Data Link Discovery Frameworks for Biomedical Linked Data: A comprehensive study

1st Houssein Dhayne
Faculty of Engineering, ESIB
Saint Joseph University
Beirut, Lebanon
houssein.dhayne@net.usj.edu.lb

2nd Hanan Farhat
Faculty of Engineering, ESIB
Saint Joseph University
Beirut, Lebanon
hanan.farhat@net.usj.edu.lb

3rd Rima Kilany
Faculty of Engineering, ESIB
Saint Joseph University
Beirut, Lebanon
rima.kilany@usj.edu.lb

***Abstract*—Data discovery, linking and integration techniques are of great importance for big data variety challenge. Linked Open Data (LOD) and Semantic Web technologies have worked as a driver to address this challenge. However, until 2015, the linkage of triples of LOD has increased to 40%, of which only 3% of overall triples are links between different datasets. Today, with the increasing amount of available LOD datasets, 9671 datasets compose the LOD, the need to link them together is becoming vital. Links are usually generated, or discovered, by specific frameworks such as SILK and LIMES, which are two of the most effective tools in this domain. They apply instance matching rather than ontology matching, and support active learning. They both have their drawbacks and their advantages, which makes it hard to disregard one of them. This paper aims to evaluate whether SILK and LIMES are potential options for interlinking large-scale biomedical datasets, comparing the two frameworks at many levels, starting from the general features, reaching the comparison measures, the resulting files, the performance and the effectiveness of the links produced. The conclusions drawn from this work are to be used as a reference for the evaluation of the core differences between SILK and LIMES and therefore for choosing the most suitable tool in a Biomedical context. It can be considered as an opening for future research and enhancements of such frameworks.**

***Index Terms*—Semantic Web, Link Discovery, Biomedical Linked Data, Data Links**

I. Introduction

With the rise of Big Data awareness among biomedical providers, there is a need to harness the techniques of data integration and analytics to create significant value towards aiding the process of care delivery and disease exploration [1], [2]. Among the different dimensions that characterize big data, the variety dimension seems to be the most intriguing one for the Semantic Web and the one where the research community can contribute [3] [4]. Resource Description Framework (RDF) paradigm, published on the Web in accordance with the Linked Data principles and best practices [5] and containing information about genes, proteins, pathways, diseases, and drugs [6], has evolved as a powerful enabler for the transition of the current unstructured data into interlinked Data [7]. For instance, linked data solve the integration of unstructured data by replacing or annotating the data elements, of medical texts or images, with unique identifiers, providing a structured querying of multiple heterogeneous sources [8].

Linked Open Data (LOD) is a set of best practices to publish RDF linked data on the Web in a machine-readable way, with an explicitly defined semantic meaning, linked to other datasets and allowed to be searched for. LOD principles can be summarized in publishing of open, linked and structured data, in non-proprietary formats using URIs. An indexed ready-to-consume crawl of a large portion of LOD (see Fig.1), called LOD-a-lot[1], contains 28,362,198,927 triples, made up of 3,214,347,198 subjects, 1,168,932 predicates, and 3,178,409,386 objects [9]. With this increasing volume of datasets, the name of the Big Linked Data has been appearing in the research terms. Big Linked Data is an instance of Big Data that is the union of big and linked data, where authors in [10] presented their list of characteristics, which was created by unifying the characteristics of Big and Linked Data.

Moreover, until 2015, the linkage of triples of LOD has increased to 40%, of which only 3% of overall triples are links between different datasets (Fig. 1 showing the growth from 2007 to 2016), therefore new problems are arising that require new solutions from the data science community. Wherefore, the importance of Link-Discovery Frameworks, which are responsible for creating the links, has increased, taking into consideration the efficiency and effectiveness. Efficiency is the optimized process run-time, in addition to the execution time of the preceding and the following steps, excluding complex criteria and computations that consume both time and resources. On the other hand, effectiveness is having the resulting evaluated links accurate and complete.

Link-Discovery problem can be defined as a task that takes two datasets as input and produces a set of links between entities of the two datasets as output. In a formal definition, let S(source) and T(target) two sets of RDF instances as well as s and t two instances of S and T respectively, and a similarity threshold $\Theta \in [0, 1]$. Link- Discovery is the process that leads to the discovery of all set of pairs $(s, t) \in S\text{x}T$ that are linked by a relation φ relying on their properties by using a similarity metric ρ. if the value of $\rho(s, t) \geq \Theta$, then the two entities s and t are considered to be linked by φ.

This paper aims to compare two Link-Discovery Frameworks, SILK and LIMES, at many levels starting from the

[1]http://lod-a-lot.lod.labs.vu.nl/

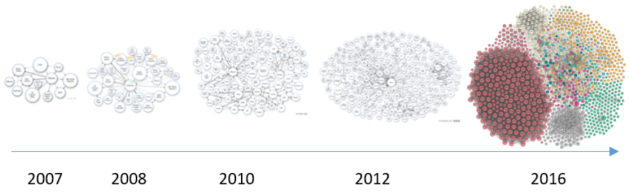


Fig. 1. Growth of the Linking Open Data Cloud from 2007 to 2016

general features, the comparison measures, the resulting files and reaching the performance. In addition, it aims to evaluate the quality of links produced.

