Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

HESML: A scalable ontology-based semantic similarity measures library with a set of reproducible experiments and a replication dataset

Juan J. Lastra-Díaz[a,*], Ana García-Serrano[a], Montserrat Batet[b,1], Miriam Fernández[c,1], Fernando Chirigati[d,1]

[a] NLP & IR Research Group, E.T.S.I. Informática, Universidad Nacional de Educación a Distancia (UNED), C/Juan del Rosal 16, 28040 Madrid, (Spain)
[b] Internet Interdisciplinary Institute (IN3), Universitat Oberta de Catalunya, Av. Carl Friedrich Gauss, 5. 08860 Castelldefels, Spain
[c] Knowledge Media Institute, The Open University, Walton Hall, Mlton Keynes, MK7 6AA, United Kingdom
[d] Department of Computer Science and Engineering, New York University, New York City, NY, United States

A R T I C L E I N F O

Article history:
Received 26 July 2016
Revised 6 February 2017
Accepted 10 February 2017
Available online 21 February 2017

Keywords:
HESML
PosetHERep
Semantic measures library
Ontology-based semantic similarity measures
Intrinsic and corpus-based Information Content models
Reproducible experiments on word similarity
WNSimRep v1 dataset
ReproZip
WordNet-based semantic similarity measures

A B S T R A C T

This work is a detailed companion reproducibility paper of the methods and experiments proposed by Lastra-Díaz and García-Serrano in (2015, 2016) [56–58], which introduces the following contributions: (1) a new and efficient representation model for taxonomies, called *PosetHERep*, which is an adaptation of the half-edge data structure commonly used to represent discrete manifolds and planar graphs; (2) a new Java software library called the *Half-Edge Semantic Measures Library* (*HESML*) based on *PosetHERep*, which implements most ontology-based semantic similarity measures and Information Content (IC) models reported in the literature; (3) a set of reproducible experiments on word similarity based on *HESML* and ReproZip with the aim of exactly reproducing the experimental surveys in the three aforementioned works; (4) a replication framework and dataset, called *WNSimRep v1*, whose aim is to assist the exact replication of most methods reported in the literature; and finally, (5) a set of scalability and performance benchmarks for semantic measures libraries. *PosetHERep* and *HESML* are motivated by several drawbacks in the current semantic measures libraries, especially the performance and scalability, as well as the evaluation of new methods and the replication of most previous methods. The reproducible experiments introduced herein are encouraged by the lack of a set of large, self-contained and easily reproducible experiments with the aim of replicating and confirming previously reported results. Likewise, the *WNSimRep v1* dataset is motivated by the discovery of several contradictory results and difficulties in reproducing previously reported methods and experiments. *PosetHERep* proposes a memory-efficient representation for taxonomies which linearly scales with the size of the taxonomy and provides an efficient implementation of most taxonomy-based algorithms used by the semantic measures and IC models, whilst *HESML* provides an open framework to aid research into the area by providing a simpler and more efficient software architecture than the current software libraries. Finally, we prove the outperformance of *HESML* on the state-of-the-art libraries, as well as the possibility of significantly improving their performance and scalability without caching using *PosetHERep*.

1. Introduction

Human similarity judgments between concepts underlie most of cognitive capabilities, such as categorization, memory, decision-making and reasoning. Thus, the proposal for concept similarity models to estimate the degree of similarity between word and concept pairs has been a very active line of research in the fields of cognitive sciences [106,124], artificial intelligence and Information Retrieval (IR) [107]. The semantic similarity measures esti-

* Corresponding author.
E-mail addresses: jlastra@invi.uned.es (J.J. Lastra-Díaz), agarcia@lsi.uned.es (A. García-Serrano), mbatetsa@uoc.edu (M. Batet), m.fernandez@open.ac.uk (M. Fernández), fchirigati@nyu.edu (F. Chirigati).
[1] Reviewers

http://dx.doi.org/10.1016/j.is.2017.02.002

mates the degree of similarity between concepts by considering only 'is-a' relationships, whilst the semantic relatedness measures also consider any type of co-occurrence relationship. For instance, a *wheel* is closely related to a *car* because the wheels are part of any car; however, a *wheel* neither is a car nor derives from another common close concept as *vehicle*, thus their degree of similarity is low. Whilst hand-coded taxonomies, such as WordNet and other sources of knowledge, can be efficiently and reliably used to retrieve the 'is-a' relationships between concepts and words, the co-occurrence relationships required by the semantic relatedness measures need to be retrieved from a large corpus. For this reason [57, §1.1], ontology-based semantic similarity measures exclusively based on 'is-a' relationships are currently the best and most reliable strategy to estimate the degree of similarity between words and concepts [58], whilst the corpus-based similarity measures are the best strategy for estimating their degree of relatedness [8].

An ontology-based semantic similarity measure is a binary concept-valued function $sim : C \times C \to \mathbb{R}$ defined on a single-root taxonomy of concepts $(C, \leq_C)$, which returns an estimation of the degree of similarity between concepts as perceived by a human being. The ontology-based similarity measures have become both a very active research topic, and a key component in many applications. For instance, in the fields of Natural Language Processing (NLP) and IR, ontology-based semantic similarity measures have been used in Word Sense Disambiguation (WSD) methods [92], text similarity measures [86], spelling error detection [20], sentence similarity models [44,66,91], paraphrase detection [36], unified sense disambiguation methods for different types of structured sources of knowledge [73], document clustering [31], ontology alignment [30], document [74] and query anonymization [11], clustering of nominal information [9,10], chemical entity identification [40], interoperability among agent-based systems [34], and ontology-based Information Retrieval (IR) models [55,62] to solve the lack of an intrinsic semantic distance in vector ontology-based IR models [23]. In the field of bioengineering, ontology-based similarity measures have been proposed for synonym recognition [24] and biomedical text mining [14,98,112]. However, since the pioneering work of Lord et al. [72], the proposal of similarity measures for genomics and proteomics based on the Gene Ontology (GO) [5] have attracted a lot of attention, as detailed in a recent survey on the topic [76]. Many GO-based semantic similarity measures have been proposed for protein functional similarity [28,29,101,132], giving rise to applications in protein classification and protein-protein interactions [41,129], gene prioritization [117] and many others reported in [76, p.2].

In [57], Lastra-Díaz and García-Serrano introduce a new family of similarity measures based on an Information Content (IC) model, whose pioneering work is introduced by Resnik [108]. Their new family of semantic similarity measures is based on two unexplored notions: a non-linear normalization of the classic Jiang-Conrath distance [52], and a generalization of this latter distance on non tree-like taxonomies defined as the length of the shortest path within an IC-weighted taxonomy. One of the similarity measures introduced in [57], called *coswJ&Csim*, obtains the best results on the RG65 dataset. In another subsequent work [56], the same aforementioned authors introduce a new family of intrinsic and corpus-based IC models and a new algebraic framework for their derivation, which is based on the estimation of the conditional probabilities between child and parent concepts within a taxonomy. This latter family of IC models is refined in another subsequent paper [58], which also sets out the new state of the art and confirms the outperformance of the *coswJ&Csim* similarity measure in a statistically significant manner among the family of ontology-based semantic similarity measures based on WordNet.

Given a taxonomy of concepts defined by the triplet $\mathcal{C} = ((C, \leq_C), \Gamma)$, where $\Gamma \in C$ is the supreme element called the root, an Information Content model is a function $IC : C \to \mathbb{R}^+ \cup \{0\}$, which represents an estimation of the information content for every concept, defined by $IC(c_i) = -log_2(p(c_i))$, $p(c_i)$ being the occurrence probability of each concept $c_i \in C$. Each IC model must satisfy two further properties: (1) nullity in the root, such that $IC(\Gamma) = 0$, and (2) growing monotonicity from the root to the leaf concepts, such that $\forall c_i \leq_C c_j \Rightarrow IC(c_i) \geq IC(c_j)$. Once the IC-based measure is chosen, the IC model is mainly responsible for the definition of the notion of similarity and distance between concepts.

The main aim of this work is to introduce the *PosetHERep* representation model and make the *Half-Edge Semantic Measures Library* (HESML*)* publicly available for the first time, together with a set of reproducible experiments whose aims are the exact replication of the three aforementioned experimental surveys [56–58], as well as the proposal for a self-contained experimental platform which can be easily used for extensive experimentation, even with no software coding. In addition, this work also introduces a new replication framework and the *WNSimRep v1* dataset for the first time provided as supplementary material in [63], whose aim is to provide a gold standard to assist in the exact replication of ontology-based similarity measures and IC models. Finally, we have carried-out a series of experiments in order to evaluate the scalability and performance of HESML as regards the Semantic Measures Library (SML) [48] and WNetSS [15], which sets out the current state of the art. This work is part of a novel innitiative on computational reproducibility recently introduced by Chirigati et al. [26], whose pioneering work is introduced by Wolke et al. [127] with the aim of leading to the exact replication of several dynamic resource allocation strategies in cloud data centers evaluated in a companion paper [128].

1.1. Main motivation and hypothesis

The two main motivations of this work are three drawbacks in the current semantic measures libraries, detailed below, and the lack of a set of self-contained and easily reproducible experiments into ontology-based semantic similarity measures and IC models based on WordNet. Another significant motivation, also related to the reproducibility, is the lack of a gold standard to assist in the exact replication of ontology-based similarity measures and IC models.

1.1.1. On the current semantic measures libraries

Our first motivation is the discovery of several scalability and performance drawbacks in the current state-of-the-art semantic measures libraries. We argue that these aforementioned drawbacks are derived from the use of naive graph representation models which do not capture the intrinsic structure of the taxonomies being represented. As a consequence of this latter fact, all topological algorithms based on naive representation models demand a high computational cost which degrades their performance. In turn, in order to solve the performance problem of their graph-based algorithms, the current semantic measures libraries adopt a caching strategy, storing the ancestors and descendant sets of all vertexes within the taxonomy, among other topological queries in memory. This latter caching strategy significantly increases the memory usage and leads to a scalability problem as regards the size of the taxonomy, in addition to impacting the performance because of the further memory allocation and dynamic resizing of the caching data structures, or the interrogation of external relational databases.

Our main hypothesis is that a new representation model for taxonomies which properly encodes their intrinsic structure, together with a new software library based on it, should bridge the aforementioned gap of scalability and performance of the current semantic measures libraries. Thus, *our main research questions* are as follows: (Q1) is a new intrinsic representation model for taxonomies able to improve significantly the performance and scalability of the state-of-the-art semantic measures libraries?, and (Q2) is it possible to significantly improve the performance and scalability of the state-of-the-art semantic measures libraries without using any caching strategy?.

The current state-of-the-art libraries are based on caching for most topological queries and the *delocalization of attributes* from their base objects (vertexes and edges). For instance, SML represents the ontologies by graphs, in which each vertex and oriented edge is defined by a URI key in a Java hash set. Thus, any further information associated to each vertex or edge needs to be stored in any independent external data structure, an approach that we call *delocalized attributes*. In addition, SML uses hash sets to store all pre-computed information and topological queries associated to each vertex as follows: its incoming and outcoming edge sets, its ascendant and descendant sets, its minimum and maximum depths, its subsumed leaves and its IC values, among others. Following the same *delocalized approach*, the edge weights in SML are also stored in Java hash sets indexed by edge URIs. All the aforementioned taxonomical features are computed during the pre-processing step, or the first time that they are requested, being stored in their corresponding caching structures defined as hash sets or tables. All topological queries, as well as the shortest path algorithm implemented by SML, are based on the traversal of the SML graph model, as well as the cache information of the vertexes and their delocalized attributes. The cached taxonomical features are represented in a distributed collection of hash maps and sets indexed by edge and vertex URI keys. In short, the entire topological model of the SML is based on caching, hash maps and delocalized attributes from their base objects. One of the first consequences of caching the vertex sets, as the ancestor or descendant sets, is that it implies a non-linear increase in the use of memory. On the other hand, the delocalized approach adds a performance penalty because of the need to interrogate different hash maps in order to retrieve multiple attributes from the same underlying object, in addition to an increase in the memory required derived from the internal searching and storing structures required by the underlying hash maps. Finally, all graph traversal algorithms, especially the shortest path computation, suffer a significant decrease in performance derived from the lack of an efficient representation of the adjacency model. The SML algorithms needs to interrogate the hash maps continuously by storing the incoming and outcoming edge sets of each vertex in order to retrieve the adjacency information and traverse the graph. Thus, the traversing method is especially time consuming in complex algorithms as the shortest path computation. Another significant example of caching is the approach adopted by the WNetSS semantic measures library introduced recently by Aouicha et al. [15]. Unlike SML, which computes the topological features on-the-fly by storing them in an in-memory cache, WNetSS carries-out a time-consuming off-line preprocessing of all WordNet-based topological information which is stored in a MySQL server. This latter caching strategy based on MySQL could be appropriate for supporting a large Web-based experimental platform, such as the SISR system proposed in [15]. However, it severely impacts the performance, scalability and extensibility of WNetSS.

