DLinker Results for OAEI 2022*

Bill Gates Happi Happi[1,*,†], Géraud Fokou Pelap[2,†], Danai Symeonidou[3,†] and Pierre Larmande[1,†]

[1]DIADE, University of Montpellier, IRD, CIRAD, 911 Av. Agropolis, 34394 Montpellier, France
[2]University of Dschang, Dschang ville, Cameroon
[3]INRAE, SupAgro, UMR MISTEA, University of Montpellier, 2 place pierre viala 34060 Montpellier, France

Abstract

DLinker is a system for matching instances of two RDF data sources. Its performance is mainly based on the deep comparison of literals. The main comparison algorithm is based on the search for the longest common subsequence (LCS) present in the literals. The validation of the similarity between two literals is performed by a mathematical formula. This formula computes the confidence percentage of the similarity between the literals and compares it with a threshold provided as input among the expected hyperparameters. To validate the similar instances, our system validates only the instances that have reached the value of the acceptation threshold provided in the list of required hyperparameters. The current version focuses on the processing of strings on the spot without taking into account synonyms to make its decisions. This is DLinker's first participation in the OAEI campaign on two principal tracks (SPIMBENCH and SPATIAL) with 9 challenges. DLinker demonstrated its ability to process different data with good accuracy and in a very short time. Additionally, in the context of the SPATIAL challenge DLinker has outperformed the state of the art finishing first with the shortest time. Overall, DLinker exposes different strengths and weaknesses that are discussed in this work.

Keywords

Instance Matching, Syntactic Similarity, Data Linking Algorithm, Data Processing, Synonyms

1. Presentation of DLinker

1.1. General Statement

DLinker is a generic instance matching system. Its performance is mainly based on a recursive algorithm which compares literals using the longest common subsequence (LCS[1]). DLinker performs the matching in two steps. First, during the training step, the algorithm learns from the hyperparameters to optimize the prediction of pairs of similar instances. Finally the tool exploits the model tuning for the prediction step. DLinker participated in 2 tracks composed of 9

The 17th International Workshop on Ontology Matching, collocated with the 21th International Semantic Web Conference ISWC-2022, October 23rd, 2022, Hangzhou, China
*Corresponding author.
†These authors contributed equally.
bill.happi@ird.fr;billhappi@gmail.com (B. G. H. Happi); geraud.fokou@univ-dschang.org (G. F. Pelap); danai.symeonidou@inrae.fr (D. Symeonidou); pierre.larmande@ird.fr (P. Larmande)
... (B. G. H. Happi); ... (G. F. Pelap); ... (D. Symeonidou); ... (P. Larmande)
0000-0003-1199-7726 (B. G. H. Happi); 0000-0002-4024-1154 (G. F. Pelap); ... (D. Symeonidou); 0000-0002-2923-9790 (P. Larmande)

challenges during the OAEI campaign. The tool achieved good results during the SPATIAL track, as it was the best on temporal and accuracy performances on small and large scale EQUALS and OVERLAPS topologicals tasks for TOMTOM and SPATEN. During the SPIMBENCH track, we participated on small datasets while providing good accuracy in a considerable time. The main similarity measure of the tool is based on the Longest Common Sub-Sequence (LCS[1]) algorithm, which is seen as a deep variant of this one, imposing a very large capacity for various applications. The temporal factor of DLinker is justified by the parallelization of the pairs of literal objects to be compared. DLinker was wrapped by Hobbit Framework [2].

1.2. System overview

Initially, the theoretical formalization of DLinker consisted in setting up a multilingual data binding tool during data processing. However, the current version performs this task properly when the sources to be linked belong to the same language. As shown in figure 1, DLinker matches only instances. We give as inputs two data sources to be linked in different formats, linked or not, to produce as output a set of alignments in a suitable data format.