The rest of the paper is structured as follows: Section 2 spots the light on comparison studies referring to both frameworks SILK and LIMES. Section 3 shows the general process of link discovery frameworks. In Section 4 we give an overview of SILK and LIMES respectively and compare their general features, while in section 5, the criteria used to perform the comparison experiments is detailed. Section 6 presents the results of our experiments, and finally, section 7 concludes and provides recommendations for future work.

II. Related work

Link Discovery frameworks are divided into two types: Domain Specific (ex: GNAT, specific for music [11]) and Universal frameworks. SILK and LIMES are both Universal Frameworks that aim to generate links between entities of data resources. They have many common features, as well as dissimilar ones that we will detail later.

Many studies compared SILK and LIMES. Nevertheless, the studies covered only the efficiency challenge (run-time) assuming the effectiveness (link quality) is guaranteed. The main two studies comparing the frameworks were published in 2011, where the first compared SILK of version 2 to LIMES of version 0.3.21 [12], and the second compared SILK of version 2.3 to LIMES of version 0.5 [13]. Both studies ended favoring LIMES over SILK speed wise. However, we cannot rely on this result since both frameworks have newer and refactored versions.

An important element that makes the comparison unrighteous in these two studies is the difference in defining comparison thresholds. While -even in older versions- LIMES had the option for specifying the threshold for each operator by itself in addition to the threshold for the aggregated output, in SILK the threshold could only be specified for the output of the aggregation. This per-comparison threshold was recently introduced into the newer versions of SILK.

Even though effectiveness or link quality was out of concern in the latest studies on SILK and LIMES, the implemented tests, whether comparing those two frameworks or evaluating the performance of each alone, have helped develop the criterion to be followed in order to evaluate generated links quality, and deduce which framework, SILK or LIMES, has better impact on effectiveness. Some of those constructive studies are summarized in Table I, where each implemented test is shown in details; the source datasets and target datasets, the source and target classes, the properties compared and the comparison operators used for each test. As a result, it is notable that the choice of the comparison operator is pertinently related to the property to compare.

A study [14] published in 2015 and updated in 2017 has handled the issue of link-discovery frameworks, and compared ten of them using a specified benchmark. The frameworks compared were RiMOM, KnoFuss, AgreementMaker, CODI, SERIMI, LogMap, SLINT+, Zhishi.links, SILK and LIMES. However, these ten tools can be classified as follows: Machine Learning(Involved or Not Involved), and Matching (Ontology Matching or Instance Matching). Seven frameworks appeared to exclude machine learning from the matching process, and support ontology matching. Yet, semantic web has deviated the research from ontology to instance matching gradually considering the heterogeneity of documents and real-world entities, which may appear at many web locations with different descriptions. As the objective of semantic web was to make data more understandable by machines, machine learning set the best example for making use of linked data. Therefore, three frameworks: SILK, LIMES, and KnoFuss, classified as instance matching frameworks including machine learning accessed the interest zone. In contrary to SILK and LIMES, KnoFuss supports only one type of data links which is owl:sameAs, while they support, in addition to owl:sameAs link, other RDF link types such as alternate, start, and next.

Consequently, it would be essential for any comparison study to start by the classification of comparison measures offered by both tools, and this is what will be detailed in Section IV-C. An overview of both SILK and LIMES and a description of their internal architecture is presented in the preceding sections.

III. Link Discovery process

The matching process is the core part of link-discovery. A single comparison can be summarized in a few steps. First, the datasets from which the instances will be picked should be determined (a source and a target). Second, the required classes are to be picked, and then comes the choice of instances that will be compared. Each instance has its description

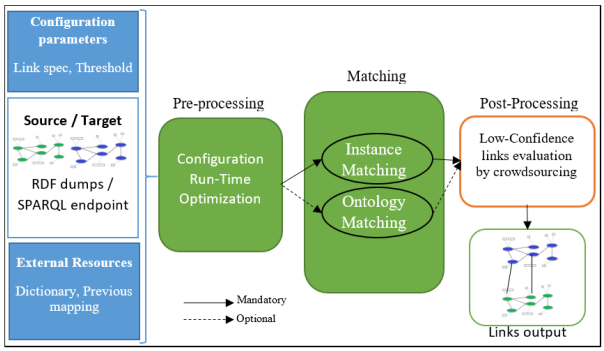


Fig. 2. General Workflow of Link Discovery Frameworks

TABLE I
COMPARISONS APPLIED IN RECENT STUDIES.