A second motivation is related to several software architecture issues that lead to practical difficulties for the functional extension of current software libraries. For instance, WordNet::Similarity [99] and WS4J [121] were designed before the emergence of the intrinsic IC models described in Section 2.1, thus, these libraries maintain in-memory tables with the concept frequency counts which are interrogated in order to compute the IC values required in a similarity evaluation step; however, their data structures does not provide any proper abstraction layer or software architecture to integrate new intrinsic IC models easily. On the other hand, SML separates the in-memory storage of the IC values and edge weights from the edge and nodes within the base taxonomy by defining two Java abstract interfaces to integrate new weighting schemes and IC models as external data providers which are interrogated on-the-fly. This latter software design decision looks fine from an abstract point of view; however, it hinders the implementation of weighted IC-based measures like the weighted J&C and coswJ&C similarity measures introduced by Lastra-Díaz and García-Serrano [57], because the edge weights depend on the IC values of the nodes.

A third motivation is the lack of software implementations for the most recent ontology-based similarity measures and intrinsic IC models developed during the last decade. This latter fact prevents the publication of exhaustive experimental surveys comparing the new proposed methods with most recent methods reported in the literature, because of the effort and difficulty in replicating previous methods and experiments.

1.1.2. On the reproducibility in the area

A fourth motivation of this work is the lack of a set of self-contained and easily reproducible experiments that allow the research community to be able to replicate methods and results reported in the literature exactly, even without the need for software coding. The lack of reproducible experiments, together with the aforementioned lack of software libraries covering the most recent methods, and the difficulties in replicating methods and experiments exactly have contributed, with few exceptions, to improvable reproducibility practices in the area. Many works introducing similarity measures or IC models during the last decade have only implemented or evaluated classic IC-based similarity measures, such as the Resnik [108], Lin [70] and Jiang-Conrath [52] measures, avoiding the replication of IC models and similarity measures introduced by other researchers. Some works have not included all the details of their methods, or the experimental setup to obtain the published results, thus, preventing their reproducibility. Most works have copied results published by others. This latter fact has prevented the invaluable confirmation of previously reported methods and results, which is an essential feature of science. Pedersen [94], and subsequently Fokkens et al. [37], warn of the need to reproduce and validate previous methods and results reported in the literature, a suggestion that we subscribe to in our aforementioned works [56–58], where we also refuted some previous conclusions and warn of finding some contradictory results. A recent study [6,33] on the perception of this reproducibility ‘crisis’ in science shows that the aforementioned reproducibility problems in our area are not the exception but the rule. Precisely, this latter fact has encouraged the recent manifesto for reproducible science [90], which we also subscribe.

And finally, our last motivation is the lack of a gold standard to assist in the exact replication of ontology-based similarity measures and IC models. Most ontology-based similarity measures and intrinsic IC models require the computation of different taxonomical features, such as node depths, hyponym sets, node subsumers, the Least Common Subsumer (LCS), and subsumed leaves, among others. WordNet is a taxonomy with multiple inheritance, thus, some of these features are ambiguously defined, or their computation could be prone to errors. For example, the node depth can be defined as the length of the shortest ascending path from the node to the root, or the length of the longest ascending path as defined by Taieb et al. [43]. Different definitions of depth also lead us to different values for the LCS concepts. On the other hand, the computation of the hyponym set, subsumed leaves and subsumer set requires a careful counting process to avoid node repetitions, as is already noted in [119, §3]. Another potential source of error is the ambiguity in the definition and notation of some IC models and similarity measures. For example, Zhou et al. [134] define the root depth as 1, whilst the standard convention in graph theory is 0. Most authors define the hyponym set as the descendant node set without including the base node itself. However, in [43], the hyponym set also includes the base concept. In addition, we find works that do not detail the IC models used in their experiments, or how these IC models were built. Finally, many recent hybrid-type measures also require the computation of the length of the shortest path between concepts. These sources of ambiguity and difficulty demand a lot of attention to the fine details for replicating most IC models and similarity measures in the literature. In a recent work [57], we find some contradictory results and difficulties in replicating previous methods and experiments reported in the literature. These reproducibility problems were confirmed in another subsequent work, such as [56], whilst new contradictory results are reported in [58]. Several replication problems were solved with the kind support of most authors. However, we were not able to confirm all previous results, whilst others could not be reproduced through lack of information. As we have explained above, many taxonomical features are ambiguously defined or prone to errors. Thus, all the aforementioned facts lead us to conclude that the exact replication of ontology-based similarity measures and IC models is a hard task, and not exempt from risk. Therefore, it follows that it is urgent and desirable to set off a gold standard for this taxonomical information in order to support the exact replication of the methods reported in the literature.

1.2. Definition of the problem and contributions

This work tackles the problem of designing a scalable and efficient new representation model for taxonomies and a new semantic measures library based on the former, as well as the lack of self-contained reproducible experiments on WordNet-based similarity, tools and resources to assist in the exact replication of methods and experiments previously reported in the literature. In order to bridge the aforementioned gap, the main contributions of this work are as follows: (1) a new and efficient representation model for taxonomies, called *PosetHERep*, which is an adaptation of the half-edge data structure commonly used to represent discrete manifolds and planar graphs in computational geometry; (2) a new Java software library called *Half-Edge Semantic Measures Library* (*HESML)* based on *PosetHERep*, which implements most ontology-based semantic similarity measures and Information Content (IC) models reported in the literature; (3) a set of reproducible experiments on word similarity based on *HESML* and ReproZip [27] with the aim of exactly reproducing the experimental surveys reported in [56–58]; (4) a replication framework and dataset, called *WN-SimRep v1*, which is provided as supplementary material at [63], and whose aim is to assist the exact replication of most methods reported in the literature; and finally, (5) the definition and evaluation of a set of scalability and performance benchmarks to compare the state-of-the-art semantic measures libraries.

The rest of the paper is structured as follows. Section 2 introduces the related work. Section 3 introduces the HESML software library and the PosetHERep representation model for taxonomies. Section 4 introduces a set of reproducible experiments as a companion work to the aforementioned works introduced by Lastra-Díaz and García-Serrano [56–58]. Section 5 briefly introduces the WNSimRep v1 dataset, which is detailed and made publicly available in [63] as complementary material. Section 6 introduces a series of benchmarks between HESML and two state-of-the-art semantic measures libraries with the aim of evaluating and comparing their scalability and performance. Section 7 introduces our discussion of the experimental results. Section 8 introduces our conclusions and future work, whilst Section 9 introduces the revision comments made by the reviewers. Finally, Appendix A details the resources and datasets included in the HESML V1R2 distribution.

2. Related work

This section is divided into four subsections according to the categorization of the related work detailed as follows. Section 2.1 categorizes the family of ontology-based similarity measures. Section 2.2 introduces the IC models which have been implemented in *HESML*. Section 2.3 introduces the main software libraries of ontology-based semantic similarity measures on WordNet reported in the literature. And finally, Section 2.4 introduces some potential applications in information systems. We only introduce herein a categorization of the methods reported in the literature, mainly those implemented in HESML. However, for an in-depth review of the latter topics, we refer the reader to the reviews by Lastra-Díaz and García-Serrano on IC-based similarity measures [57] and IC models [56,58], as well as the short review by Batet and Sánchez [12] and the book by Harispe et al. [49].

2.1. Ontology-based semantic similarity measures

Table 1 shows our categorization of the current ontology-based semantic similarity measures into four subfamilies as follows. First, edge-counting measures, the so-called path-based measures, whose core idea is the use of the length of the shortest path between concepts as an estimation of their degree of similarity, such as the pioneering work of Rada et al. [107]. Second, the family of IC-based similarity measures, whose core idea is the use of an Information Content (IC) model, such as the pioneering work of Resnik [108], and the subsequent measures introduced by Jiang and Conrath [52] and Lin [70]. Third, the familiy of feature-based similarity measures, whose core idea is the use of set-theory operators between the feature sets of the concepts, such as the pioneering work of Tversky [124]. And fourth, other similarity measures that cannot be directly categorized into any previous family, which are based on similarity graphs derived from WordNet [122], novel contributions of the hyponym set [43], or aggregations of other measures [75].

In turn, the more recent IC-based measures can be divided into four subgroups: (1) a first group made up by the aforementioned three classic IC-based similarity measures by Resnik [108], Jiang and Conrath [52], and Lin [70]; (2) a second group defined by those measures that make up an IC model with any function based on the length of the shortest path between concepts, such as the pioneering work of Li et al. [69], and other subsequent works shown in Table 1; (3) a third group of IC-based measures based on the reformulation of different approaches, such as the IC-based reformulations of the Tversky measure by Pirró and Seco [103], and the IC-based reformulation of most edge-counting methods introduced by Sánchez et al. [112]; and finally, (4) a fourth group of IC-based measures based on a monotone transformation of any classic IC-based similarity measure, such as the exponential-like scaling of the Lin measure introduced by Meng and Gu [81], the reciprocal similarity measure of the Jiang-Conrath distance introduced by Garla and Brandt [39], another exponential-like normalization of the Jiang-Conrath distance introduced by Lastra-Díaz and García-Serrano [57], and the monotone transformation of the Lin measure called FaITH introduced by Pirró and Euzenat [104]. Table 2 shows a summary of the ontology-based semantic similarity measures implemented by the main publicly available semantic measures libraries.

Finally, we mention five significant further lines of research into ontology-based similarity measures. Stanchev [122] introduces an asymmetric similarity weighted graph derived from WordNet, whilst Martínez-Gil [75] proposes an aggregated similarity measure based on a combination of multiple ontology-based similarity measures and Van Miltenburg [125] proposes a method to compute the semantic similarity between adjectives based on the use of the similarity between their sets of derivational source names in WordNet. More recently, Meymandpour et al. [85] propose several semantic similarity measures for Linked Open Data (LOD) based on IC models, whilst Batet and Sánchez [13] propose a semantic

Table 1
Categorization of the main ontology-based semantic similarity measures based on WordNet reported in the literature and implemented in HESML, excepting those measures with an asterisk (*). The categorization above excludes most GO-based semantic similarity measures, which are in-depth analyzed in a recent survey by Mazandu et al. [76].

Path-based measures		Rada et al. [107], Wu & Palmer [130]
		Leacock & Chodorow [65], Hirst & St-Onge [51]*
		Pedersen et al. [98], Al-Mubaid & NGuyen [3]
IC-based measures	Classic IC-based measures	Resnik [108]
		Jiang & Conrath [52]
		Lin [70]
	Hybrid (path-based) IC-based measures	Li et al. [69]
		Zhou et al. [133]
		Meng et al. [83]
		Gao et al. [38]
		Lastra-Díaz & García-Serrano(coswJ&C) [57]
	Reformulations of other types of measure	Pirró & Seco [103]
		Sánchez et al. [112]*
	Monotone transformations of classic IC-based measures	Pirró & Euzenat [104]
		Meng & Gu [81]
		Garla & Brandt [39]
		Lastra-Díaz & García-Serrano(cosJ&C) [57]
Feature-based measures		Tversky [124]
		Batet et al. [14]
		Sánchez et al. [115]
Other types of measure		- Taxonomical features (hyponym sets): Taieb et al. [43]
		- Aggregation of different of measures: Martínez-Gil [75]*
		- Asymmetrically weighted graphs based on WordNet: Stanchev [122]*
		- IC-based reformulation on LinkedOpenData (LOD): Meymandpour et al. [85]*
		- IC-based reformulation on Wikipedia: Jiang et al. [53]*

relatedness measure based on the combination of highly-accurate ontology-based semantic similarity measures with a resemblance measure derived from corpus statistics.

