SAIM – One Step Closer to Zero-Configuration Link Discovery

Klaus Lyko, Konrad Höffner, René Speck, Axel-Cyrille Ngonga Ngomo, and Jens Lehmann

Universität Leipzig, Postfach 100920, 04009 Leipzig, Germany
`{lastname}@informatik.uni-leipzig.de`
`http://aksw.org/projects/saim`

Abstract. Link discovery plays a central role in the implementation of the Linked Data vision. In this demo paper, we present SAIM, a tool that aims to support users during the creation of high-quality link specifications. The tool implements a simple but effective workflow to creating initial link specifications. In addition, SAIM implements a variety of state-of-the-art machine-learning algorithms for unsupervised, semi-supervised and supervised instance matching on structured data. We demonstrate SAIM by using benchmark data such as the OAEI datasets.

Keywords: Interlinking, Machine Learning, Data Integration

1 Introduction

Links between instances are of central importance for a large number of tasks such as data integration, federated querying and knowledge retrieval as often pointed out in the literature [1]. Two main problems arise when trying to discover links between data sets or deduplicate data sets. First, naive solutions to Link Discovery (LD) need to compare all resources in the source dataset with all resources in the target dataset and, thus, have quadratic time complexity. Consequently, naive approaches are impractical when computing links across large datasets such as DBpedia [2][1] or Yago[2]. Time-efficient algorithms and frameworks such as LIMES [4] and SILK [3] have been developed to reduce the number of comparisons which need to be made between resources. While these approaches achieve practicable runtimes even on large datasets, they do not guarantee the quality of the links that are returned by LD frameworks. Addressing this second problem of LD demands the development of techniques that can compute accurate *link specifications* for deciding whether two resources should be linked. Both supervised (e.g., [6,7]) and unsupervised machine-learning approaches (e.g., [8]) have been proposed to achieve this goal.

[1] `http://dbpedia.org`
[2] `http://www.mpi-inf.mpg.de/yago-naga/yago/`
[3] `https://www.assembla.com/spaces/silk/`

SAIM [4] encompasses solutions for both problems within a simple interface which implements a flexible workflow. The tool relies on algorithms implemented in LIMES [5], which have been shown to outperform the state of the art in previous work w.r.t. time efficiency [3]. In addition to allowing expert users to create specifications manually, SAIM implements supervised and unsupervised learning algorithms including extensions of EAGLE [6] and the novel COALA [7] approach (presented at the same conference), which have been shown to lead to high-quality specifications. By these means, SAIM can support domain experts and lay users during the creation of link specifications. Moreover, it implements the time-efficient algorithms for class and property matching algorithms proposed in [5]. SAIM goes beyond existing interfaces for link discovery (e.g., SILK Workbench [6]) by supporting several self-configuration algorithms that allow the automatic creation of link specifications. In the following, we present the workflow underlying SAIM and then focus on the content of the SAIM demonstration.

2 SAIM

The workflow underlying SAIM consists of four main steps: (1) data selection, (2) schema matching, (3) creation of the specification and (4) execution of the specification. In the following, we explain how SAIM supports each of these steps.

2.1 Data Selection

SAIM allows users to specify SPARQL endpoints or local RDF files (for users with logins) as data sources. SPARQL endpoints are specified by stating the URL of the endpoint and (if necessary) the graph from which the data is to read. Moreover, each endpoint can be given a name. Our software signalizes to its user whether the endpoint he selected is alive by issuing a simple SPARQL query to the specified endpoint. Moreover, it provides a list of commonly used endpoints such as DBpedia [7]. Local RDF files can be in any of the serialization formats supported by the Jena Framework [8] on which SAIM relies. In addition, the tool supports using data stored as CSV files. In the latter case, the data is converted to RDF on the fly by using the strings contained in each column of the first row as property labels and the elements of the first column as URI for the resources.

2.2 Schema Matching

Our approach relies on simple yet the time-efficient schema matching algorithms presented for matching classes and properties (see Figure 1). The user can choose

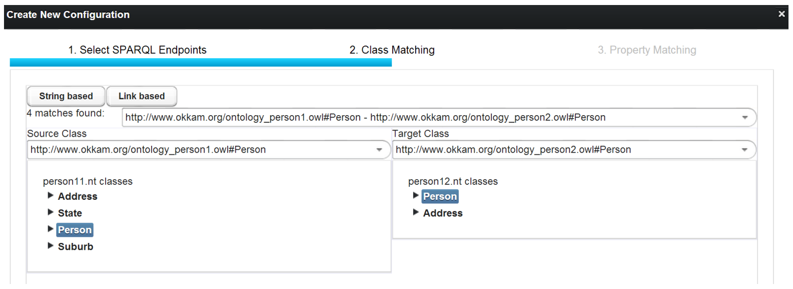
[4] SAIM stands for (Semi-)Automatic Instance Matcher and is pronounced like "same". All information to the project including a demo and a screencast can be found at `http://aksw.org/projects/saim`.