	Study	Source Dataset/Class	Target Dataset/Class	Properties	Comparison Operator
1	LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data [12]	Dbpedia/Villages	linkedct/Villages	rdf:type vs. linkedct: condition_name	Levenshtein
		DBpedia	DrugBank	rdfs:label	Levenshtein
		DBpedia	DBpedia	rdfs:label	Levenshtein
		MESH	LinkedCT	rdfs:label	Levenshtein
2	A Time-Efficient Hybrid Approach to Link Discovery [13]	DBpedia/cities & towns	Geonames/ populatedPlaces	gn:name vs. gn:alternateName	
		Linkedgeodata	Geonames	wgs84:long vs. wgs84:long	trigrams for strings
				wgs84:lat vs. wgs84:lat	Euclidean for Numerics
		DBpedia	Linkedgeodata	rdfs:label	
				population	
3	SILK: A Link Discovery Framework for the Web of Data [15]	Dbpedia/cities	Geonames/ PopulatedPlaces	rdfs:label vs. gn:name	JaroSimilarity
				rdfs:label vs. gn:alternateName	JaroSimilarity
				foaf:page vs. gn:wikipediaArticle	maxSimilarityInSets
				dbpedia:populationTotal vs. gn:population	numSimilarity
				wgs84_pos:lat vs. wgs84_pos:lat	numSimilarity
				wgs84_pos:long vs. wgs84_pos:long	numSimilarity
4	Active Learning of Expressive Linkage Rules using Genetic Programming [16]	SiderDrugBank NewYorkTimes LinkedMDB DBpediaDrugBank		Out of concern, active learning is the target	Levenshtein Jaccard Numeric Geographic

defined by Subjects and Values of the Subjects (Ex. subject: Name, value: Amoxicillin; subject: Chemical_Formula, value: $C_{16}H_{19}N_3O_5S$). Next, the file containing Link specifications is imported, subjects are specified, and operators of both aggregation and comparison are chosen, in order to execute the comparison operation. Finally, the generated links should be filtered into link-candidates to be evaluated and exported to users in user-specified output files.

A link discovery process can be divided into 3 stages (Fig. 2), and all Link Discovery Frameworks apply this process in order to generate relatively accurate links.

The first stage is the Pre-Matching stage. It is concerned about configuring the framework in an optimized manner, and includes bringing to-be-linked data from their corresponding resources, which are a source dataset and a target dataset that might be in the form of RDF dumps or SPARQL endpoints. After source and target datasets are drawn out, the specifications of the to-be-generated links are written into a file and imported by the framework to compare data based on them. Some other framework-specific parameters are to be stated in the pre-processing stage too, as for example, the link acceptance threshold value. In addition, if machine-learning is involved in the link discovery process, training datasets are to be imported at this stage. On the other hand, some frameworks provide the option of benefiting from external resources like dictionaries of RDF vocabulary or previous mappings that are a form of crowds participation (crowd-sourcing).

As the Pre-Matching stage ends, the matching stage starts, and it can be of two types: Instance Matching or Ontology Matching. End-users can intervene in the automated process in cases of learning-based matching, and this intervention is a candidate role for crowd-sourcing. SILK and LIMES are instance matching frameworks and thus they are not involved in ontology matching. When the matching process is complete, the result would be the discovered link candidates with a percentage or a relative value clarifying its accurateness for each.

Post-processing the outcome is evaluating the link candidates that are below the acceptance threshold and above the verification threshold. This stage can be done automatically (using Machine Learning) based on the framework architecture, or manually which would be a form of crowd-sourcing. Finally, the links are to be exported in a user-specified format, get published or saved.

IV. SILK & LIMES IN A NUTSHELL

A. SILK

SILK [15] is an open-source link discovery framework for the web, based on RDF. SILK offers a Workbench that enables the user to manage sets of data sources, easily edit and observe linking and transformation tasks graphically, create and edit reference links, and easily evaluate the generated links.

SILK queries the data lists drawn out of the corresponding resources using resources listers. The source and target datasets could be local data dumps, or resources accessed via a SPARQL endpoint. Only the target lists pass through an indexer that is responsible for indexing them as a pre-processing step, in order to facilitate the matching process. The indexing process separates data into blocks and indexes them by one or more of their properties (mostly labels). The

source lists do not pass through the indexer but are directly cached on disk in order to be retrieved later at the matching stage. The reason why only target lists get indexed is that the matching process will be executed on each instance of the source list, in order to compare it to the best potential matches from the target list. The indexing of the target list, will yield to a run-time optimization at the comparison level of the matching process. This time-optimizing step might cause missing some links when excluding blocks of lower matching potential that contain correct links. Only the retained links will be written to an output file which format can be specified by the user (i.e. CSV).

In the matching process, a similarity value for every pair of instances is computed and the corresponding aggregation metric (specified by the user) is evaluated. Then, comparison measures (that are metrics or semi-metrics) are acted upon by RDF path translators, which transform them into SPARQL queries and send them to SPARQL endpoints to get evaluated. Results of the query are cached temporarily in memory, until links of values that are above the acceptance threshold are picked and saved into it. The number of links to be picked for each resource is specified initially by the user. This is called link limit. The resulting links are picked from a list of link candidates, in which each has a corresponding similarity value (similarity between the source instance and target instance), such that they have the highest similarity values within the highest potential "blocks of matching" (according to the calculated indices of instances).

B. LIMES

The word "LIMES" [12] stands for Link Discovery Framework for Metrics Spaces. Like SILK, it is a tool to generate links between similar entities belonging to different data resources. However, LIMES tool estimates the similarity values using the mathematical characteristics of metric spaces. This helps reduce the number of comparisons and thus decreases the complexity of the run-time process.