2.2. Information Content models

The first known IC model is based on corpus statistics and was introduced by Resnik [108], and subsequently detailed in [109]. The main drawback of the corpus-based IC models is the difficulty in getting a well-balanced and disambiguated corpus for the estimation of the concept probabilities. To bridge this gap, Seco et al. [119] introduce the first intrinsic IC model in the literature, whose core hypothesis is that the IC models can be directly computed from intrinsic taxonomical features. Thus, the development of new intrinsic IC-based similarity measures is divided into two subproblems: (1) the proposal of new intrinsic IC models, and (2) the proposal for new IC-based similarity measures. During the last decade, the development of intrinsic IC models has become one of the mainstreams of research in the area. Among the main intrinsic and corpus-based IC models proposed in the literature, we find the proposals by Zhou et al. [133], Sebti and Barfroush [118], Blanchard et al. [18], Sánchez et al. [113,114] , Meng et al. [82], Yuan et al. [131], Hadj Taieb et al. [42], Lastra-Díaz and García-Serrano [56,58], Adhikari et al. [1], Aouicha et al. [4,16], and Harispe et al. [46].

Finally, in another recent work, Jiang et al. [53] introduce a new intrinsic IC model based on the Wikipedia category structure which has obtained outstanding results in several word-similarity benchmarks. Table 3 shows a summary of the IC models implemented by the current semantic measures libraries.

2.3. Ontology-based semantic measures libraries

The main publicly available software libraries focusing on the implementation of ontology-based similarity measures based on WordNet are WordNet::Similarity (WNSim) [99] and WS4J [121], whose development is more stable, and the Semantic Measures Library (SML) [47] and the recent WNetSS [15] which are active ongoing projects.

The pioneering WNSim library was developed in Perl by Pedersen et al. [99], and subsequently migrated to Java by Tedeki Shima, under the name of WS4J [121]. WS4J includes, like its parent library, the most significant path-based similarity measures, the three aforementioned classic IC-based measures and several corpus-based IC models [95]. However, WNSim and WS4J do not include most ontology-based similarity measures developed during the last decade, nor any intrinsic IC model. WNSim has been used in a series of papers on word similarity by Patwardhan and Pedersen [93,96], and it has been extended in order to support the UMLS biomedical ontology, thus becoming an independent Perl software library called UMLS::Similarity [78], which is used in a WSD evaluation by McInnes et al. [77]. On the other hand, Harispe et al. [47] introduce the aforementioned SML library, which is the largest semantic measures library. SML is an ongoing project whose v0.9 version implements most classic path-based and IC-based similarity measures as well as several intrinsic IC models; however, it does not include most ontology-based similarity measures and intrinsic IC models developed during the last decade, as shown in Tables 2 and 3. However, SML includes direct support to import OWL and other significant biomedical ontologies such as GO, MeSH and SNOMED-CT. In addition, SML includes several most significant groupwise and pairwise GO-based semantic similarity measures, as

Table 2
Ontology-based semantic similarity measures implemented by the main publicly available software libraries based on WordNet.

Gloss-based similarity measures	WNSim	WS4J	SML	WNetSS	HESML
Banerjee and Pedersen (2003) [7]	X	X			
Patwardhan and Pedersen (2006) [93]	X	X			
Path-based and taxonomy-based measures	WNSim	SML	SML	WNetSS	HESML
Rada et al (1989) [107]	X	X	X	X	X
Wu and Palmer (1994) [130]	X	X	X	X	X
Hirst and St. Onge (1998) [51]	X	X			
Leacock and Chodorow (1998) [65]	X	X		X	X
Stojanovic et al. (2001) [123]			X		
Pekar and Staab (2002) [100]			X		
Li et al (2003) [69], strategy 3				X	X
Li et al (2003) [69], strategy 4					X
Liu et al. (2007) [71]				X	
Pedersen et al (2007) [98]					X
Al-Mubaid and NGuyen (2009) [3]				X	X
Kyogoku et al. (2011) [54]			X		
Hao et al. (2011) [45]				X	
Hadj Taieb et al (2014) [43], sim1				X	X
Hadj Taieb et al (2014) [43], sim2				X	X
IC-based similarity measures	WNSim	WS4J	SML	WNetSS	HESML
Resnik (1995) [108]	X	X	X	X	X
Jiang and Conrath (1997) [52]	X	X	X	X	X
Lin (1998) [70]	X	X	X	X	X
Li et al (2003) strategy 9 [69]					X
Schlicker et al. [116] (GO-based)			X		
Zhou et al (2008) [134]				X	X
Pirró and Seco (2008) [105]				X	X
Pirró and Euzenat (2010) [104], FaITH					X
Garla and Brandt (2012) [39]					X
Meng and Gu (2012) [81]				X	X
Meng et al (2014) [83]					X
Gao et al (2015) [38], strategy 3				X	X
Lastra and García (2015) [57], weighted J&C					X
Lastra and García (2015) [57], cos J&C					X
Lastra and García (2015) [57], cosw J&C					X
Feature-based similarity measures	WNSim	WS4J	SML	WNetSS	HESML
Tversky (1977) [124]				X	
Rodríguez and Egenhofer (2003) [110]				X	
Petrakis et al. (2006) [102]				X	
Sánchez et al (2012) [115]					X

well as a well-supported website and community forum. Thus, SML is currently the most complete and versatile software library reported in the literature. However, there are many other libraries and tools exclusively focused on Gene Ontology (GO), as detailed by Mazandu et al. [76], which should be considered in this specific domain. In addition to the aforementioned Tables 2 and 3, which summarize the methods implemented by the software libraries analyzed herein, Table 4 compares the programming languages and ontologies supported by them.

Finally, we have the WNetSS semantic measures library introduced recently by Aouicha et al. [15], which is based on an off-line pre-processing and caching in a MySQL server of WordNet, as well as all WordNet-based topological features and implemented IC models. As we mentioned previously in Section 1.1.1, the caching strategy used by WNetSS severely impacts its performance and scalability. In addition, WNetSS exhibits two other significant extensibility drawbacks which prevent its use for researching and prototyping of new methods, as follows: (1) the current distribution of WNetSS does not include its source files, thus, their architecture, representation model for taxonomies and implementation details are missing; and (2) the current WNetSS version does not allow any type of functional extension, such as including a new taxonomy parser, as well as a new semantic similarity library or IC model. Finally, despite one of the main motivations of WNetSS being to provide a software implementation for the most recent methods, looking at Tables 2 and 3, you can see that WNetSS [15] neither implements nor cites many recent similarity measures and IC models reported in the literature.

2.4. Potential applications in Information Systems

Another interesting field of application of the family of ontology-based similarity measures is the problem of business process modeling as detailed below. A very old problem in business process management is the construction and analysis of concept maps that model business processes. Mendling et al. [80] study the current practices in the activity labeling of business processes, whilst Dijkman et al. [32] propose a similarity metric between business process models based on an ad-hoc semantic similarity metric between words in the node labels and attributes, as well as the structural similarity encoded by the concept map topology. Likewise, Leopold et al. [68] propose an automatic refactoring method of activity labels in business process modeling based on the automatic recognition of labeling styles, and Leopold et al. [67] propose the inference of suitable names for business process models automatically. Finally, Montani and Leonardi [89] introduce a framework for the retrieval and clustering of process models based on a semantic and structural distance between models. It is clear that a notion of semantic similarity between components of the models underlies most tasks on process modeling in the latter

Table 3
Intrinsic and corpus-based IC models implemented by the main publicly available software libraries based on WordNet. The above list represents, to the best of our knowledge, all IC models reported in the literature. (*) The Aouicha et al. [16] IC model is implemented in HESML; however, this latter IC model has not yet been evaluated because several missing details need to be clarified by the authors, as described in HESML source code [60].

Corpus-based IC models	WNSim	WS4J	WNetSS	WNetSS	HESML
Resnik corpus-based (1995) [108][109]	X	X	X		X
Lastra & García (2015) [56], CPCorpus					X
Lastra & García (2016) [58], CPRefCorpus					X
Intrinsic IC models	WNSim	WS4J	SML	WNetSS	HESML
Seco et al (2004) [119]			X	X	X
Blanchard et al (2008) [18], IC_g					X
Zhou et al (2008) [133]			X	X	X
Sebti and Barfroush (2008) [118]				X	X
Sánchez et al (2011) [114]			X	X	X
Sánchez et al (2012) [113]					X
Meng et al (2012) [82]				X	X
Harispe (2012) [47]			X		X
Yuan et al (2013) [131]					X
Hadj Taieb et al (2014) [42]				X	X
Adhikari et al (2015) [1]					X
Aouicha et al (2016) [4]					X
Aouicha et al (2016) [16]*				X	X
Harispe et al. (2016) [46]					
Intrinsic IC models for relatedness measures					
Seddiqui and Aono [120]					
Pirró and Euzenat [104]					
IC models introduced by Lastra-Díaz and García-Serrano (2015) [56]					
CondProbHyponyms					X
CondProbUniform					X
CondProbLeaves					X
CondProbCosine					X
CondProbLogistic					X
IC models introduced by Lastra-Díaz and García-Serrano (2016) [58]					
CondProbRefHyponyms					X
CondProbRefUniform					X
CondProbRefLeaves					X
CondProbRefCosine					X
CondProbRefLogistic					X
CondProbCosineLeaves					X
CondProbRefLogisticLeaves					X
CondProbRefLeavesSubsumerRatio					X

Table 4
Further features of the main publicly available semantic software libraries based on WordNet.

Features	WNSim	WS4J	SML	WNetSS	HESML
Programming language	Perl	Java	Java	Java	Java
Source files availability	public	public	public	no	public
Ongoing development	no	no	yes	yes	yes
Supported ontology file formats: own parser (own) / external parser					
WordNet	own	own	own	extJWNL	own
OWL			own		
GO			own		
MeSH			own		
SNOMED			own		
RDF triples files			own		

semantic-aware applications. Thus, we argue herein that many of these methods could potentially benefit from the use of ontology-based semantic similarity measures.

3. The HESML software library

HESML V1R2 [60] is distributed as a Java class library (*HESML-V1R2.jar*) plus a test driver application (*HESMLclient.jar*), which have been developed using NetBeans 8.0.2 for Windows, although it has been also compiled and evaluated on Linux-based platforms using the corresponding NetBeans versions. *HESML* V1R2 is freely distributed for any non-commercial purpose under a Creative Commons By-NC-SA-4.0 license[2] recognized by citing the present work, whilst the *commercial use* of the similarity measures introduced in [57], as well as part of the intrinsic IC models introduced in [56] and [58], is protected by a patent application [58]. HESML is currently being evaluated by Castellanos et al. [22] in a taxonomy recovering task from DBpedia based on Formal Concept Analysis (FCA) methods like the proposed ones in [21]. HESML V1R2 significantly improves the performance of the HESML V1R1 version [59] which was released on September 7 2016 with the original submission of this work.

In order to make the experimental work with HESML easier, as well as supporting the reproducible experiments detailed in Section 4, HESML is distributed as a self-contained development and testing platform including the set of complementary resources shown in Table 22 in appendix, which includes three different WordNet[3] versions, a WordNet-based frequency file dataset developed by Ted Pedersen [95], and the five most significant word similarity benchmarks. For this reason, any user of HESML must fulfill the licensing terms of these third-party resources by recognizing their authorship accordingly.

[2] https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode .
[3] https://wordnet.princeton.edu/wordnet/license/ .