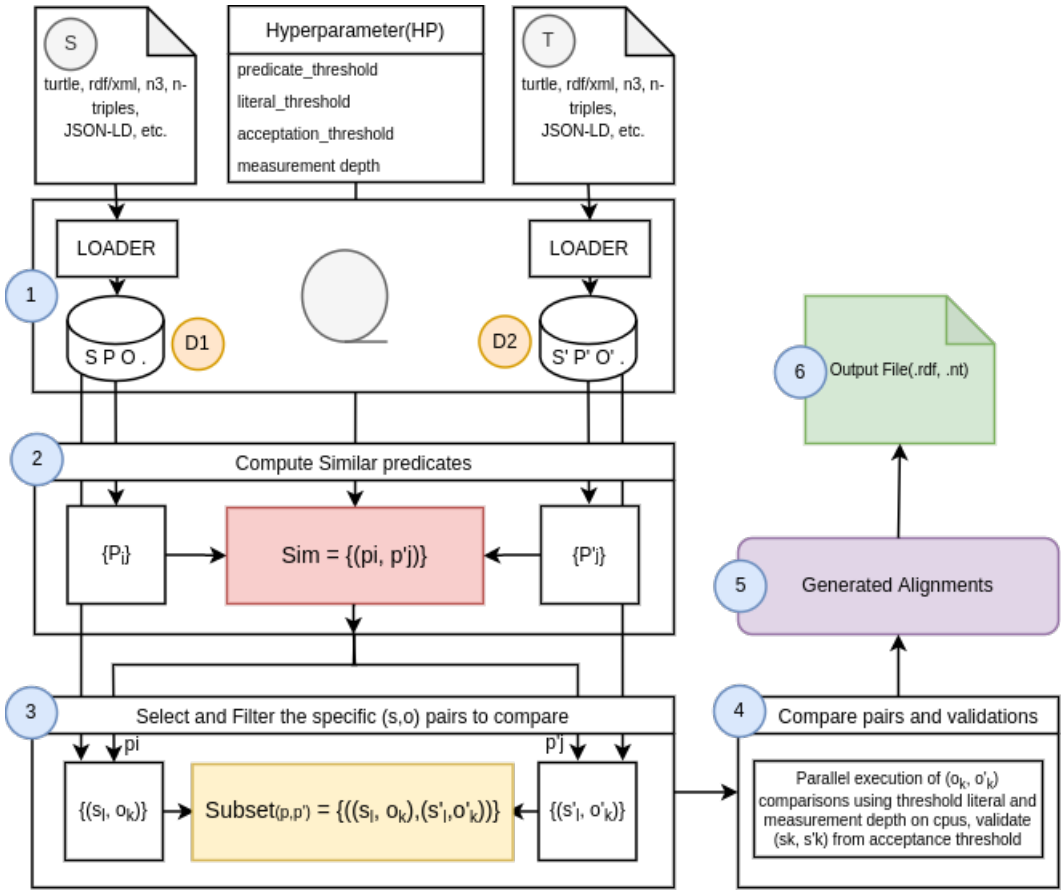

Figure 1: Overview of DLinker's pipeline

1. Loading: loads the input files and returns an RDF data graph in the form of triples. Also loads the hyperparameters (HP) from the data training. Let us give a brief description of these hyperparameters:
    - predicate threshold: represents the numerical value of similarity recognition between two predicates;
    - literal thresold: represents the numerical value of similarity recognition between two literals;
    - acceptation threshold: represents the minimal number of pairs of similar objects belonging to the instances that can be linked;
    - measurement depth: represents the number of times we should search for the longest common subsequences between the pairs of objects of the instances. It can also be seen as the depth of recursion during the search for similar literals;
2. Compute Similar predicates: Retrieves the unique predicates of each graph and returns pairs of similar or complementary predicates existing between them using the threshold of the predicate;
3. Select and Filter the specific (s, o) pairs to compare: First, a selection of each pair (s, o) (respectively (s', o')) in each of the two data sources from the predicate pairs (pi, p'j) (similar) is performed. Then, a construction of sub-lists of cartesian products ((s, o),(s', o')) for the (s, o) (respectively (s',o')) of these predicates, we proceed to the step of reducing the number of unnecessary comparisons between the literal objects of the pairs of the sub-sets by calculating the completeness or proximity ratio between them;
4. Compare pairs and validations: Here we realize the comparison of the literal objects associated to the couples ((s, o),(s', o')). These comparisons are validated using the threshold of the literal before or after the similarity search depth (measurement depth hyperparameter). The validation is performed once the Acceptation Threshold (AT) is reached after several successful object comparisons. To ensure proper validation of comparison peer topics, we perform a hash of the concatenation of the topics which are then stored as <key, value>. The value of the key increments to reach AT if the associated object comparisons are positive;
5. Generate Alignment: The process of comparisons being parallel, a list of pairs (s, s', relation, 1) is filled after validations by each sub-processing of the subset of pairs of ((s, o), (s', o')) based on the acceptation threshold hyperparameter.
6. Output file: the alignments are generated according to the format expected in output, for the moment we provide ".rdf" and ".nt".

