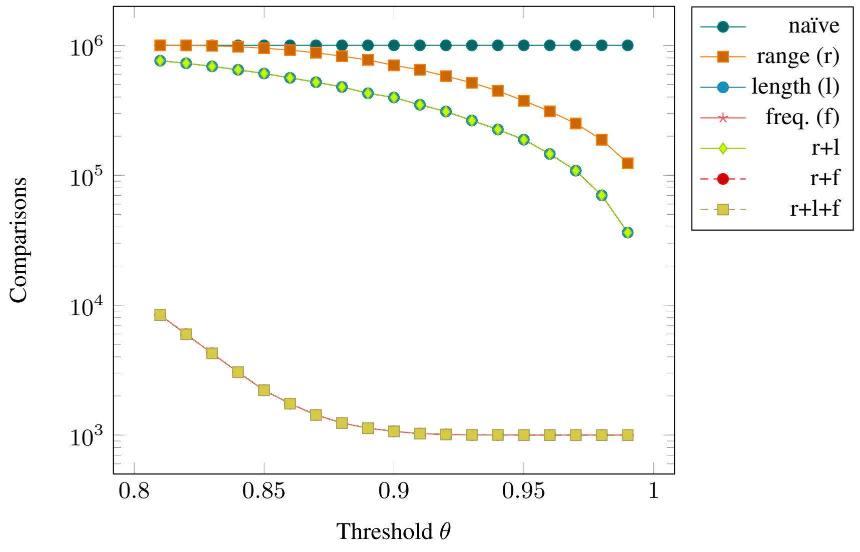

Fig. 2. Comparison of number of similarity computations on input size 1000, scaling threshold