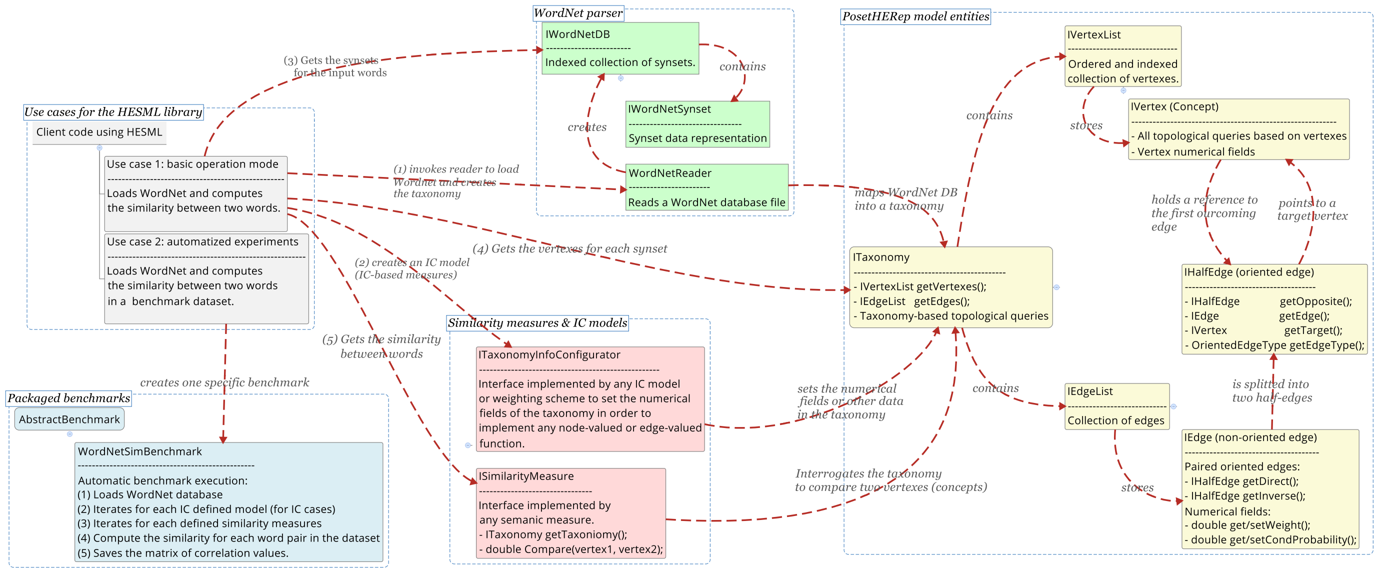
WordNet parser
IWordNetDB
Indexed collection of synsets.
contains
IWordNetSynset
Synset data representation
creates
WordNetReader
Reads a WordNet database file
(3) Gets the synsets for the input words
Use cases for the HESML library
Client code using HESML
Use case 1: basic operation mode
Loads WordNet and computes the similarity between two words.
Use case 2: automatized experiments
Loads WordNet and computes the similarity between two words in a benchmark dataset.
(1) invokes reader to load Wordnet and creates the taxonomy
(2) creates an IC model (IC-based measures)
(4) Gets the vertexes for each synset
(5) Gets the similarity between words
creates one specific benchmark
Packaged benchmarks
AbstractBenchmark
WordNetSimBenchmark
Automatic benchmark execution:
(1) Loads WordNet database
(2) Iterates for each IC defined model (for IC cases)
(3) Iterates for each defined similarity measures
(4) Compute the similarity for each word pair in the dataset
(5) Saves the matrix of correlation values.
Similarity measures & IC models
ITaxonomyInfoConfigurator
Interface implemented by any IC model or weighting scheme to set the numerical fields of the taxonomy in order to implement any node-valued or edge-valued function.
ISimilarityMeasure
Interface implemented by any semanic measure.
- ITaxonomy getTaxionomy();
- double Compare(vertex1, vertex2);
PosetHERep model entities
maps WordNet DB into a taxonomy
contains
IVertexList
Ordered and indexed collection of vertexes.
stores
IVertex (Concept)
- All topological queries based on vertexes
- Vertex numerical fields
holds a reference to the first outcoming edge
points to a target vertex
ITaxonomy
- IVertexList getVertexes();
- IEdgeList getEdges();
- Taxonomy-based topological queries
IHalfEdge (oriented edge)
- IHalfEdge getOpposite();
- IEdge getEdge();
- IVertex getTarget();
- OrientedEdgeType getEdgeType();
sets the numerical fields or other data in the taxonomy
contains
IEdgeList
Collection of edges
is splitted into two half-edges
IEdge (non-oriented edge)
Paired oriented edges:
- IHalfEdge getDirect();
- IHalfEdge getInverse();
Numerical fields:
- double get/setWeight();
- double get/setCondProbability();
stores
Interrogates the taxonomy to compare two vertexes (concepts)

Fig. 1. HESML architecture showing main objects and interfaces. The core HESML component is the half-edge taxonomy representation defined by the yellow entities. Red entities in the block entitled 'Similarity measures & IC models' represent the two interfaces that should be implemented to define new IC models and similarity measures. All the HESML objects are provided as Java interfaces, being instanced by factory objects not represented in the figure. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

HESML V1R2 currently supports the WordNet taxonomy, most ontology-based similarity measures and all the IC models for concept similarity reported in the literature with the only exception of the IC models introduced by Harispe et al. [46], although the latter IC model could be included in future versions. In addition to the aforementioned IC models [46], Seddiqui and Aono [120] and Pirró and Euzenat [104] propose two further intrinsic IC models not implemented by HESML which are based on the integration of all types of taxonomical relationships, and thus especially designed for semantic relatedness measures. In addition, we plan to provide ongoing support for further ontologies such as Wikidata [126] and the Gene Ontology (GO) [5] among others, as well as further similarity and relatedness measures. On the other hand, the HESML architecture allows further similarity measures, IC models and ontology readers to be developed easily. We also urge potential users to propose further functionality. In order to remain up to date on new HESML versions, as well as asking for technical support, we invite the readers to subscribe to the HESML forum detailed in Table 8.

3.1. Software Architecture

The HESML software library is divided into four functional blocks as follows: (1) *PosetHERep* model objects shown in yellow in Fig. 1; (2) abstract interfaces implemented by the IC models or weighting schemes (*ITaxonomyInfoConfigurator)* and all the taxonomy-based similarity measures (*ISimilarityMeasure*) shown in red; (3) ontology readers shown in green; and (4) a family of automatized benchmarks shown in blue, which allow reproducible experiments on ontology-based similarity measures, IC models and word similarity benchmarks with different WordNet versions to be easily implemented, as well as computing and saving the results matrices with Pearson and Spearman correlation values. The automatized benchmarks allow the efficient and exact replication of the experiments and data tables included in the aforementioned works introduced by Lastra-Díaz and García-Serrano. These latter automatized benchmarks can be defined in an XML-based file format, which allows the definition of large experimental surveys without any software coding. All HESML objects are provided as private classes by implementing a set of Java interfaces, thus, they can only be instantiated by invoking the proper factory classes. All the similarity measures, IC models or weighting schemes are invoked with a reference to the base taxonomy object (*ITaxonomy*) as an input argument, which provides a complete set of queries to retrieve all types of information and topological features. The children, parent, subsumed leaves, ancestor and descendant (hyponym) sets are computed on-the-fly, while the nodes and edges hold the IC values and weights respectively. Any IC model or weighting scheme is defined as an abstract taxonomy processor whose main aim is to annotate the taxonomy with the proper IC values, edge-based weights, concept probabilities or edge-based conditional probabilities. The node-based and edge-based data is subsequently retrieved by the ontology-based semantic similarity measures in their evaluation.

3.2. The PosetHERep representation model for taxonomies

PosetHERep is a new and linearly scalable representation model for taxonomies which is introduced herein for the first time. *PosetHERep* is based on our adaptation of the well-known half-edge representation in the field of computational geometry [19], also known as a double-connected edge list [17, § 2.2], in order to efficiently represent and interrogate large taxonomies.

PosetHERep model is the core component of the *HESML* architecture, it being the mainly responsible for their performance and scalability. Fig. 2 shows the core idea behind the *PosetHERep* representation model: all the outcoming and incoming oriented edges (half-edges) from any vertex are connected in such a way that their connection induces a cyclic ordering on the set of adjacent vertexes. Given any single or multiple-root taxonomy $\mathcal{C} = (C, \leq_C)$, we can define its associated graph $G = (V, E)$ in the usual way, in which every concept $c_i \in C$ is mapped onto a vertex $v_i \in C$ and every order relationship between a parent concept and their children is mapped onto an oriented edge, hereinafter called as a half-edge. The core component of the *PosetHERep* model is the *neighbourhood iteration loop* algorithm detailed in Table 5 and three half-edge-valued functions as follows: (1) the *Target* function returns the vertex which the oriented edge points, (2) the *Next* function returns the next outcoming half-edge for each incoming half-edge to any base vertex, and (3) the *Opposite* function returns the opposite and paired half-edge. *PosetHERep* is based on the following *topological*

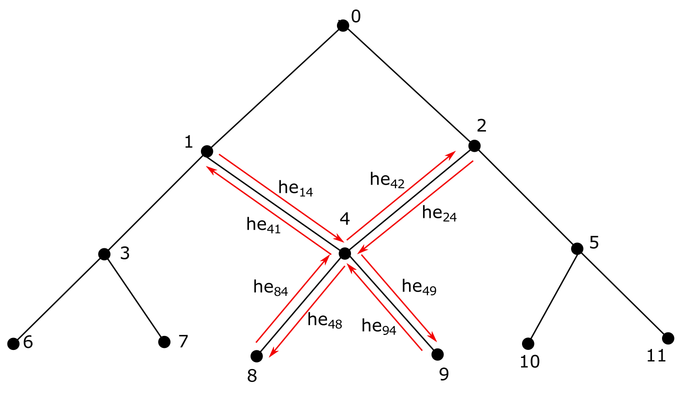

Fig. 2. *PosetHERep*: half-edge representation around the vertex (concept) with id = 4. Every edge is split into two paired and opposite oriented (half) edges. Given the first outcoming half-edge he_{ab} from any vertex a, the set of adjacent vertexes is recovered in linear time through a cyclic iteration, as described by Algorithm 1.

Table 5
Iteration loop from a base vertex in order to recover its adjacent vertexes.

Algorithm 1	Neighbourhood iteration loop
Input:	a base vertex v
Output:	an ordered list *adjVertexes* of adjacent vertexes
1:	*IVertexList adjVertexes*;
2:	*IHalfEdge loop* $= v.firstOutComingEdge$;
3:	do
4:	{
5:	*adjVertexes.Add*(*loop.Target*);
6:	*loop* $=$ *loop.Opposite.Next*;
7:	} while ($loop\ != v.firstOutComingEdge$);

consistency axiom: all the incoming and outcoming half-edges of any vertex are connected in such a way that a full cycle of the *neighbourhood iteration loop* returns the set of adjacency vertexes on any taxonomy vertex. The *HESML* method that inserts the vertexes onto the taxonomy is mainly responsible for the verification of the latter axiom.

The *PosetHERep* model allows most topological queries to be answered in linear time, providing a very efficient implementation for all the graph-traversing algorithms, such as the computation of the depth of the vertexes, ancestor and descendant sets, subsumed leaf sets, and the length of the shortest path between vertexes, among others. Given any taxonomy with an associated graph $G = (V, E)$, it is easy to prove that the memory cost of its *HESML* representation is $O(k_1|V| + k_2|E|)$, in which the constants k_1 and k_2 are defined by the memory size of the vertex and edge attributes. Thus, in any large taxonomy with a small number of concepts with multiple parents we can assume $|V| \approx |E|$, which proves that *HESML* linearly scales with the number of concepts in the taxonomy.