The mathematical principles underlying the LIMES framework are summarized by defining the Metric Space and the Matching Task. The metric space is described by four conditions: non-negativity, identity of indiscernible, symmetry, and triangle inequality (TI) [13]. The difference with semi-metrics is that they do not satisfy the fourth condition of metrics (TI). On the other side, a matching task is computing the list of instances from source and target sets such that they match the metric conditions.

The General work-flow of LIMES starts by reading three inputs; source and target datasets, and the link specification file. After being imported, the data from the datasets is separated into "strings", "numeric values and values mappable into vector space", and "leftover values". They are then mapped using String Mapper, Numeric Mapper, and Miscellaneous Mapper respectively. The Mappers guarantee that the data is converted into values belonging to the metric space, according to the boundary condition realized from the TI [13], as follows:

$$m(x, y) - m(y, z) \leq m(x, z) \leq m(x, y) + m(y, z)$$

The distance from x to z can be approximated when knowing the distance from x to reference point y as well as the distance from reference point y to z. The reference point y is called an *exemplar* [13], and is used to define the center of a portion of the total metric space. Exemplars help calculate the upper and lower bounds of the distance from point x to point z, so when compared to theta, the threshold, the decision can be taken regarding link generation. A set of exemplars is selected in a way they be distributed in a uniform way in the metric space, and to be as dissimilar as possible. The approximations of distances from points to exemplars allow reducing the time needed for comparisons.

C. SILK vs. LIMES

In this section, we will compare SILK and LIMES according to the following features: General Information and Accessibility, Framework Configuration, Run-time Optimization and Link Discovery and Evaluation. Current studies emphasize on run-time optimization issues while disregarding other important features of the frameworks. Add to that, the fact that tests are performed on older versions of both frameworks (Table I). From here rises the need for an updated complete comparison that covers all the features of SILK and LIMES in order to be able to do an informed evaluation of their effectiveness regarding link quality.

1) General Information and Accessibility: Features under this category are as follows:

- While LIMES is based on Java, SILK initial release is developed with Python(2009) and the second version was reimplemented using Scala(2010).
- A web interface is available for SILK[2], while a practical desktop application is available for LIMES[3].
- Tools and sources are available to download for both frameworks; SILK[4] and LIMES[5].
- Both frameworks have a link specification language; SILK-LSL and LIMES-LSL.
- SILKs latest version -until the writing of this report- was published on 12/2/2016 while LIMES' was on 4/4/2017.

2) Framework Configuration: The difference between SILK and LIMES regarding the framework configuration are:

- Both frameworks support manual and learning based link discovery process, but SILK supports two additional methods; the generation of links using genetic programming and batch learning [16].
- The configuration information of Link Specification file of SILK and LIMES, which specify the elements of the comparison, are very similar (datasets, classes, aggregation operators, comparison operators, link limit, xml version, etc) but are organized differently.
- The Input/Output file formats supported by both frameworks are various, and differ between SILK and LIMES.

[2]http://SILKframework.org/
[3]http://aksw.org/Projects/LIMES.html
[4]https://github.com/silk-framework/silk
[5]https://github.com/dice-group/LIMES

SILK supports eight source-file formats and LIMES supports six, of which XML, RDF Dump, SPARQL endpoint, and N-TURTLE are in common. As for the format of the output, both support N-triples.

- Acceptance threshold: By comparing SILK and LIMES, there is a threshold value for each specification to accept the similarity values. In addition, the user can set threshold values for which links could be automatically accepted or links should be reviewed manually. Regarding SILK, it allows, specifying the number of links of a single data item to be picked. Only the highest-rated links per source data item will remain after the filtering.

3) Run-Time Optimization: Run-time is the execution time of the matching process excluding the execution time of pre-processing and post-processing operations. A comparison of how each framework achieves run-time optimization are:

- Parallel clustering: parallel processing is supported by both frameworks using customized versions of MapReduce,
- Pre-processing methods: The pre-processing method adopted by SILK relies on dividing data into blocks in order to enable indexing (oftenly indexed by Labels) and thus reduce comparisons. As for LIMES, the pre-processing method applied is filtering. Sources of pre-processing functions are: Xapian search engine library for SILK, while the novel version of the LIMES framework integrates an extended version of PPJoin+ algorithm [13].

Efficiency tests performed to compare the run-time of SILK and LIMES concluded that LIMES is faster by 60 times, which we justify by the difference in the pre-matching methods. Yet, the versions tested upon are old, and should be updated.

4) Link Discovery and Evaluation: Link Discovery and Evaluation parameters depend on multiple features, as follows:

- Supported measures: SILK, in its latest version, has 16 similarity measures, versus 60 similarity measures for LIMES in its 1.1.2 version. However, it should be noted that both frameworks do support the addition of new measures, with the difference that in LIMES the user should add mappers to such measures to fit in the filtering pre-matching process.
- Generated links: Both frameworks support the generation of owl:sameAs link types in addition to other RDF link [17] types. When links are generated, they are to be evaluated by the framework, then judged it to be accepted or not, before releasing them to output files. However, for SILK, the user can interfere in filtering the results and accepting the links by approving links with similarity measures under the threshold, or declining links with high similarity values (crowd-sourcing).
- Links evaluation: SILK makes evaluation and comparison weight optional to state whether the measure is mandatory (required) or not, and optional to give each measure a certain weight of the total weight too. As for LIMES, those parameters are not optional. A measure is supposed to be specified when it is required, with no ability to disregard its similarity values. Add to this that all measures are weighed the same in LIMES (no weight parameter).