2. Results

This section describes the results of the DLinker system on two tracks namely: SPATIAL and SPIMBENCH. The evaluation was executed on a Linux virtual machine with 256 GB of RAM and 32 vCPUs (2.4 GHz) processors. The table below presents the different hyperparameters that were used to obtain the following results:

Table 1
Hyperparameters of DLinker for each track.

Tracks	Predicate Threshold	Literal threshold	Acceptation threshold	Measurement depth
Spatial	1.0	0.3	1	0
Spimbench	1.0	0.8	1	1

2.1. SPATIAL Data

This track[1] concerns data that are part of SPATIAL data management systems and that store topological relationships in the form of SPATIAL resources that can be linked together. These SPATIAL resources are described from a large information set such as LinkedGeoData. These data are sent from a SPATIAL benchmark generator[2]. This Benchmark supports several topological relations (Equals, Disjoint, Touches, Contains/Within, Covers/CoveredBy, Intersects, Crosses, Overlaps). This SPATIAL generator contains three data generators (TomTom, Spaten and DEBS).

1. Spaten is an open-source configurable spatio-temporal and textual dataset generator, that can produce large volumes of data based on realistic user behavior. Spaten extracts GPS traces from realistic routes utilizing the Google Maps API, and combines them with real POIs and relevant user comments crawled from TripAdvisor. Spaten publicly offers GB-size datasets with millions of check-ins and GPS traces;
2. TomTom provides a Synthetic Trace Generator developed in the context of the HOBBIT Project, that facilitates the creation of an arbitrary volume of data from statistical descriptions of vehicle traffic. More specifically, it generates traces, with a trace being a list of (longitude, latitude) pairs recorded by one device (phone, car, etc.) throughout one day. TomTom was the only data generator in the first version of SPgen;
3. DEBS provides a selection of AIS data collected from the MarineTraffic coastal network. It has been used for the EU H2020 Research Project BigDataOcean and the ACM DEBS Grand Challenge 2018.

The results below are available at this address: https://hobbit-project.github.io/OAEI_2022.html.

2.1.1. Evaluation on SandBox LINESTRINGS - LINSTRINGS

- Under the EQUALS topological relationship, DLinker terminated in 1667ms for Spaten and 10487ms for TomTom providing the highest accuracy and smallest overall time;
- Under the OVERLAPS topological relationship, DLinker finished in 3236ms for Spaten and 3087ms for TomTom providing the highest accuracy and lowest overall time. The summary can be found in the figure 2.

[1]https://project-hobbit.eu/hobbit-spatial-benchmark-v2-0/
[2]https://github.com/hobbit-project/SpatialBenchmark

2.1.2. Evaluation on MainBox LINESTRINGS - LINSTRINGS

- Under the OVERLAPS topological relationship, DLinker terminated in 2026ms for Spaten and 37072ms for TomTom providing the highest accuracy and smallest overall time;
- Under the OVERLAPS topological relationship, DLinker terminated in 2547ms for Spaten and 5458ms for TomTom providing the highest accuracy and smallest global time. The summary can be found in the figure 3.