Finally, in order to implement the *PosetHERep* representation model, you must define the behaviour and interface of the six objects shown in yellow in Fig. 1 (ITaxonomy, IVertex, IHalfEdge, IEdge, IVertexList, and IEdgeList), as well as the collection of eight algorithms introduced below. Because of the lack of space, we do not detail seven of these algorithms, thus, we refer the reader to the source code implementing them. The eight algorithms run in linear time as regards the size of the taxonomy, with the only exception being the shortest path algorithm 6. Apart from the output data structures filled by the algorithms detailed below, none of them demands caching or other intensive-memory structures for their implementation. For this reason, the aforementioned algorithms are computationally efficient and scalable.

Algorithm 1. *Neighbourhood iteration loop*. Table 5 details this algorithm, which encodes all the adjacency relationships within the taxonomy. The current *PosetHERep* model only supports 'is-a' relationships, because it only supports two types of half-edges: 'SubClassOf' and 'SuperClassOf'. For this reason, the current HESML version is only able to represent 'ìs-a' taxonomies. However, the extension of the *PosetHERep* model to manage any type of ontological relationship is straightforward. Thus, we plan to extend its representation capabilities in future versions to include any type of semantic relationship between concepts within an ontology. In addition, *PosetHERep* could be extended to represent many other types of semantic graphs. We also call this algorithm a *vertex iteration loop*, and it is extensively used by most algorithms detailed in this section. Indeed, you can see this piece of code in the software implementation of the aforementioned methods in HESML. The iteration loop runs in linear time, it being the time proportional to the number of adjacent vertexes.

Algorithm 2. *Insertion of a vertex in the taxonomy*. This algorithm inserts a new vertex into the taxonomy, as detailed in the source code of the *Taxonomy.addVertex()* function. The method links the vertex to its parent vertexes in order to satisfy the aforementioned *topological consistency axiom*. Once the vertex has been inserted into the taxonomy, it can be directly interrogated without any further inference process, such as that required by other libraries like SML. The method runs in linear time, it being the time proportional to the number of adjacent vertexes.

Algorithm 3. *Retrieval of the ancestor set of a vertex*. This algorithm retrieves the ancestor set of any vertex within the taxonomy without caching, as detailed in the source code of the *Vertex.getAncestors()* function. The algorithm climbs up the taxonomy by traversing the 'SubClassOf' oriented edges in each local vertex iteration loop. The method runs in linear time, it being the time proportional to the maximum depth of the base vertex.

Algorithm 4. *Retrieval of the descendant set (hyponyms) of a vertex*. This algorithm retrieves the descendant set of any vertex within the taxonomy without caching, as detailed in the source code of the *Vertex.getHyponyms()* function. The algorithm climbs down the taxonomy by traversing the 'SuperClassOf' oriented edges in each local vertex iteration loop. The method runs in linear time, it being the time proportional to the difference between the maximum depth of the taxonomy and the base vertex.

Algorithm 5. *Retrieval of the set of subsumed leaves of a vertex*. This algorithm retrieves the set subsumed leaves by any vertex within the taxonomy without caching, as detailed in the source code of the *Vertex.getSubsumedLeaves()* function. The algorithm is identical to the method for retrieving the descendant set with the exception that this method only selects the leaf vertexes, instead of all descendant vertexes. It shares the same computational complexity as algorithm 4.

Algorithm 6. *Shortest path*. This algorithm computes the length of the shortest weighted or unweighted path between two vertexes in the taxonomy, as detailed in the source code of the *Vertex.getShortestPathDistanceTo()* function. The method is a classic Dijkstra algorithm based on a min-priority queue [25,79] and the aforementioned PosetHERep vertex iteration loop in order to efficiently traverse the graph. Despite our implementation of the Dijkstra algorithm being very efficient in comparison with other semantic measures libraries, it is still a general-graph method approach with an exponential time complexity.

Algorithm 7. *Minimum depth computation.* This algorithm computes the minimum depth of the vertex, which is defined as the length of the shortest ascending path from the vertex to the root, as detailed in the source code of the *Vertex.computeMinDepth()* function. The algorithm is divided into two steps: (1) retrieval of the ancestor set, and (2) computation of the shortest ascending path using a modified Dijkstra algorithm constrained to the ancestor set. The core idea of speeding up this algorithm is to reduce the search space for the shortest path algorithm to the ancestor set, which is very efficiently retrieved using algorithm 3. The method runs in linear time, it being the time proportional to the maximum depth of the base vertex.

Algorithm 8. *Maximum depth computation.* This algorithm computes the maximum depth of the vertex, which is defined as the length of the longest ascending path from the vertex to the root, as detailed in the source code of the *Vertex.computeMaxDepth()* function. This algorithm is identical to the algorithm 7, but in this case it computes the longest ascending path from the vertex to the root.

3.3. Software Functionalities

HESML V1R2 includes the implementation of all the ontology-based similarity measures shown in Table 2, all the IC models shown in Table 3, a set of automatized benchmarks and a reader of WordNet databases. The set of IC models included in *HESML* represents most known intrinsic and corpus-based IC models based on WordNet reported in the literature. The library includes its own WordNet parser and in-memory database representation, it being fully independent of any other software library. In addition, HESML defines the *AbstractBenchmark* and *WordnetSimBenchmark* classes in order to provide a family of automatized word similarity benchmarks based on WordNet, as well as an input XML-based reproducible experiment file format which allows all the reproducible experiments detailed in Section 4 and the *WNSimRep v1* dataset to be easily replicated with no software coding.

3.4. Impact

In addition to providing a larger collection of ontology-based similarity measures and intrinsic IC models than other publicly available software libraries, *HESML* provides a more efficient and scalable representation of taxonomies for the prototyping, development and evaluation of ontology-based similarity measures. These aforementioned features convert HESML into an open platform to assist the research activities in the area, such as: (1) the development of large experimental surveys, (2) the fast prototyping and development of new methods and applications, (3) the replication of previous methods and results reported in the literature such as in this work, and (4) the dissemination and teaching of ontology-based similarity measures and IC models.

The functionality and software architecture of *HESML* allow the efficient and practical evaluation of large word similarity benchmarks such as SimLex [50] and ontology-based similarity measures based on the length of the shortest path, whose implementation in other software libraries requires a high computational cost that prevents their evaluation in large experimental surveys [58] and datasets. Thus, *HESML* is an essential tool for allowing the fast prototyping and evaluation of new path-based similarity measures on weighted taxonomies or other complex taxonomical features, such as the measures introduced in [57].

Lastra-Díaz and García-Serrano are currently carrying-out a very active research campaign into ontology-based similarity measures and IC models based on *HESML*. Thus, it is expected that HESML functionality will grow accordingly. Finally, because of the growing interest in the integration of ontology-based similarity measures in many applications in the fields of NLP, IR, the Semantic Web and bioengineering, especially genomics, we expect that *HESML* will be helpful and interesting to a larger audience.

3.5. Illustrative examples of use

The HESMLclient.java source code file includes a set of sample functions in order to show the functionality of the library as shown in Table 6, which are listed in the function *SampleExperiments()*. All source files are well documented and extensively commented on, in addition to providing a Javadoc documentation. Thus, we think that a careful reading of the source code examples, as well as the understanding of the software architecture detailed in Fig. 1 and the extensibility procedures detailed in Section 3.6, should be enough to use HESML to its best advantage. Next, we highlight two examples of use of HESML, whilst the next subsection explains how to extend the functionality of the library:

- *Reproducing previous methods and experiments*. We refer the reader to the sample functions in Table 6.
- *Running large experimental surveys*. In addition to checking the aforementioned sample functions, we refer the reader to the Section 4 in which a set of large reproducible experiments is detailed.

3.6. Extending the library

One of the main goals of HESML is to replicate previous methods, as well as facilitating the prototyping and development of new methods. The main extensibility axes of the library are the development of new similarity measures and IC models, as well as further ontology parsers. We detail how to carry-out these functionality extensions as follows:

- *Developing and prototyping a new similarity measure.* In order to design a new ontology-based similarity measure, the users must create and register a new class by implementing the *ISimilarityMeasure* interface. The steps to create a new similarity measure are as follows: (1) create a new measure class in the *hesml/measures/impl* namespace, which extends the *SimilaritySemanticMeasure* abstract class and implements the *ISimilarityMeasure* interface; (2) include a new type of measure in the *SimilarityMeasureType.java* enumeration; and (3) register the creation of the new measure in the *getMeasure()* method implemented by the factory class defined in the *hesml/measures/impl/MeasureFactory.java* source file.
- *Developing and prototyping a new IC model.* In order to design a new intrinsic/corpus-based IC model, the users must create and register a new class implementing the *ITaxonomyInfoConfigurator* interface. The steps to create a new intrinsic IC model are as follows: (1) create a new IC model class in the *hesml/configurators/icmodels* namespace, which extends the *AbstractICmodel* class and implements the *ITaxonomyInfoConfigurator* interface; (2) include a new intrinsic IC model type in the *IntrinsicICModelType.java/ CorpusBasedICModelType.java* enumerations; and (3) register the creation of the new IC model either the *getIntrinsicICmodel()* or *getCorpusICmodel()* methods implemented by the factory class defined in the *hesml/configurators/icmodels/IntrinsicICFactory.java* source file.
- *Developing a new taxonomy reader.* Any taxonomy reader must be able to read a taxonomy file and return an instance of an *ITaxonomy* object. You can use the implementation of the WordNet reader in the *taxonomyreaders/wordnet/impl* namespace as example.

Table 6
Examples of use included in the HESMLclient.java source code file in order to show the functionality of HESML.

HESMLClient method	Description
testAllSimilarityBenchmarks	Runs different types of word similarity benchmarks.
testMultipleICmodelsMultipleICmeasuresBenchmarks	Runs a cross-evaluation of IC models and IC-based similarity measures.
testSingleNonICbasedMeasure	Runs the evaluation of a single non IC-based similarity measures.
testSingleICSimMeasureMultipleICmodels	Runs the evaluation of a single IC-based similarity measure with multiple intrinsic IC models.
testSingleICSimMeasureSingleICmodel	Runs the evaluation of a single IC-based similarity measure with single intrinsic IC models.
testWordPairSimilarity	Shows the computation of the similarity between two words by using the noun database of WordNet and any similarity measure.
testSingleICmodelMultipleICbasedMeasures	Runs the evaluation of a single intrinsic IC model with multiple IC-based similarity measures.
testCorpusBasedSimilarityBenchmarks	Runs the evaluation of multiple corpus-based IC models with multiple IC-based similarity measures.
buildWNSimRepFiles	Builds the WNSimRep v1 dataset.
createTestTaxonomy	This function shows how to create a tree-like taxonomy with the number of vertexes defined by the input parameter. Thus, it shows what should be done by any new ontology parser in order to populate a HESML taxonomy.

Table 7
Complementary Mendeley datasets published with the current work.

Dataset	Content description
HESML V1R2 distribution package [60]	Java source files and NetBeans projects. WordNet 2.1, 3.0 and 3.1 databases. Pedersen's WordNet-based frequency files. Word similarity benchmarks enumerated in table 1.
WordNet-based word similarity reproducible experiments [64]	A ReproZip reproducible experiment file which allows the experimental surveys on WordNet-based word similarity introduced in [57], [56] and [58] to be reproduced, as well as a Zip file with all the raw output files for an easy verification.
WNSimRep v1 dataset [63]	A framework and replication dataset for ontology-based semantic similarity measures and IC models.
HESML_VS_SML [61]	Set of benchmarks introduced herein which evaluate and compare HESML, SML and WNetSS.

Table 8
Summary of technical and legal information of the HESML software library.

HESML source code data	Description
Current code version.	V1R2
Legal Code License.	Creative Commons By-NC-SA 4.0
Permanent code repository used for this version.	http://dx.doi.org/10.17632/t87s78dg78.2
GitHub repository	https://github.com/jjlastra/HESML.git
Software code languages and tools.	Java 8, Java SE DevKit 8, NetBeans 8.0 or higher
Compilation requirements and operating systems.	Java SE Dev Kit 8, NetBeans 8.0 or higher and any Java-compliant operating system.
Documentation and source code examples	This work and the sample source code in the HESMLclient program.
Community forum for questions.	hesml+subscribe@googlegroups.com, hesml+unsubscribe@googlegroups.com

4. The Reproducible Experiments

The aim of this section is to introduce a set of detailed experimental setups in order to exactly replicate the methods and experiments introduced by Lastra-Díaz and García-Serrano in [56–58], whose contributions were stated in the introduction.