V. PRE-EVALUATION PHASE

Before evaluating the two frameworks performance and impact on link quality, the data to-be-matched should be carefully selected, and the link specification parameters set, taking into account the difference at the level of threshold concept in each framework.

A. Comparison Measures

There exists five String-specific measures in both tools. While SILK splits up Character-based from Token-based measures for Strings, LIMES does not. For Numeric measures, while SILK, provides a date-specific measure, LIMES does not. In fact, the only Numeric measure supported by LIMES is the Euclidean distance measure. SILK classifies wgs84 measure as numeric. On the other hand, wgs84 is specific for geo properties (such as georss:point). This led us to include it under the geo type too, in parallel with the 19 geo-specific LIMES measures. However, SILK has, in addition, two extensions of Spatial Relations and Temporal Relations, that are Temporal Distances and Spatial Distances specific to centroid, minimum distances, days, hours, milliseconds, minutes, months, seconds, and years. Therefore, the only common measures between the two tools are Jaro, JaroWinkler, Levenshtein, Cosine and Jaccard. So, it is most relevant to compare the two frameworks based on these measures in order to detect the differences in their behavior, precisely and fairly.

B. Threshold-based Similarity and Distance

Based on the Link discovery behavior, we can formally use two type of threshold [13]:

- *Link Discovery on the similarity threshold*. Given two sets S and T of instances, a similarity measure ρ over the properties of s $\in$ S and t $\in$ T and a similarity threshold $\tau \in [0, 1]$, the goal of LD is to compute the set of pairs of instances (s, t) $\in$ S $\times$ T such that ρ(s, t) $\geqslant \tau$.
- *Link Discovery on the distance threshold*. Given two sets S and T of instances, a distance measure ι over the properties of s $\in$ S and t $\in$ T and a distance threshold $\theta \in [0, +\infty[$ the goal of LD is to compute the set of pairs of instances (s, t) $\in S \times T$ such that $\iota(s, t) \leqslant \theta$.

While LIMES uses similarities, SILK works with distances. Therefore, we use the setting $\tau = (1 + \theta)^{-1}$ to transform the distance threshold θ to the similarity threshold τ

Moreover, it is worth noting that not all measures available in SILK are normalized: Levenshtein, wgs84, date, dateTime, and num20 are not normalized. The use of non-normalized measures may lead to similarity values that are higher than the threshold set. For instance, Normalized Levenshtein Distance should be used instead of Levenshtein in SILK. Conversely, all measures are normalized in LIMES.

In our tests, we used 0.6, 0.8 and 0.95 thresholds (according to LIMES threshold concept) in order to calculate the precision and compare it between both frameworks.

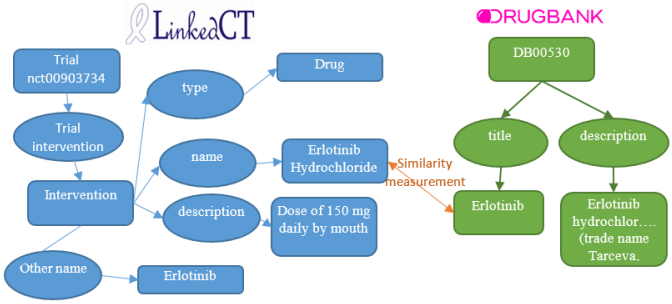


Fig. 3. The measurement similarity between the two datasets is based on intervention name of linkedct and title of the drug from drugbank.

C. Datasets & Link Specifications

Linking biomedical datasets will lead to novel facilitation for global health systems and thus humanity. Therefore, in this experiment, we will test and study the behavior of both SILK and LIMES in link discovery between two biomedical datasets which are the following:

LinkedCT[6] is a dataset derived from a service named ClinicalTrials.gov, which is initially provided by the U.S. National Institute of Health. The mentioned service is mainly a registry of more than 60 thousands entries of clinical trials conducted in 158 countries. Each clinical trial is associated with relevant information such as a brief description of the trial, disorders and interventions related to it, eligibility criteria, sponsors, locations (investigators),etc. The RDF version of the dataset contains 48,909,090 triples and 2,023,055 links [18].

DrugBank[7] is a large repository of around 5000 small molecule and biotech drugs that are FDA-approved. It contains detailed information about drugs (pharmacological, chemical and pharmaceutical data) in addition to comprehensive drug target data (like structure, sequence, and pathway information). Triples contained by the Linked data version of DrugBank are 3,649,531 triples, while links are 1,828,410 links [19].

Regarding comparison metrics, the discerned common measures were tested (Levenshtein, Jaro, JaroWinkler, Jaccard and cosine), in order to guarantee the most relevant results. The properties compared were *title* property of Drugbnak and intervention holds the drug name *(intervention_intervention_name)* property for LinkedCT. Then run-times were calculated.