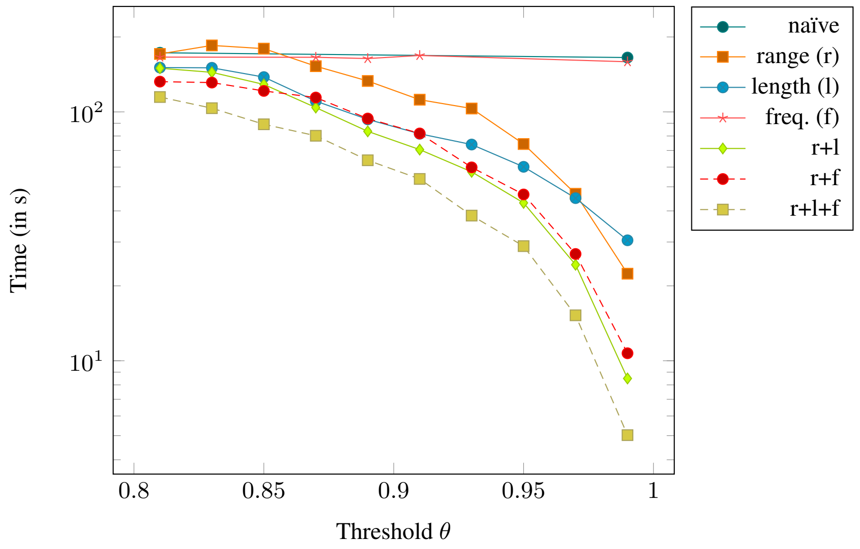

Fig. 3. Runtimes on sample of DBpedia labels with size 10^4, scaling threshold

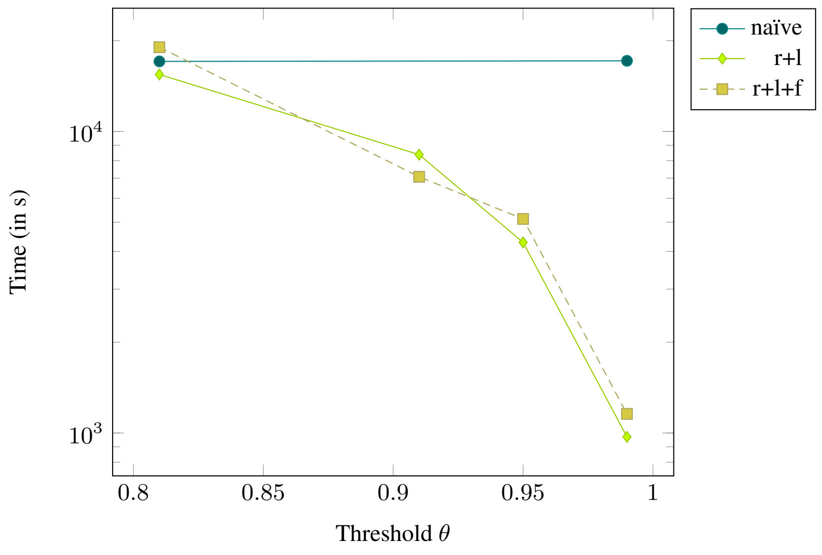

Fig. 4. Runtimes on sample of DBpedia labels with size 10^5, scaling threshold

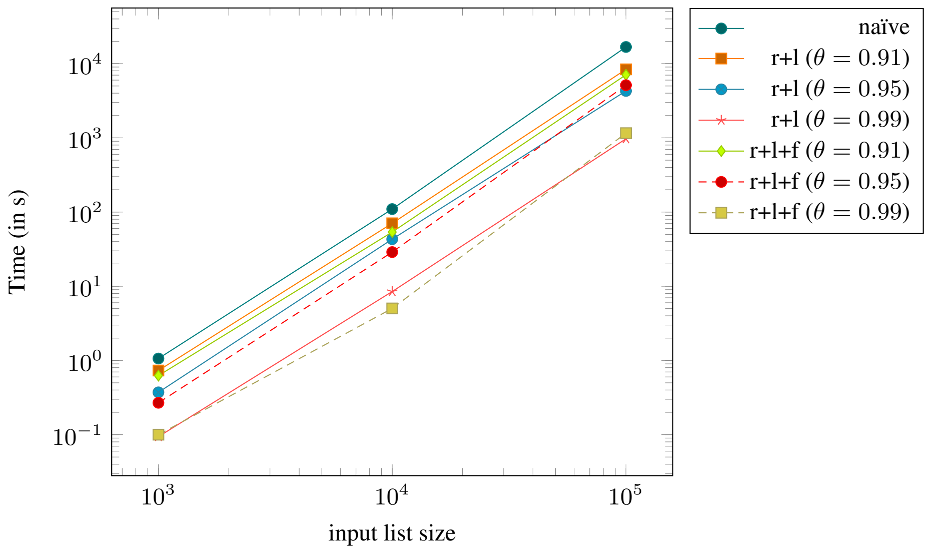

Fig. 5. Runtimes with multiple thresholds θ for growing input sizes

4.3 Comparison with existing approaches

We compared our approach with SILK2.6.0. To this end, we retrieved all `rdfs:label` of instances of subclasses of `Person`. We only compared with SILK on small datasets (i.e., on classes with small numbers of instances) as the results on these small datasets already showed that we outperform SILK consistently.[10] Our results are shown in Table 3. They suggest that the absolute difference in runtime grows with the size of the datasets. Thus, we did not consider testing larger datasets against SILK as in the best case, we were already 4.7 times faster than SILK (Architect dataset, $\theta = 0.95$).