4.1. Experimental setup and complementary datasets

We follow the same experimental setup as that detailed in [56] and [58], including the same datasets, preprocessing steps, evaluation metrics, baselines, management of polysemic words and reporting of the results. All the experiments compute the Pearson and Spearman correlation metrics for a set of ontology-based similarity measures on each word similarity benchmark shown in Table 22, as detailed in [56]. Table 7 details the four complementary Mendeley datasets which are distributed in the current work.

4.2. Obtaining and compiling HESML

Table 8 shows the technical information required to obtain and compile the *HESML* source code and run the experiments detailed in Table 11. There are two different ways of obtaining the *HESML* source code: (1) by downloading the current version from the permanent Mendeley Data link [60]; and finally, (2) by downloading it from its GitHub repository detailed in Table 8.

Once the source code package has been downloaded or extracted onto your hard drive, the project will have the following folder structure:

1. *HESML_Library*. The root folder of the project.
2. *HESML_Library\HESML*. This folder is the main software library folder containing the NetBeans project and HESML source code. Below this folder you find the *dist* folder which contains the *HESML-V1R2.jar* distribution file generated during the compilation.
3. *HESML_Library\HESMLclient*. This folder contains the source code of the HESMLclient console application. The main aim of the *HESMLclient.jar* application is to provide a collection of sample functions in order to show the HESML functionality, as well as running the collection of reproducible experiments.
4. *HESML_Library\PedersenICmodels*. This folder contains the full WordNet-InfoContent-3.0 collection of WordNet-based frequency files created by Ted Pedersen [95]. The file names denote the corpus used to build each file. The readme file details the method used to build the frequency files, which is also detailed in [97].
5. *HESML_Library\ReproducibleExperiments.* This folder contains three subfolders with the reproducible experiment files shown in Table 11, as well as a XML-schema file called *WordNet-BasedExperiments.xsd*, which describes the syntax of all XML-based experiment files (*.exp), and the *All_paper_tables.exp* file with the definition of all the reproducible experiments shown in Table 11. All files have been created with the XML Spy editor.

Table 9
Configuration of the computers used to reproduce the accompanying set of reproducible experiments, and their running times on the main reproducibility experiments.

Experimental platform	Operating system	CPU	RAM
Ubuntu-base (2011)	Ubuntu MATE 16.04 LTS	Intel Pentium B950 @ 2.10 GHz	4 Gb
Windows-base (2015)	Windows 8.1x64	Intel Core i7-5500U @ 2.40 GHz	8 Gb

In addition, this folder also contains the *RawOutputFiles* subfolder with all the raw output files shown in Table 11, and the *Post-scripts* folder containing the set of post-processing R scripts detailed in Table 12.

6. *HESML_Library\WN_datasets*. This folder contains a set of '*.csv' data files corresponding to the word similarity benchmarks shown in Table 22.
7. *HESML_Library\WordNet-2.1*. This folder contains the database files of WordNet 2.1.
8. *HESML_Library\WordNet-3.0*. This folder contains the database files of WordNet 3.0.
9. *HESML_Library\WordNet-3.1*. This folder contains the database files of WordNet 3.1.

In order to compile *HESML*, you must follow the following steps:

1. Install Java 8, Java SE Dev Kit 8 and NetBeans 8.0.2 or higher in your workstation.
2. Launch NetBeans IDE and open the *HESML* and *HESMLclient* projects contained in the root folder. NetBeans automatically detects the presence of a *nbproject* subfolder with the project files.
3. Select *HESML* and *HESMLclient* projects in the project treeview respectively. Then, invoke the 'Clean and Build project (Shift + F11)' command in order to compile both projects.

4.3. *Running the experiments*

Table 11 shows the full collection of reproducible experiment files, as well as the corresponding output files that will be generated in order to reproduce the results reported in [57], [56] and [58] respectively.

There are two ways of running the accompanying reproducible experiments: (1) by compiling *HESML* and running the *HESMLclient* program with any input experiment file shown in Table 11, as detailed in Section 4.3.1; or (2) by running the *HESMLv1r1_reproducible_exps.rpz* reproducible experiment file [64] based on ReproZip, as detailed in Section 4.3.4. The name of the reproducible experiment files in Table 11 encodes the name of each corresponding table of results that is obtained as output, thus, the table of results that is reproduced. These experiment files reproduce most results reported in [56–58]. However, there are several summary tables in these aforementioned works that are not directly reproduced from the raw output files, thus, the post-processing of several output files is necessary to obtain these missing tables as detailed in Section 4.3.3.

4.3.1. *Running the experiments with* HESMLclient

Once you have compiled the *HESML* and *HESMLclient* projects as detailed in Section 4.2, you are ready to run the reproducible experiments as detailed below. The original *HESMLclient* source code is defined to fetch the required input files from the folder structure of *HESML*. Thus, you only need to follow the steps below:

1. Open a Linux or Windows command prompt in the *HESML_Library\HESMLclient* directory.
2. Run the following command using any reproducible experiment file shown in Table 11:

Table 10
Running times for the main reproducible experiments.

PC name	EAAI_all_tables	KBS_all_tables	AI_all_tables
Ubuntu-base	13491 min ≈ 9.37 days	38 s	16 days
Windows-base	–	25 s	–

$prompt:> java -jar dist\HESMLclient.jar ..\ReproducibleExperiments \<anyfile.exp>.

3. You must run the latter command for each experiment file defined in the aforementioned tables. Optionally, you can run all the experiments automatically by loading any summary file in step 2 above as follows: (1) *EAAI_all_tables.exp,* (2) *KBS_all_tables.exp*, (3) *AI_all_tables.exp,* or (4) *All_paper_tables.exp.* This latter file contains all the experiments shown in Table 11. Table 10 shows the running times for the latter reproducible experiments on the two experimental platforms detailed in Table 9.

Finally, the *WNSimRepv1* dataset [63] can be computed automatically by running the command in step 4 below. The program automatically creates and stores all *WNSimRepv1* data files in the output directory. If the output directory does not exist then it is automatically created.

4. *$prompt:> java -jar dist\HESMLclient.jar -WNSimRepV1 <outputdir>*

4.3.2. *System requirements and performance evaluation*

The reproducible experiments detailed in the previous section have been reproduced by the authors in two different experimental platforms shown in Table 9, which are defined by an old low-end laptop called *Ubuntu-base* and a more recent professional laptop called *Windows-base*. The *Ubuntu-base* workstation sets the minimal system requirements in order to reproduce the experiments detailed in previous section, as well as the ReproZip package introduced in Section 4.3.4. Table 10 shows the running times for the main reproducible experiments on the two experimental platforms.

4.3.3. *Processing of the result files*

The running of each experiment file in Table 11 produces one or two comma-separated files (*.csv) with the values separated by a semicolon. The first column in Table 11 shows the number of the table in which the output data computed by each reproducible experiment file (*.exp) appears. All output files are saved in the same folder as their corresponding input experiment files.

Many output files detailed in Table 11 need certain post-processing in order to match the tables shown in the papers exactly. In order to automate this post-processing, we provide the set of R scripts detailed in Table 12. These scripts take the raw output files generated by the experiments in Table 11 and produce the final assembled tables as shown in [56–58], as well as Figs. 2 and 3 showing the interval significance analysis in [56]. The output files shown in the second column in Table 12 are the only files requiring post-processing, the remaining raw output files match the tables shown in thee aforementioned works exactly. In order to run

Table 11
Collection of reproducible experiment files for the data tables reported in [57], [56] and [58]. The first column shows the table corresponding to the data generated in the output file. The column entitled 'Measures' denotes the type of similarity measures evaluated by each experiment. Each reproducible experiment file is defined by a XML-based text file with extension (.exp), which can contain the definition of one or more reproducible experiments. Thus, some experiment files produce one output file whilst others produce two output files that must be merged in order to reproduce the original data tables in the papers exactly. Because of the computational cost of the experiments reported in [58], the experiment files corresponding to the latter work generate a single output file containing the Pearson and Spearman correlation metrics that appear separately in the aforementioned work. Thus, it is necessary to split and arrange the columns of the output data tables in order to reproduce the Pearson and Spearman metrics reported in [58] exactly.

Tables	WN	Datasets	IC models	Measures	Metrics	Reproducible experiment file	Output files
Reproducible experiments for the results reported in [57]							
4	All	All	–	Non IC	Pearson	EAAI_table4_nonICmeasures.exp	EAAI_table4_nonICmeasures.csv
5	2.1	RG65, $P\&S_{full}$	intrinsic	IC-based	Pearson	EAAI_table5_RG65_PS.exp	EAAI_table5_RG65.csv EAAI_table5_PS.csv
6	3.0	RG65	all	IC-based	Pearson	EAAI_table6_RG65.csv	EAAI_table6_RG65.csv
7	3.0	$P\&S_{full}$	all	IC-based	Pearson	EAAI_table7_PS.csv	EAAI_table7_PS.csv
8	3.1	RG65, $P\&S_{full}$	intrinsic	IC-based	Pearson	EAAI_table8_RG65_PS.exp	EAAI_table8_RG65.csv EAAI_table8_PS.csv
All	3.0	All	all	all	Pea/Spea	EAAI_all_tables.exp	All output files above
Reproducible experiments for the results reported in [56]							
6	3.0	All	–	H. Taieb [43]	Pea/Spea	KBS_table6_Taieb.exp	KBS_table6_Taieb.csv
7	3.0	RG65	all	IC-based	Pea/Spea	KBS_table7_RG65.csv	KBS_table7_RG65.csv
8	3.0	MC28	all	IC-based	Pea/Spea	KBS_table8_MC28.exp	KBS_table8_MC28.csv
9	3.0	Agirre201	all	IC-based	Pea/Spea	KBS_table9_Agirre201.exp	KBS_table9_Agirre201.csv
10	3.0	$P\&S_{full}$	all	IC-based	Pea/Spea	KBS_table10_PS.exp	KBS_table10_PS.csv
11	3.0	SimLex665	all	IC-based	Pea/Spea	KBS_table11_SimLex665.exp	KBS_table11_SimLex665.csv
All	3.0	All	all	all	Pea/Spea	KBS_all_tables.exp	All output files above
Reproducible experiments for the results reported in [58]							
12	3.0	All	best	All	Pea/Spea	AI_table12.exp	AI_table12.csv
15,16	3.0	RG65	all	IC-based	Pea/Spea	AI_table15_16_RG65.exp	AI_table15_16_RG65.csv
17,18	3.0	MC28	all	IC-based	Pea/Spea	AI_table17_18_MC28.exp	AI_table17_18_MC28.csv
19,20	3.0	Agirre201	all	IC-based	Pea/Spea	AI_table19_20_Agirre201.exp	AI_table19_20_Agirre201.csv
21,22	3.0	$P\&S_{full}$	all	IC-based	Pea/Spea	AI_table21_22 PS.exp	AI_table21_22_PS.csv
23,24	3.0	SimLex665	all	IC-based	Pea/Spea	AI_table23_24_SimLex665.exp	AI_table23_24_SimLex665.csv
All	3.0	All	all	all	Pea/Spea	AI_all_tables.exp	All output files above

the scripts in Table 12, you need to setup the well-known R statistical program[4] in your workstation. Once R is installed, you need to install the 'BioPhysConnectoR' package, and follow the steps below:

1. Launch the R program
2. Select the menu option '*File->Open script*'. Then, load any R-script file contained in the *HESML_Library\Reproducible Experiments\Post-scripts* folder.
3. Edit the 'inputDir' variable at the beginning of the script in order to match the directory containing the raw output files onto your hard drive.
4. Select the menu option '*Edit->Run all*'. The final assembled tables will be saved in the input directory defined above, whilst the figures will be shown within R and saved as independent PDF files.

