

LIMESWEBUI– Link Discovery Made Simple★

Mohamed Ahmed Sherif, Pestryakova Svetlana, Kevin Dreßler, and Axel-Cyrille Ngonga Ngomo

Paderborn University, Data Science Group, Pohlweg 51, D-33098 Paderborn, Germany
E-mail: {firstName.lastName}@upb.de

Abstract. In this paper we present LIMESWEBUI, our web interface of LIMES. LIMES, the *Link Discovery Framework for Metric Spaces*, is a framework for discovering links between entities contained in Linked Data sources. LIMESWEBUI assists the end user during the link discovery process. By representing the link specifications (LS) as interlocking blocks, our interface eases the manual creation of links for users who already know which LS they would like to execute. However, most users do not know which LS suits their linking task best and therefore need help throughout this process. Hence, our interface provides wizards which allow the easy configuration of many link discovery machine learning algorithms, that does not require the user to enter a manual LS. We evaluate the usability of the interface by using the standard system usability scale questionnaire. Our overall usability score of 76.5 suggests that the online interface is consistent, easy to use, and the various functions of the system are well integrated.

1 Introduction

Establishing links between knowledge bases is one of the key steps of the Linked Data publication process.¹ A plethora of approaches has thus been devised to support this process [2]. LIMES² was designed as a declarative framework to address two main challenges of *time-efficiency* and *accuracy*.

The formal specification of Link Discovery (LD) adopted herein is akin to that proposed in [3]. Given two (not necessarily distinct) sets S resp. T of source resp. target resources as well as a relation R , the goal of LD is to find the set $M = \{(s, t) \in S \times T : R(s, t)\}$ of pairs $(s, t) \in S \times T$ such that $R(s, t)$. In most cases, computing M is a non-trivial task. Hence, in LIMES [3], we aim to approximate M by computing the *mapping* $M' = \{(s, t) \in S \times T : \sigma(s, t) \geq \theta\}$, where σ is a similarity function and θ is a similarity threshold. For example, one can configure LIMES to deduplicate census records by comparing the dates of birth, family names and given names of persons.

We call the equation which specifies M' a *link specification* (short LS). The grammar we use for representing LSs in LIMES is based on set semantics. This grammar assumes that LSs consist of two types of atomic components: (i) *similarity measures* m ,

★ Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

¹ <http://www.w3.org/DesignIssues/LinkedData.html>

² Source code as well as manual are available at <https://github.com/dice-group/LIMES>

which allow the comparison of property values or portions of the concise bound description of 2 resources and (ii) *operators* ω , which can be used to combine these similarities into more complex specifications. Without loss of generality, we define an *atomic similarity measure* a as a function $a : S \times T \rightarrow [0, 1]$. An example of an atomic similarity measure is the edit similarity dubbed `edit`³. Every atomic measure is a measure. We define a *filter* as a function $f(m, \theta)$. We call a LS *atomic* when it consists of exactly one filtering function. A complex LS can be obtained by combining two specifications through an *operator* such as AND, OR and MINUS.

2 LIMES Web User Interface

LIMES Web user interface (LIMESWEBUI⁴) is our novel tool to ease the usage of LIMES. The aim of LIMESWEBUI is to aid our users throughout the configuration as well as the execution process of LIMES. LIMESWEBUI consists of the following main components:

1. *The prefixes component* consists of the set of name spaces to be used through the rest of the LIMES configuration process. In most cases, our interface is able to automatically find the common prefixes⁵. In case the user wants to add a custom prefix, (s)he still can type the prefix manually.
2. In the *data source and target components*, the user can define the source and target sets of resources to be linked. In particular, the `Endpoint` field provide a list of common endpoints, where the user can select the one that provide the datasets (s)he interested on. Still, the user can manually input other endpoints if not in the provided list. Then, in the `Restriction` field the user can select the class within the dataset to retrieve its instances for linking. Note that, our web UI is able to retrieve all the classes automatically for the user via a SPARQL query.
3. *The manual metric component*. Once the user chooses the source/target datasets' endpoints and classes, LIMESWEBUI will automatically load the respective properties within the source/target instances. The user can either use our interface either to build a manual LS or to configure one of our machine learning algorithms to learn it. For building the *manual metric*, our interface provides a workspace that uses a custom version of the *Blockly API*⁶, where the user can simply drag and drop the LS elements from the toolbox. Using our work-space, the user can define complex LS which consists of multiple measures, operators and preprocessing functions. Figure 1 shows an example of a complex LS in our work-space. Note that, our interface is able to save/load the work-space for later use.
4. *The machine learning (ML) component* consists of: (1) The ML algorithm name to be used. e.g., WOMBAT [5] and EAGLE [4]. (2) The type of the ML algorithm. i.e., supervised batch, supervised active or unsupervised. (3) A list of parameters for fine tuning the currently selected ML algorithm. Note that, the list of the default parameters of the currently selected ML algorithm will be loaded initially by our web UI.

³ We define the edit similarity of two strings s and t as $(1 + \text{Levenshtein}(s, t))^{-1}$.