Table 3. Runtimes (in seconds) of our approach (OA) and SILK 2.6.0

DBpedia Class	Size	OA(0.8)	OA(0.9)	OA(0.95)	SILK(0.8)	SILK(0.9)	SILK(0.95)
Actors	9509	15.07	10.13	6.38	27	25	25
Architect	3544	5.58	5.48	2.32	11	11	11
Criminal	5291	11.54	7.77	4.52	18	18	18

5 Related Work

The work presented herein is related to record linkage, deduplication, link discovery and the efficient computation of Hausdorff distances. An extensive amount of literature has been published by the database community on record linkage (see [11,6] for surveys). With regard to *time complexity*, time-efficient deduplication algorithms such as PPJoin+ [29], EDJoin [28], PassJoin [12] and TrieJoin [26] were developed over the last years. Several of these were then integrated into the hybrid link discovery framework LIMES [16]. Moreover, dedicated time-efficient approaches were developed for LD. For example, RDF-AI [24] implements a five-step approach that comprises the preprocessing, matching, fusion, interlink and post-processing of data sets. [17] presents an approach based on the Cauchy-Schwarz that allows discarding a large number of unnecessary computations. The approaches HYPPO [14] and $\mathcal{HR}^3$ [15] rely on space tiling in spaces with measures that can be split into independent measures across the dimensions of the problem at hand. Especially, $\mathcal{HR}^3$ was shown to be the first approach that can achieve a relative reduction ratio r' less or equal to any given relative reduction ratio $r > 1$. Standard blocking approaches were implemented in the first versions of SILK and later replaced with MultiBlock [9], a lossless multi-dimensional blocking technique. KnoFuss [20] also implements blocking techniques to achieve acceptable runtimes. Further approaches can be found in [25,4,21,22,7].

In addition to addressing the runtime of link discovery, several machine-learning approaches have been developed to learn link specifications (also called linkage rules) for link discovery. For example, machine-learning frameworks such as FEBRL [2] and MARLIN [1] rely on models such as Support Vector Machines [3] and decision

[10] We ran SILK with -Dthreads = 1 for the sake of fairness.

trees [23] to detect classifiers for record linkage. RAVEN [18] relies on active learning to detect linear or Boolean classifiers. The EAGLE approach [19] combines active learning and genetic programming to detect link specifications. KnoFuss [20] goes a step further and presents an unsupervised approach based on genetic programming for finding accurate link specifications. Other record deduplication approaches based on active learning and genetic programming are presented in [5,8].

6 Conclusion and Future Work

In this paper, we present a novel approach for the efficient execution of bounded Jaro-Winkler computations. Our approach is based on three filters which allow discarding a large number of comparisons. While our evaluation suggests that the filters are complementary, the character-based filter seems not to contribute to a significant reduction of the runtime once we deal with large datasets. We showed that our approach scales linearly with the amount of data it is faced with. Moreover, we showed that our approach can be make effective use of large thresholds by reducing the total runtime of the approach considerably. We also compared our approach with the state-of-the-art framework SILK 2.6.0 and showed that we outperform it on all datasets.
In future work, we will study the character-based filter in more detail and aim to eradicate its exact performace bottleneck. Moreover, we will evaluate partitioning of datasets and parallelization of filters to further improve the runtime of large datasets. Finally, we will test whether our approach improves the accuracy of specification detection algorithms such as EAGLE.

References

1. Mikhail Bilenko and Raymond J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 39–48, New York, NY, USA, 2003. ACM.
2. Peter Christen. Febrl -: an open source data cleaning, deduplication and record linkage system with a graphical user interface. In *KDD*, pages 1065–1068, 2008.
3. Nello Cristianini and Elisa Ricci. Support vector machines. In *Encyclopedia of Algorithms*. 2008.
4. Philippe Cudré-Mauroux, Parisa Haghani, Michael Jost, Karl Aberer, and Hermann de Meer. idmesh: graph-based disambiguation of linked data. In *WWW*, pages 591–600, 2009.
5. J. De Freitas, G.L. Pappa, A.S. da Silva, M.A. Gonçalves, E. Moura, A. Veloso, A.H.F. Laender, and M.G. de Carvalho. Active learning genetic programming for record deduplication. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.
6. Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.
7. Jérôme Euzenat, Alfio Ferrara, Willem Robert van Hage, et al. Results of the Ontology Alignment Evaluation Initiative 2011. In *OM*, 2011.
8. Robert Isele and Christian Bizer. Learning expressive linkage rules using genetic programming. *PVLDB*, 5(11):1638–1649, 2012.
9. Robert Isele, Anja Jentzsch, and Christian Bizer. Efficient Multidimensional Blocking for Link Discovery without losing Recall. In *WebDB*, 2011.

10. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. Journal of the American Statistical Association 84(406):414–420, 1989.
11. Hanna Köpcke and Erhard Rahm. Frameworks for entity matching: A comparison. *Data Knowl. Eng.*, 69(2):197–210, 2010.
12. Guoliang Li, Dong Deng, Jiannan Wang, and Jianhua Feng. Pass-join: a partition-based method for similarity joins. *Proc. VLDB Endow.*, 5(3):253–264, November 2011.
13. Axel-Cyrille Ngonga Ngomo. Orchid - reduction-ratio-optimal computation of geo-spatial distances for link discovery. In *International Semantic Web Conference (1)*, pages 395–410, 2013.
14. Axel-Cyrille Ngonga Ngomo. A Time-Efficient Hybrid Approach to Link Discovery. In *OM*, 2011.
15. Axel-Cyrille Ngonga Ngomo. Link discovery with guaranteed reduction ratio in affine spaces with minkowski measures. In *International Semantic Web Conference (1)*, pages 378–393, 2012.
16. Axel-Cyrille Ngonga Ngomo. On link discovery using a hybrid approach. *J. Data Semantics*, 1(4):203–217, 2012.
17. Axel-Cyrille Ngonga Ngomo and Sören Auer. LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data. In *IJCAI*, pages 2312–2317, 2011.
18. Axel-Cyrille Ngonga Ngomo, Jens Lehmann, Sören Auer, and Konrad Höffner. Raven - active learning of link specifications. In *OM*, 2011.
19. Axel-Cyrille Ngonga Ngomo and Klaus Lyko. Eagle: Efficient active learning of link specifications using genetic programming. In *ESWC*, pages 149–163, 2012.
20. Andriy Nikolov, Mathieu d'Aquin, and Enrico Motta. Unsupervised learning of link discovery configuration. In *ESWC*, pages 119–133, 2012.
21. Andriy Nikolov, Victoria Uren, Enrico Motta, and Anne Roeck. Overcoming schema heterogeneity between linked semantic repositories to improve coreference resolution. In *Proceedings of the 4th Asian Conference on The Semantic Web*, ASWC '09, pages 332–346, Berlin, Heidelberg, 2009. Springer-Verlag.
22. George Papadakis, Ekaterini Ioannou, Claudia Niederée, Themis Palpanas, and Wolfgang Nejdl. Eliminating the redundancy in blocking-based entity resolution methods. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, JCDL '11, pages 85–94, New York, NY, USA, 2011. ACM.
23. S. R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(3):660–674, 1991.
24. Francois Scharffe, Yanbin Liu, and Chuguang Zhou. Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. In *Proc. IJCAI 2009 workshop on Identity, reference, and knowledge representation (IR-KR), Pasadena (CA US)*, 2009.
25. Dezhao Song and Jeff Heflin. Automatically generating data linkages using a domain-independent candidate selection approach. In *International Semantic Web Conference (1)*, pages 649–664, 2011.
26. Jiannan Wang, Guoliang Li, and Jianhua Feng. Trie-join: Efficient trie-based string similarity joins with edit-distance constraints. *PVLDB*, 3(1):1219–1230, 2010.
27. William E. Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research*, pages 354–359, 1990.
28. Chuan Xiao, Wei Wang, and Xuemin Lin. Ed-Join: an efficient algorithm for similarity joins with edit distance constraints. *PVLDB*, 1(1):933–944, 2008.
29. Chuan Xiao, Wei Wang, Xuemin Lin, and Jeffrey Xu Yu. Efficient similarity joins for near duplicate detection. In *WWW*, pages 131–140, 2008.