[5] See `http://limes.sf.net`.

[6] `https://www.assembla.com/spaces/silk/wiki/Silk_Workbench`

[7] `http://dbpedia.org/sparql`

[8] `http://jena.apache.org/`



4 matches found: http://www.okkam.org/ontology_person1.owl#Person - http://www.okkam.org/ontology_person2.owl#Person

Source Class: http://www.okkam.org/ontology_person1.owl#Person

Target Class: http://www.okkam.org/ontology_person2.owl#Person

person11.nt classes
- Address
- State
- Person
- Suburb

person12.nt classes
- Person
- Address

Fig. 1: Schema Matching step in SAIM.

between intensional matching (`String based` button) and a matching approach based on stable marriage on links (`Link based` button, see [5] for more details). Per default, SAIM compute the string-based matching between the class labels by using the trigram similarity and return a sorted list of matching classes. We chose this approach because of its time-efficiency. The user can either choose the stable-marriage-based approach or navigate through the schemas of the dataset to perform the schema matching manually. Note that SAIM implements fallback solutions to be able to display the schemas of datasets with incomplete ontologies. For example, if no statement of the form `?x a rdf:Class` is found in the dataset, our approach falls back to retrieving all distinct `?x` such that triples of the form `?y a ?x` is contained in the dataset.

2.3 Creation of Specifications

The creation of specifications is the most involved part of SAIM's workflow. Once the schema matching has been carried out, the user is presented with SAIM's main window. Initially, this window contains an *output node*. On the left, the expert user can choose between the different properties, several similarity and distance

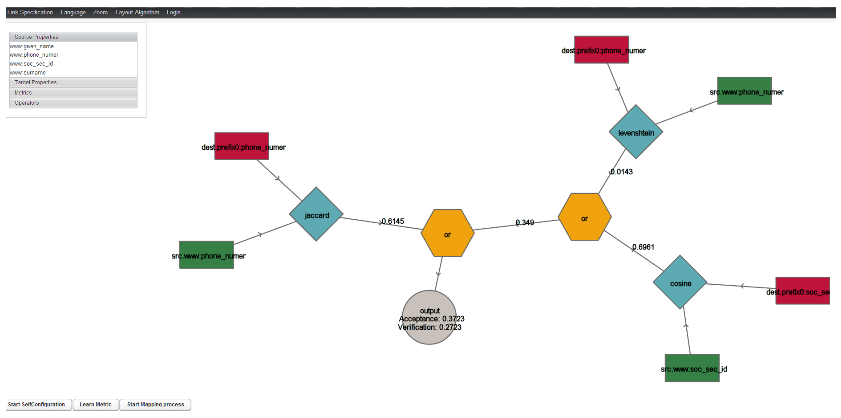
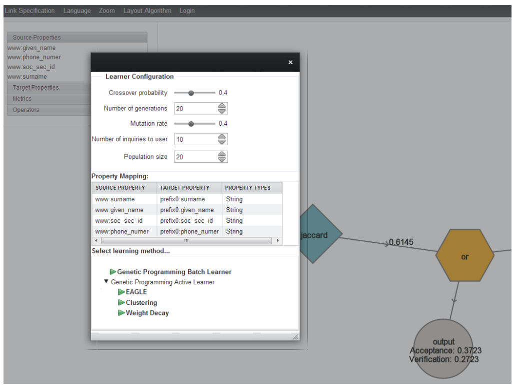


Fig. 2: SAIM specification window.



Learner Configuration

Crossover probability: 0,4
Number of generations: 20
Mutation rate: 0,4
Number of inquiries to user: 10
Population size: 20

Property Mapping:

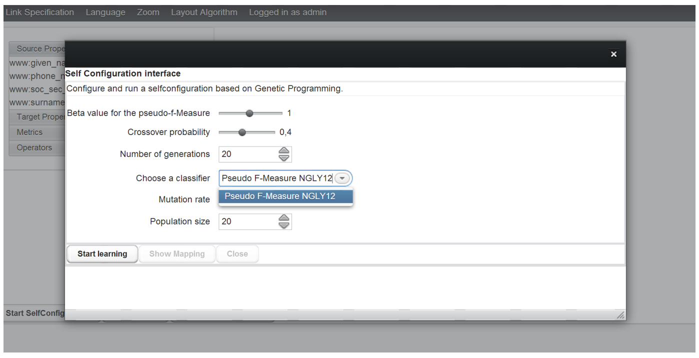
SOURCE PROPERTY	TARGET PROPERTY	PROPERTY TYPES
www:surname	prefix0:surname	String
www:given_name	prefix0:given_name	String
www:soc_sec_id	prefix0:soc_sec_id	String
www:phone_numer	prefix0:phone_numer	String

Select learning method...