D. Gold Standard

To evaluate links created when testing SILK and LIMES discovery frameworks, we have chosen to leverage existing "seeAlso" links between Linkedct and Drugbank. Therefore 52084 links were extracted from Linkedct dataset and prepared to be used as a gold standard. Moreover, we developed a Java application[8] to compare links discovered by SILK and LIMES with the gold standard as well as measure the quality metrics of links. The application takes the gold standard and the generated (.nt) file in order to compare and calculate various metrics (such as precision and recall) detailed in the next section.

[6]http://linkedct.org/
[7]https://old.datahub.io/dataset/bio2rdf-drugbank
[8]https://github.com/housseindh/LinkDiscoveryEvaluationMetrics

VI. EVALUATION

A. Experimental objectives and Set-Up

The twofold objective of the study of SILK and LIMES is to: 1)evaluate the quality of discovered links in case of biomedical datasets, according to two dimensions: thresholds and similarity measures; and 2)evaluate the run-time of each experiment.

We built our scenario around the *Intervention Name* property of type Drug as source and the *Title* property of Drug instances as a target using "Linkedct" and "Drugband" datasets respectively. We chose these datasets in a way to emphasize the difference at the level of link quality. For instance, the compared data could have different length of the string and possible token permutations. Fig. 3 describes examples of triples from the two datasets as well as their similar properties.

We held this test on 346576 different entities of source dataset (LinkedCT) against 7678 entities of target dataset (Drugbank). We performed it using each of the four String-specific comparison measures (Levenshtein, Jaccard, Jaro and JaroWinkler). Cosine comparison measure was excluded as the data was not compatible with it in SILK. Testing using different thresholds is important because sometimes, correct links have low similarity values, which needs lower thresholds to allow their detection.

All experiments were performed on a laptop equipped with Intel Core i7 quadcore processor (2.90 GHz), 20 GB RAM, the maximum heap size is set to 10 GB, running Windows 10, Java version JDK/JRE 1.8.

To evaluate the correctness of the links generated by the matching process, three measures should be calculated for different experiment sets: Precision, Recall, and F-Score.

$$Precision = \frac{TP}{(TP+FP)} \quad Recall = \frac{TP}{(TP+FN)}$$
$$FScore = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{1}$$

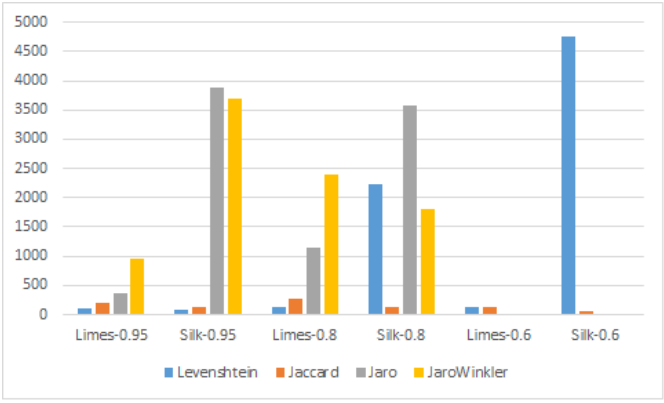
Where TP = True Positive, FP = False Positive and FN = False Negative

B. Experimental Results

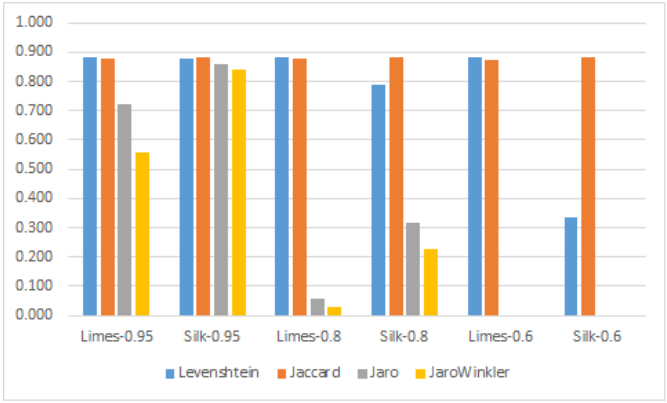
The columns in table II indicate the average result of 3 runs: False Positive(FP), True Positive(TP) and False Negative(FN) for three different thresholds(0.95, 0.8, 0.6). We used four different similarity measures for evaluation. As an overall observation, TP retained an approximately similar value for each test, which corresponds to the number of entities in the gold standard dataset. Accordingly, LIMES performed particularly well by retaining the same values with different thresholds using Levenshtein and Jaccard. However, SILK accomplished that only with Jaccard. All other values varied according to the 3-dimensions of computation; frameworks, thresholds and similarity measures.

TABLE II
FP, TP AND FN FOR THE TASK OF INTERLINKING LINKEDCT AND DRUGBANK USING DIFFERENT THRESHOLDS AND SIMILARITY MEASURES.