Table 12
Collection of R scripts in order to assemble several tables as shown in the three aforementioned works by Lastra-Díaz and García-Serrano, whose content is not directly obtained from the experimental raw output files. Load the script files in the same order below.

R script file	Post-processing output files and/or figures	
EAAI_final_tables.r	EAAI_final_table_4.csv	
AI_final_tables.r	AI_final_table_10.csv	AI_final_table_11.csv
	AI_final_table_12.csv	
	AI_final_table_15.csv	AI_final_table_16.csv
	AI_final_table_17.csv	AI_final_table_18.csv
	AI_final_table_19.csv	AI_final_table_20.csv
	AI_final_table_21.csv	AI_final_table_22.csv
	AI_final_table_23.csv	AI_final_table_24.csv
	AI_final_table_25.csv	AI_final_table_26.csv
KBS_final_tables.r	KBS_final_table_4.csv	KBS_final_table_6.csv
	KBS_final_table_6.csv	KBS_figure{2,3}.pdf

4.3.4. Running the ReproZip experiments

ReproZip is a virtualization tool introduced by Chirigati et al. [27], whose aim is to warrant the exact replication of experimental results onto a different system from that originally used in their creation. Reprozip captures all the program dependencies and is able to reproduce the packaged experiments on any host platform, regardless of the hardware and software configuration used in their creation. Thus, ReproZip warrants the reproduction of the experiments introduced herein in the long term.

The ReproZip program was used for recording and packaging the running of the *HESMLclient* program with all the reproducible experiments shown in Table 11 in the *HESMLv1r1_reproducible_exps.rpz* file available at [64]. This ReproZip file was generated by running Reprozip on the Ubuntu-base workstation, which was also used to run ReproUnzip based on Docker as detailed below. In order to set up and run the reproducible experiments introduced herein, you need to use ReproUnzip. ReproUnzip can be used with two different virtualization platforms: (1) Vagrant + VirtualBox, or (2) Docker. For a comparison of these two types of virtualization platform, we refer the reader to the survey introduced by Merkel [84], in which the author introduces Docker and compares it with classic Virtual Machines (VM) such as VirtualBox.

Our preferred ReproUnzip configuration is that based on Docker. For instance, in order to setup ReproUnzip based on Docker for Ubuntu, you should follow the detailed steps shown in Table 13, despite several steps possibly being unnecessary depending on your starting configuration. Once ReproUnzip and Docker have been successfully installed, Table 14 shows the detailed instructions to set up and run the reproducible experiments. Those read-

[4] https://www.r-project.org/.

Table 13
Detailed instructions on installing ReproUnzip with Docker for Ubuntu.

Step	Detailed setup instructions
1	sudo apt-get update
2	sudo apt-get install libffi-dev
3	sudo apt-get install libssl-dev
4	sudo apt-get install openssl
5	sudo apt-get install openssh-server
6	sudo apt-get install libsqlite3-dev
7	sudo apt-get install python-dev
8	sudo pip install reprouzip[all]
9	Docker for Ubuntu setup: follow the detailed instructions at https://docs.docker.com/engine/installation/linux/ubuntulinux/

Table 14
Detailed instructions on how to reproduce the packaged experiments once Reprounzip has been installed.

Step	Detailed experiment setup and running instructions
1	Setup the Reprounzip program onto any supported platform (Linux, Windows and MacOS) as detailed in the ReproZip setup page detailed in table.
2	Download the HESMLv1r1 reproducible exps.rpz from its Mendeley repository [64], as detailed in Table 8.
3	Open a command console in the directory containing the HESMLv1r1_reproducible_exps.rpz file and executes the two commands below: (1) reprounzip docker setup HESMLv1r1_reproducible_exps.rpz docker_folder (2) reprounzip docker run docker_folder

Table 15
The first instruction shows a list with the output files generated by the experiments, whilst the second instruction extracts all the output files from the container and downloads them to the current folder.

Step	Detailed instructions to recover the output files
1	reprounzip showfiles docker_folder
2	sudo reprounzip docker download –all docker_folder

Table 16
Tested software platforms for the reproducible experiments based on ReproZip.

Platform	ReproUnzip configuration	Tested
Ubuntu-base	ReproUnzip based on Docker	Yes
Mac Pro (OS X El Capitan – 10.11.6) with 16 Gb RAM	ReproUnzip based on Vagrant	Yes

ers who prefer to use ReproUnzip with VirtualBox instead of Docker can consult the ReproZip installation page.[5]

The running of the reproducible experiments based on Docker for Ubuntu took around 16 days on the aforementioned Ubuntu-base workstation. Once the running has finished, you should follow the instructions shown in Table 15 to recover the output files from the Docker container, as detailed in Table 11. Finally, Table 16 summarizes the software platforms in which the reproducible experiments [64] have been successfully reproduced.

The old low-end Ubuntu-base workstation with only 4Gb RAM is enough to successfully run the experiments detailed in Table 11. However, we suggest a high-end workstation in order to reduce the overall running time.

5. The WNSimRep v1 dataset

WNSimRep v1 is a replication dataset defined by a collection of intrinsic and corpus-based IC models based on WordNet 3.0, which is enriched with the most common taxonomical features used in the computation of similarity measures and intrinsic IC models, as well as the similarity values reported by most similarity measures in order to assist the replication of previously reported methods and experiments. The *WNSimRep v1* dataset is part of the experimental data reported in our three aforementioned works [56–58], and it was automatically generated using HESML as detailed in Section 4.3.1.

Despite *WNSimRep v1* being based on WordNet 3.0, the proposed framework could be adapted and extended to any type of base ontology, or intrinsic similarity measure. Because of the lack of space, *WNSimRep v1* is detailed in a complementary paper, which together with the dataset files, is publicly available at [63]. *WNSimRep v1* includes three different types of data files: (1) node-valued IC data files with taxonomical features, (2) edge-valued IC data files with the conditional probability between child and parent concepts, and (3) synset-pair-valued data files with taxonomical features and IC-based similarity measures for the synset pairs derived from the classic RG65 benchmark introduced by [111]. The dataset includes 22 intrinsic IC models, 8 corpus-based IC models based on the Resnik method, 8 corpus-based IC models based on the well-founded *CondProbCorpus* IC model, and 8 corpus-based IC model based on the *CondProbRefCorpus*, which have been evaluated with 22 similarity measures. All the corpus-based IC models are derived from the family of "**add1.dat*" WordNet-based frequency files included in the Pedersen dataset [95], which is a dataset of corpus-based files created for a series of papers on similarity measures in WordNet, such as [93] and [96]. The dataset includes all the IC models and similarity measures evaluated in the experimental surveys carried-out in the three aforementioned works by Lastra-Díaz and García-Serrano in [56–58].

6. Evaluation of HESML

The goals of the experiments described in this section are as follows: (1) the experimental evaluation of the PosetHERep representation model and HESML, as well as their comparison with the state-of-the-art semantic measures libraries called SML [48] and WNetSS [15]; (2) a study of the impact of the size of the taxonomy on the performance and scalability of the state-of-the-art semantic measures libraries; and finally, (3) the confirmation or refutation of our main hypothesis and research questions; Q1 and Q2 introduced in Section 1.1.

6.1. Experimental setup

Our experiments compare the performance of the HESML V1R2 library version available at [60], with the SML 0.9 library version whose source files are available at GitHub,[6] and the recent WNetSS library.[7] We used the compiled *slib-dist-0.9-all-jar.jar* file available at the SML web site[8] for our experiments. As WNetSS is not distributed with its source files, we were not able to carry-out a side-by-side detailed comparison of WNetSS with HESML and SML, as is done between HESML and SML. Thus, we divided our benchmarks into two blocks: (1) a detailed side-by-side comparison between HESML and SML based on the benchmarks detailed in Table 17 ; and (2) a WordNet-based similarity benchmark based on the SimLex665 dataset in order to evaluate the three aforementioned libraries, which is implemented by the *EvaluateWordNetSimilarityDataset* functions in the complementary dataset [61].

In order to evaluate HESML and SML, we have carried out a series of benchmarks based on the creation and interrogation of

[5] https://reprozip.readthedocs.io/en/1.0.x/install.html .

[6] https://github.com/sharispe/slib.

[7] http://wnetss-api.smr-team.org/.

[8] http://www.semantic-measures-library.org/sml/downloads/releases/sml/0.9/slib-dist-0.9-all-jar.jar.

Table 17
Sequence of benchmarks implemented by the *HSMLtests* and *SMLtests* classes within the *HESML_vs_SML_tests.jar* program. The test functions carry-out the same operations on both software libraries, thus, their results can be compared directly.

Benchmark	Description
overallCreation	This test creates a tree-like taxonomy with a defined number of vertexes in which each vertex has a random number of children nodes (2 to 8),
avgCreation	$\frac{overallCreation}{\#vertexes}$
AncDescLea	This test matches the pre-processing made by the SML, and it consists of the computation of the ancestor and descendant sets of each vertex, and the overall leaf set.
avgAncDesLea	$\frac{AncDescLea}{\#vertexes}$
overallCaching	This test measures the number of vertexes cached during the execution of the AncDescLea test (SML pre-processing).
avgCaching	$\frac{overallCaching}{\#vertexes}$
avgShortestPath	Average computation time of the shortest path (5 samples).
allMinDepth	Overall computation time of minimum depth for all vertexes.
avgMinDepth	$\frac{allMinDepth}{\#vertexes}$
allMaxDepth	Overall computation time of the maximum depth for all vertexes.
avgMaxDepth	$\frac{allMaxDepth}{\#vertexes}$
avgLCA	Average time to retrieve the LCA vertex (10,000 samples).
avgMICA	Average time to retrieve the MICA vertex (10,000 samples).
avgSubLea	Average time to retrieve the set of subsumed leaves (10,000 samples).

a sequence of randomly created tree-like taxonomies, whose size grows from 20,000 to 1 million vertexes. The benchmarks have been designed with the aim of evaluating a selection of the most significant topological algorithms used by most ontology-based semantic similarity measures and IC models reported in the literature. Table 17 details the set of benchmarks defined to evaluate the performance of HESML and SML. Because of its high computational cost, we limit the evaluation of the shortest path algorithm to taxonomies with up to 50,000 vertexes. On the other hand, in order to evaluate and compare the performance of WNetSS with HESML and SML, we compare the running-time of the three libraries in the evaluation of the Jiang-Conrath similarity measure [52] with the Seco et al. IC model [119] in the SimLex665 dataset [50].

6.2. Reproducing our benchmarks

All benchmarks detailed in Table 17 are implemented on a single Java console program called *HESML_VS_SML_test.jar*, which is publicly available at [61]. The HESML_vs_SML program links directly with the *HESML-V1R2.jar, slib-dist-0.9-all-jar.jar* and *WNetSS.jar* files containing the latest publicly available software releases of these libraries. The HESML_vs_SML dataset contains all source files and the NetBeans project used to create the entire program, including the pre-compiled version with their dependencies in the *'dist'* subfolder. The *HESML_VS_SML_test/src* folder contains five files as follows: (1) *HESML_vs_SML_test.java* contains the main function; (2) *HESMLtests.java* contains the functions implementing the aforementioned benchmarks on the HESML V1R2 library; whilst (3) *SMLtests.java* contains the same functions as *HESMLtests.java*, but implementing the benchmarks on the SML 0.9 library; and (4) the *WNetSStests.java* contains the function implementing the WordNet-based similarity benchmark; and finally, (5) the *TestResults.java* file implements a class with the aim of collecting all output results in a structured way. In order to reproduce our benchmarks and see the results reported in Tables 20 and 21, and Fig. 3, you should follow the steps detailed in [61].