- Genetic Programming Batch Learner
- Genetic Programming Active Learner
  - EAGLE
  - Clustering
  - Weight Decay

Fig. 3: Selection of algorithms.

measures (incl. Levenshtein, Trigrams, Cosine) as well as operators (incl. AND, OR, MAX, MIN) to combine these metrics to a single specification manually (see Figure 2). The user can also choose the `Learn metric` button instead, which allows the user to select between several machine-learning algorithms for link specification learning including EAGLE [6] and its extensions in COALA [7] (Figure 3). After choosing these algorithms, the user is presented the most informative positive and negative examples and can choose whether they are matches or non-matches. SAIM supports the user in this process by allowing him to view the values of the properties of the resources that are part of the match or to dereference their URIs. SAIM also enables lay users to create link specification by offering a *self-configuration* mode. In the corresponding window (see Figure 4), SAIM allows the user to choose which algorithm to use and whether the specification should be tuned towards precision or recall (the default setting being that both are equally important). Once the user has selected a configuration, SAIM runs unsupervised machine-learning algorithms based on EAGLE or RAVEN and returns the specification that maximize a selected pseudo-F-measure. The specification shown in Figure 2 was learned fully automatically by the unsupervised version EAGLE.



Self Configuration interface

Configure and run a selfconfiguration based on Genetic Programming.

Beta value for the pseudo-f-Measure: 1
Crossover probability: 0,4
Number of generations: 20
Choose a classifier: Pseudo F-Measure NGLY12
Pseudo F-Measure NGLY12
Mutation rate
Population size: 20

Fig. 4: Specification of self-configuration in SAIM.

3 Demonstration

The goal of the demonstration will be to show the whole workflow described above from the datasets to the export of the resulting specification and links. We will begin by showing how SAIM deals with data in SPARQL endpoints and with local datasets. Thereafter, we will present and explain how SAIM suggests class and property matchings to the user.We will especially explain the hierarchy of fallback solutions that SAIM employs to generate both an overview of the class structure and the corresponding class matching as well as how it uses extensional and intensional schema matching approaches for both class and property matching. In a third step, we will then showcase the approaches implemented in SAIM. We will begin by showing how the expert user can use SAIM to create link specifications manually. Thereafter, we will present how domain experts can employ the active learning algorithms EAGLE and COALA to learn and refine link specifications iteratively. Then, we show how lay users can use unsupervised machine learning to have SAIM detect a high-quality link specification for them. Finally, we will demonstrate how the results of the specification process (i.e., the link specification and the resulting mappings) can be downloaded from SAIM for use in further applications. Throughout the demonstration, we will employ benchmark datasets such as those provided by the OAEI challenges [9].

4 Conclusions and Future Work

We present SAIM, an interface for the creation of high-quality link specifications. SAIM represents a further step towards the vision of zero-configuration link discovery as it allows users to create such specifications with minimal effort. In future work, we will extend SAIM with more algorithms for learning link specifications and aim to achieve our vision of easy and effective link discovery.

References

1. S. Auer, J. Lehmann, and A.-C. Ngonga Ngomo. Introduction to linked data and its lifecycle on the web. In *Reasoning Web*, pages 1–75, 2011.
2. M. Morsey, J. Lehmann, S. Auer, C. Stadler, and S. Hellmann. DBpedia and the live extraction of structured data from wikipedia. *Program: electronic library and information systems*, 46:27, 2012.
3. A.-C. Ngonga Ngomo. On link discovery using a hybrid approach. *Journal on Data Semantics*, 1:203 – 217, December 2012.
4. A.-C. Ngonga Ngomo and S. Auer. LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data. In *Proceedings of IJCAI*, 2011.
5. A.-C. Ngonga Ngomo, J. Lehmann, S. Auer, and K. Höffner. RAVEN – Active Learning of Link Specifications. In *Proceedings of OM@ISWC*, volume 814, 2011.
6. A.-C. Ngonga Ngomo and K. Lyko. Eagle: Efficient active learning of link specifications using genetic programming. In *Proceedings of ESWC*, 2012.
7. A.-C. Ngonga Ngomo, K. Lyko, and V. Christen. Coala – correlation-aware active learning of link specifications. In *Proceedings of ESWC*, 2013.
8. A. Nikolov, M. D'Aquin, and E. Motta. Unsupervised learning of data linking configuration. In *Proceedings of ESWC*, 2012.

[9] `http://oaei.ontologymatching.org/`