2.2. SPIMBENCH Data

The datasets [3] in this track are produced using SPIMBENCH benchmark generator [4] with the aim to generate descriptions of the same entity where valuebased, structure-based and semantics-aware transformations are employed on a source dataset in order to create the target dataset(s). The goal of the SPIMBENCH task is to determine when two instances describe the same Creative Work. A dataset is composed of a Tbox (contains the ontology and the instances) and a corresponding Abox (contains only the instances). The datasets share almost the same ontology (with some difference in the properties' level, due to the structure-based transformations). What we expect from participants. Participants are requested to match instances in the source dataset (Tbox1) against the instances of the target dataset (Tbox2). The task goal is to produce a set of mappings between the pairs of matching instances that refer to the same real-world entity. An instance in the source dataset (Tbox1) can have none or one matching counterparts in the target dataset (Tbox2). Note that only instances of Creative Work are to be mapped in this task[3]. The instances of the other classes appearing in the sources are used to examine if the matching systems take into account RDFS [5] and OWL [6] constructs in order to discover correspondences between instances that can be found only by considering schema information. After processing 380 instances (10000 triples) for each file (source and target), we obtained the scores presented in the following section.

2.2.1. Evaluation on small data set

DLinker participated in the evaluation of small data set and finished in 15555ms with an accuracy of **0.791** as shown in the table 2 below.

Table 2
Results of DLinker on SPIMBENCH track 2022.

Matcher	Precision	Recall	F-measure	Runtime(ms)
DLinker	**0.791**	0.632	**0.703**	15555
LogMap	0.938	0.763	0.841	6116

[3]https://hobbit-project.github.io/OAEI_2022.html

3. General comments and conclusions

Comments on the results. DLinker was ranked first on the SPACIAL track and second on the SPIMBENCH track. Its fast processing time is due to the parallelization of the processing of literal comparisons in the analysis of instances based on the LCS and its high precision. The high accuracy can be guaranteed on so-called in-place comparisons and not on those requiring synonym consideration. The non-participation of synonyms in the instance matching process is the main weakness of DLinker and prevents it from correctly projecting itself in the ontology alignment.
Space for improvement in our system. Although DLinker seems stable on several datasets already evaluated so far, we plan to make it even more robust and efficient for other challenges. We have seen many exciting possibilities for future work. For example, we intend to implement multilingual functionality to link two data sources of different languages and integrate the consideration of synonyms during comparisons. We also plan to realize an automatic hyperparameters generator from the training datasets.

Acknowledgments

This work was supported by the IRD DIADE UNIT and the FOOSIN project. We also thank the organisers of the OAEI evaluation campaign for providing test data and infrastructure.

References

[1] C. Bepery, S. Abdullah-Al-Mamun, M. Rahman, Computing a longest common subsequence for multiple sequences, 2015. doi:10.1109/EICT.2015.7391933.
[2] M. Röder, D. Kuchelev, A.-C. Ngonga Ngomo, Hobbit: A platform for benchmarking big linked data, Data Science 3 (2019) 1–21. doi:10.3233/DS-190021.
[3] E. Jiménez-Ruiz, T. Saveta, O. Šváb Zamazal, S. Hertling, M. Röder, I. Fundulaki, A.-C. Ngonga Ngomo, M. Sherif, A. Annane, Z. Bellahsene, S. Ben Yahia, G. Diallo, D. Faria, M. Kachroudi, A. Khiat, P. Lambrix, H. Li, M. Mackeprang, M. Mohammadi, C. Trojahn, Introducing the hobbit platform into the ontology alignment evaluation campaign, 2018.
[4] T. Saveta, E. Daskalaki, G. Flouris, I. Fundulaki, M. Herschel, A.-C. Ngonga Ngomo, Pushing the limits of instance matching systems: A semantics-aware benchmark for linked data, 2015, pp. 105–106. doi:10.1145/2740908.2742729.
[5] J. Pan, I. Horrocks, Rdfs (fa) and rdf mt: Two semantics for rdfs, 2003. doi:10.1007/978-3-540-39718-2_3.
[6] A. Souhaib, K. Mohamed, k. e. el kadiri, Ontology Alignment OWL-Lite, 2012. doi:10.5772/28619.