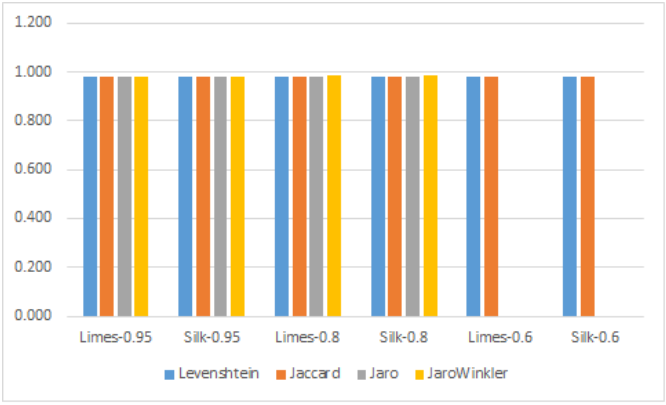
Threshold	0.95						0.8						0.6					
	LIMES			SILK			LIMES			SILK			LIMES			SILK		
Similarity	FP	TP	FN	FP	TP	FN	FP	TP	FN	FP	TP	FN	FP	TP	FN	FP	TP	FN
Levenshtein	6901	50965	1118	7088	50930	1153	6901	50965	1118	13814	50967	1116	6901	50965	1118	101677	51056	1027
Jaccard	6973	50989	1094	6928	51075	1008	6973	50989	1094	6913	50986	1097	6901	50965	1118	6908	50957	1126
Jaro	19801	51113	970	8316	50993	1090	886401	51189	894	109944	51189	894	Java heap space			GC overhead		
JaroWinkler	40301	51138	945	9734	51027	1056	1720793	51194	889	176815	51324	759	GC overhead			GC overhead		



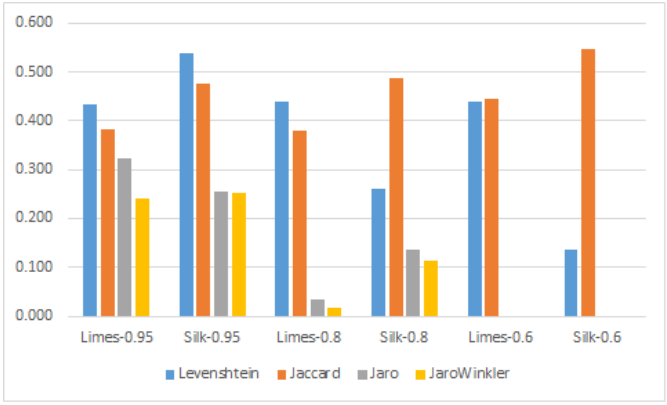
(a) Run-time in second



(b) Precision



(c) Recall



(d) EE: deviding FScore by $log(Runtime)$.

Fig. 4. Experimental Results of metrics of links generated by multiple similarity measures of interlinking LinkedCT and DrugBank.

Although giving 10 GB of memory, the two similarity measures Jaro and JaroWinkler failed to produce a result with a threshold of 0.6, because of a Java GC overhead limit exceeded and Java heap space.

Regarding the run-time evaluation, Fig 4a summarizes our experiment results of SILK and LIMES. As an overall observation, we find that Jaccard performed the optimal time for all thresholds. And as we compare the time of LIMES and SILK, we observe in most experiments that LIMES is faster than or approximately equal to SILK. The only case where SILK run-time noticeably exceeded LIMES's was with JaroWinkler comparison operator at 0.8 threshold.

Fig.4b and 4cshow the quality metrics of linked discovery. Both frameworks achieved very good results in terms of recall. In terms of precision, while LIMES maintained very good results for all thresholds when dealing with Levenshtein and Jaccard, SILK had poor results with 0.6 threshold using Levenshtein for the same experiment specifications. Moreover, for the preceding experiments of Jaro and JaroWinkler, the results were poor for both frameworks, and worse when speaking about LIMES.

In order to evaluate the effectiveness and efficiency of these two frameworks, we propose to use an equation that calculates the proportional value of F-Score to the run-time duration. However, because of the considerable differences in the value of the execution time between each experiment compared to the value of F-Score, we apply the logarithm function to smooth out the high impact of run-time for big

values. Therefore we propose to evaluate the effectiveness and efficiency by using the following EE equation:

$$EE = \frac{FScore}{log(Runtime)} \quad (2)$$

Looking at the results of the EE equation in Fig. 4d, it seems that LIMES has maintained consistent effectiveness regardless of the Threshold for both similarity measures Levenshtein and Jaccard, while it was remarkably unsteady with SILK.

C. Technical Evaluation

LIMES framework admits its drawback [12] considering its optimization only for metrics, which is not the case for non-metric measures such as JaroWinkler. It favors performance and ease-of-use over recall/precision factors, and considers that the contribution of the user in modifying the thresholds as a human-feedback that can compensate this drawback.

The results evaluation using Precision, Recall and F-Score confirms the fact that LIMES theoretically has better chances than SILK in the case of large datasets . However, the close results between SILK and LIMES in the current tests do not exclude SILK from the efficient universal link discovery frameworks.

The two frameworks tend to perform a pre-matching process to improve the performance of comparison. In addition, to speed the process up, the target objective is to reduce the number of comparisons needed to be held. SILK uses indexing, while LIMES uses the Triangle Inequality . Obviously, the algorithm used in LIMES, which depends on computing exemplars from the resources and filtering them before computing the similarity and serializing the result [12] is the reason why LIMES is faster and makes it less probable to miss links. The distribution of exemplars, where each represents a portion of the metric space and which are selected as dissimilar as possible in the set of data, allows the parallelism of the filtering process before matching. Filtering takes place by matching each point to an exemplar to compute pessimistic estimates of instance similarities, which leads to missing links. In SILK, the indexing process allows dividing the data into blocks and indexing them by some of their properties (mostly labels), then, for each comparison, matching is performed only on potential blocks. This would lower the number of comparisons and thus will speed up the process but does not guarantee not missing links.