⁴ Publicly accessible at <http://limes.aksw.org>

⁵ From <https://prefix.cc/context>

⁶ <https://developers.google.com/blockly>

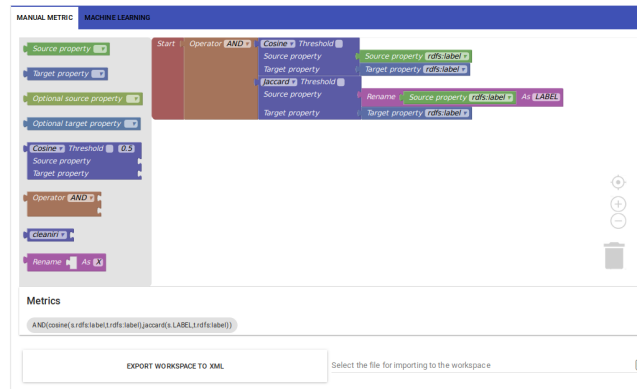


Fig. 1. Manual metric example.

Figure 2 shows an example of using LIMESWEBUI for configuring the unsupervised version of the WOMBAT-simple ML algorithm.

5. In the *acceptance and review components*, the user can define the acceptance and review thresholds. i.e., the similarity threshold by which a link should be considered by LIMES as accepted or to-be-reviewed link. Hence, LIMES save such a link into either the accepted/review file.
6. Using the *output component*, the user can choose an output serialization such as turtle, n-triples, tab separated values and comma separated values.

Finally, the user is able to display the generated XML configuration file, save or even run it. LIMESWEBUI assigns a unique *execution ID (EID)* for each linking task. In case an execution takes time, the user can simply close his browser and check for the status of his task later using his/her EID. Once an execution is done, the resulted accepted and to-be-reviewed links are stored in our server, where the user can retrieve them at any time using the respective EID.

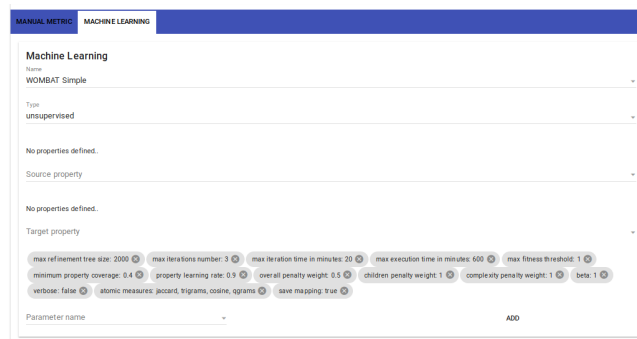


Fig. 2. Using LIMESWEBUI for configuring the machine learning algorithm WOMBAT.

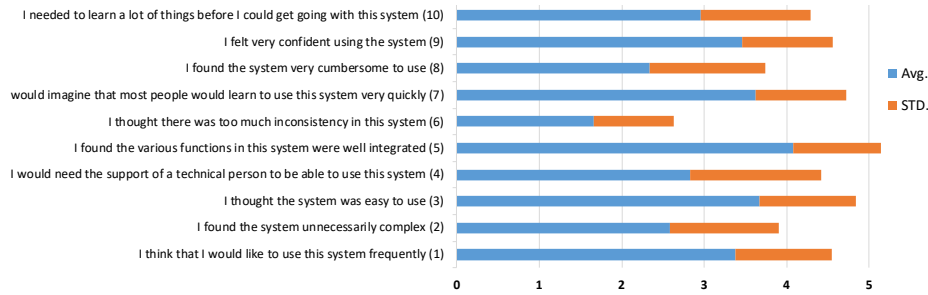


Fig. 3. Result of usability evaluation using SUS questionnaire.

Evaluation. To assess the usability of our system, we used the standard System Usability Scale (SUS) [1] questionnaire⁷. The survey was posted through the mailing lists of the DICE research group (<https://dice-research.org>) and was filled by 24 users. The results of our SUS shown in Figure 3. We achieved a mean usability score of 76.5 indicating a high level of usability according to the SUS score. The responses to question 1 suggests that our system is adequate for frequent use (average score to question 1 = 3.4 ± 1.2) by users of all types. The responses to question 3 (average score 3.7 ± 1.2) suggests that the interface is easy to use and the responses to question 5 indicates that the various functions are well integrated (average score 4.1 ± 1.1). However, the response to question 10 (average score 2.9 ± 1.3) indicates that users need to learn some basic concepts before they can use the system effectively.

Acknowledgment. This work has been supported by the BMVI projects LIMBO (GA no. 19F2029C) and OPAL (GA no. 19F2028A), Eurostars Project SAGE (GA no. E/10882) as well as the H2020 project SLIPO (GA no.731581).

References

1. James R Lewis and Jeff Sauro. The factor structure of the system usability scale. In *International conference on human centered design*, pages 94–103. Springer, 2009.
2. Markus Nentwig, Michael Hartung, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. A survey of current link discovery frameworks. *Semantic Web*, 8(3):419–436, 2017.
3. Axel-Cyrille Ngonga Ngomo. On link discovery using a hybrid approach. *J. Data Semantics*, 1(4):203–217, 2012.
4. Axel-Cyrille Ngonga Ngomo and Klaus Lyko. EAGLE: efficient active learning of link specifications using genetic programming. In *Extended Semantic Web Conference 2012*.
5. Mohamed Ahmed Sherif, Axel-Cyrille Ngonga Ngomo, and Jens Lehmann. WOMBAT - A Generalization Approach for Automatic Link Discovery. In *14th ESWC, 2017*.

⁷ Our survey is available at: <https://forms.gle/ayZE7KVwH4VhTULk6>