A. Online Resources

The sources for DLinker python and Java version are available via:

- Python Version: https://github.com/BillGates98/DLinker,
- Java version: https://github.com/BillGates98/dlinker-adapter,

B. Diagrams

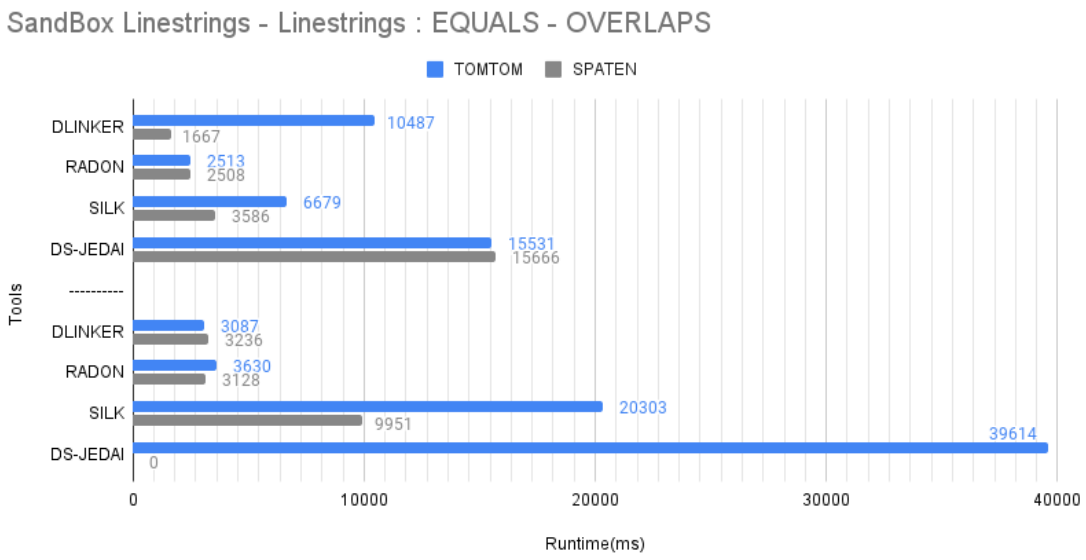

Figure 2: Results of DLinker on SandBox for TOMTOM and SPATEN

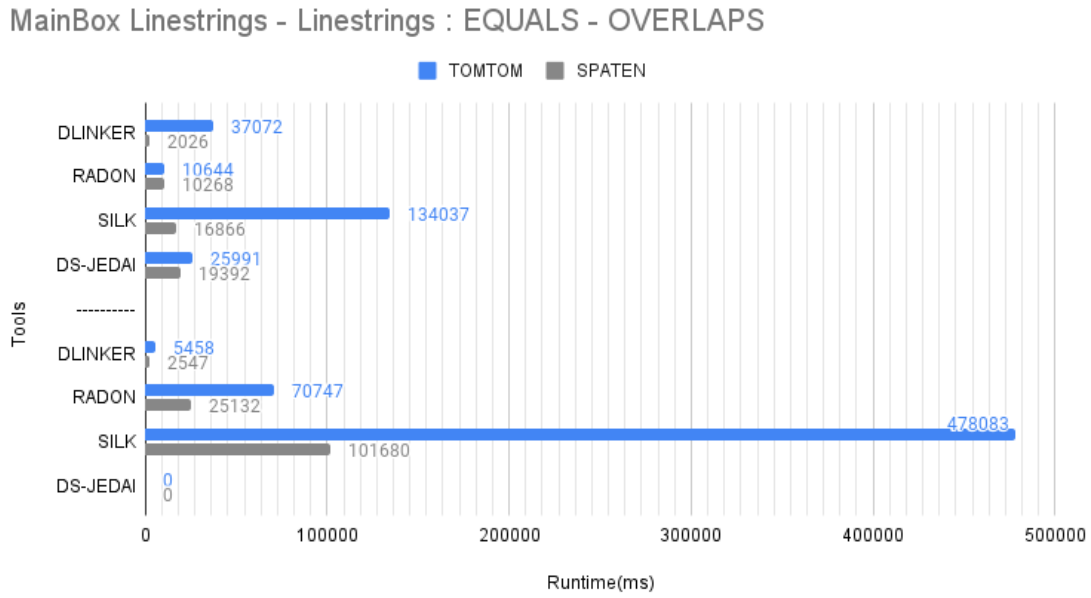

Figure 3: Results of DLinker on MainBox for TOMTOM and SPATEN.