VII. Conclusion & Future works

In this paper, we summarized the core differences between SILK and LIMES and presented an experiment that evaluates the performance of entity comparison and measures the quality of discovered links. In particular, we applied the process to large-scale biomedical data (LinkedCT and Drugbank datasets). We performed many experiments to evaluate the impact of threshold values and that of similarity measures on efficiency and effectiveness, in order to verify the points of strength and weakness of each framework. This comprehensive study clarified and validated the fact that each of SILK and LIMES has its own advantages, and is more appropriate to specific similarity measures usage. More specifically, LIMES flourishes with Levenshtein at all thresholds while SILK emerges with Jaccard at low thresholds.

As a future plan, we aim to perform more tests on the rest of the comparison measures, and upon different aggregation scenarios to get deep into the best use-case domain of each framework. On the other hand, a great deal of work shall be focused on considering active learning that is already integrated into both frameworks, and on testing the performance in a distributed environment.

References

[1] I. Merelli, H. Pérez-Sánchez, S. Gesing, and D. DAgostino, "Managing, analysing, and integrating big data in medical bioinformatics: open problems and future perspectives," *BioMed research international*, vol. 2014, 2014.

[2] H. Dhayne, R. Haque, R. Kilany, and Y. Taher, "In search of big medical data integration solutions-a comprehensive survey," *IEEE Access*, vol. 7, pp. 91 265–91 290, 2019.

[3] P. Hitzler and K. Janowicz, "Linked data, big data, and the 4th paradigm." *Semantic Web*, vol. 4, no. 3, pp. 233–235, 2013.

[4] H. Dhayne, R. K. Chamoun, and M. Sokhn, "Survey: When semantics meet crowdsourcing to enhance big data variety," in *Communications Conference (MENACOMM), IEEE Middle East and North Africa*. IEEE, 2018, pp. 1–6.

[5] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data: The story so far," in *Semantic services, interoperability and web applications: emerging concepts*. IGI Global, 2011, pp. 205–227.

[6] M. Samwald, A. Jentzsch, C. Bouton, C. S. Kallesøe, E. Willighagen, J. Hajagos, M. S. Marshall, E. Prud'hommeaux, O. Hassanzadeh, E. Pichler *et al.*, "Linked open drug data for pharmaceutical research and development," *Journal of cheminformatics*, vol. 3, no. 1, p. 19, 2011.

[7] H. Dhayne, R. Kilany, R. Haque, and Y. Taher, "Sedie: A semantic-driven engine for integration of healthcare data," in *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2018, pp. 617–622.

[8] A.-C. N. Ngomo, S. Auer, J. Lehmann, and A. Zaveri, "Introduction to linked data and its lifecycle on the web," in *Reasoning Web International Summer School*. Springer, 2014, pp. 1–99.

[9] J. D. Fernández, W. Beek, M. A. Martínez-Prieto, and M. Arias, "Lod-a-lot," in *International Semantic Web Conference*. Springer, 2017, pp. 75–83.

[10] R. Haque and M.-S. Hacid, "Blinked data: Concepts, characteristics, and challenge," in *Services (SERVICES), 2014 IEEE World Congress on*. IEEE, 2014, pp. 426–433.

[11] Y. Raimond, C. Sutton, and M. B. Sandler, "Automatic interlinking of music datasets on the semantic web." *LDOW*, vol. 369, 2008.

[12] A.-C. N. Ngomo and S. Auer, "Limes-a time-efficient approach for large-scale link discovery on the web of data." in *IJCAI*, 2011, pp. 2312–2317.

[13] A.-C. N. Ngomo, "A time-efficient hybrid approach to link discovery," *Ontology Matching*, vol. 1, 2011.

[14] M. Nentwig, M. Hartung, A.-C. Ngonga Ngomo, and E. Rahm, "A survey of current link discovery frameworks," *Semantic Web*, vol. 8, no. 3, pp. 419–436, 2017.

[15] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov, "Silk-a link discovery framework for the web of data." *LDOW*, vol. 538, 2009.

[16] R. Isele and C. Bizer, "Active learning of expressive linkage rules using genetic programming," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 23, pp. 2–15, 2013.

[17] D. Beckett and B. McBride, "Rdf/xml syntax specification (revised)," *W3C recommendation*, vol. 10, no. 2.3, 2004.

[18] O. Hassanzadeh, A. Kementsietsidis, L. Lim, R. J. Miller, and M. Wang, "Linkedct: A linked data space for clinical trials," *arXiv preprint arXiv:0908.0567*, 2009.

[19] D. S. Wishart, C. Knox, A. C. Guo, S. Shrivastava, M. Hassanali, P. Stothard, Z. Chang, and J. Woolsey, "Drugbank: a comprehensive resource for in silico drug discovery and exploration," *Nucleic acids research*, vol. 34, no. suppl_1, pp. D668–D672, 2006.