6.3. Evaluation metrics

The metrics defined for the comparison of the results are the overall and average running time of the operations, measured in microseconds ($\mu secs$), milliseconds (msecs) or seconds (secs), and the increase in memory derived from the caching process. The measurement of the memory use of a Java program is highly influenciated by the Java Virtual Machine (JVM) memory allocation and garbage collector policies. Thus, it is very difficult to carries out a set of measurements on memory use which is reliable, stable and reproducible. For this reason, the metric used for the caching memory is defined by the exact number of vertexes which are stored in the caching structures. Despite not being able to know the exact caching memory allocated in runtime, we know that it is a multiple of the number of cached vertexes, which is defined by the memory size of each vertex (URIs in SML) and the memory required by the data structures used to stored them, typically Java HashSets in SML. Finally, the statistical significance of the results between HESML and SML in the benchmarks detailed in Table 17, as well as the results of the WordNet-based similarity benchmark reported in Table 19, is evaluated using the p-values resulting from the t-student test for the difference mean between the two series of average running times considered as two paired samples sets.

6.4. Results

Tables 20 and 21 show the results of the benchmarks between HESML and SML, whilst Fig. 3 shows a graphical comparison of their performance and Table 18 shows the p-values resulting from the comparison of both series of benchmarks. SML runs out of memory on the taxonomy with 1 million of vertexes. For this reason, we only show the results up to 900,000 vertexes. On the other hand, HESML starts to run out of memory for the same Java heap (4Gb) on taxonomies with 10 million of vertexes or more, a fact that you could check by incrementing the size of the taxonomy in the HESML_vs_SML main function. Finally, Table 18 shows the p-values of the benchmarks which are computed using a one-sided t-student distribution on two paired sample sets. Our null hypothesis, denoted by H_0, is that the difference mean in the average performance between HESML and SML is 0, whilst the alternative hypothesis, denoted by H_1, is that their average performance is different. For a 5% level of significance, it means that if the p-value is greater than 0.05, we must accept the null hypothesis, otherwise we can reject H_0 with an probability of error of less than the p-value.

Table 19 shows the running-time in milliseconds for five evaluations of the Jiang-Conrath similarity measure in the SimLex665 dataset, together with the average running-time for each library on the Windows-based workstation. We evaluate the WordNet-based similarity benchmark five times to allow a statistical significance analysis and produce a more robust estimation.

7. Discussion

HESML V1R2 significantly outperforms SML 0.9 and sets the new state of the art of the problem. Looking at the Tables 20 and 21,

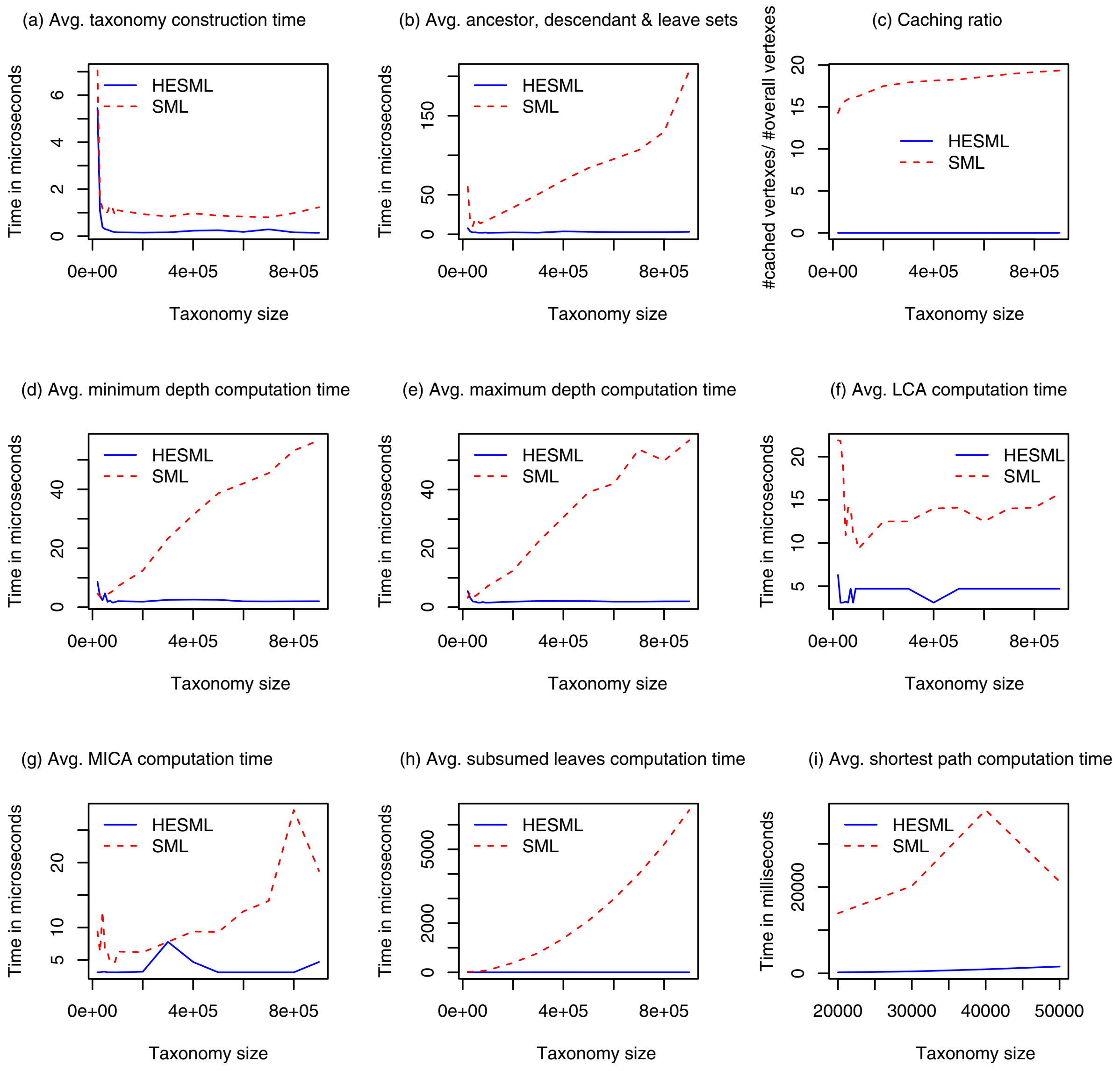

Fig. 3. This figure shows the results obtained by HESML and SML in the series of benchmarks described in the experimental setup, whose values are tabulated in Tables 20 and 21. The computation time is reported in microsecs (μsecs), milliseconds (msecs) or seconds (secs), whilst the increase in memory resulting from the caching carried-out by the SML library is reported in figure(c) as the ratio of the number of cached vertexes as regards the overall number of vertexes, the so called 'taxonomy size'.

Table 18
P-values obtained by using a one-sided t-student distribution for the mean of the differences between two paired samples defined by the HESML and SML benchmark results and a significance level of 95%. The p-values above have been computed by running the *figures_and_table18_Rscript.r* script into the R statistical package, which is provided as complementary material. Any p-value less than 0.05 implies that HESML obtains a statistically significant lower value (running time or caching) than SML. Thus, HESML outperforms SML on this benchmark in a statistically significant manner.

Avg Creation	Avg AncDesLeaves	Avg Caching ratio	Avg Minimum Depth	Avg Maximum Depth	Avg LCA	Avg MICA	Avg Subsumed leaves	Avg shortest path
5.3e-10	4.2e-04	1.6e-18	1.2e-03	8.2e-04	2.3e-09	3.6e-04	6.6e-03	1.0e-02

and Fig. 3, we conclude that HESML outperforms SML in all benchmarks detailed in Table 17. In addition, all p-values in Table 18 are less than 0.05, thus, we conclude that HESML outperforms SML in all benchmarks in a statistically significant manner. Thus, HESML sets the new state of the art in the family of semantic measures libraries in terms of performance and scalability.

Most HESML V1R2 algorithms exhibit linear complexity, thus they are linearly scalable. HESML obtains an almost constant average ratio on most benchmarks, as shown in Tables 20 and 21, and Fig. 3, with the only exception being the shortest path algorithm. The small variation in the average ratios in the aforementioned tables could be attributed to the inherent variability of the time measurement in Java. Thus, most benchmarks exhibit a linear complexity as regards the size of the taxonomy, confirming our theoretical analysis on the scalability of most PosetHERep algorithms introduced in Section 3.2. The set of benchmarks with a constant average ratio, and thus linear complexity, is defined as follows: (1) the creation of the taxonomy (vertex insertion); (2) the retrieval of the ancestor and descendant sets of the vertexes, and the overall leaf set (SML pre-processing); (3) the computation of the minimum and maximum depths of the vertexes; (4) the retrieval of the LCA vertex; (5) the retrieval of the MICA vertex; and (6) the retrieval of the subsumed leaves of the vertexes.

Table 19
Overall running time obtained by the semantic measures libraries in the evaluation of the Jiang-Conrath similarity measure with the Seco et al. IC model in the SimLex665 dataset.

Library	SML	WNetSS	HESML
Run 1 (msecs)	156	177434	110
Run 2 (msecs)	71	177224	89
Run 3 (msecs)	45	177541	97
Run 4 (msecs)	43	173151	85
Run 5 (msecs)	41	179284	82
Avg (msecs)	71.2	176926.8	92.6
t-student p-value (SML, HESML) =			0.147

HESML V1R2 outperforms SML 0.9 including in the benchmarks that use caching. Unlike SML, HESML does not use caching to store any pre-computed set of vertexes. However, HESML significantly outperforms SML in those methods in which SML uses caching, such as the retrieval of the LCA and MICA vertexes, and the set of subsumed leaves of a vertex. On the other hand, HESML makes extensive use of the PosetHeRep model and its algorithms in order to retrieve these objects, outperforming their counterparts based on caching. Thus, our results refute the common belief which states the caching of the entire collection of ancestor and descendant sets is the only solution to speed-up the computation of the aforementioned topological queries. In addition, our results prove that the caching strategy does not only impact the scalability, because of the unneeded and non-linear increment of the memory usage, but also contributes to a low performance as consequence of the continuous interrogations of large hash maps. Specifically, Table 21 shows an almost constant speed-up factor between the average running time for the LCA and MICA benchmarks of HESML as regards SML, which we attribute to the aforementioned interrogations of the caching structures. In the best case, although SML was able to obtain a similar performance to HESML in these tasks after a reengineering of its code, HESML will obtain a better or similar performance without caching. Table 20 shows that SML demands a caching of 19.34 times the taxonomy size for a taxonomy size of 900,000 vertexes, and its caching growing rate is clearly non-linear.

Most SML algorithms exhibit a non-linear time complexity, whilst its best performing methods (LCA and MICA) demand a non-scalable caching strategy. This latter conclusion follows directly from the results shown in Tables 20 and 21, as well as the Fig. 3, and our discussion in the previous paragraph.

HESML outperforms most SML benchmarks by several orders of magnitude. As shown in Tables 20 and 21, the latter statement is especially significant for large sizes of taxonomy in the following benchmarks: (1) computation of the ancestor and descendant sets, (2) computation of the minimum and maximum depths, (3) computation of the subsumed leaves, and (4) computation of the shortest path between vertexes. SML only obtains good results, for the computation of the MICA and LCA vertexes because of the caching, and even in these two latter cases it is significantly outperformed by HESML. Again, the main problem behind most SML algorithms is its low degree of scalability as consequence of its representation model for taxonomies.

The overall outperformance of HESML on SML proves our main hypothesis and answers our two main research questions positively. Thus, our results allow the following conclusions to be drawn: (1) a new intrinsic representation model for taxonomies as the proposed by PosetHERep is able to improve significantly the performance and scalability of the state-of-the-art semantic measures libraries; and (2) it is possible to significantly improve the performance and scalability of the state-of-the-art semantic mea-

Table 20
Results of the benchmarks between the HESML V1R1 (1.1.01) and SML 0.9 semantic measures libraries.

Taxonomy size	Overall Creation time(msecs)	Overall Creation time(msecs)	Avg Creation time(μsecs)	Avg Creation time(μsecs)	AncDesLeaves time(secs)	AncDesLeaves time(secs)	Avg AncDesLeaves (μsecs)	Avg AncDesLeaves (μsecs)	Overall caching# Vertexes	Overall caching# Vertexes	Avg Caching	Avg Caching	Avg Shortest path(msecs)	Avg Shortest path(msecs)
#Vertexes	HESML	SML	HESML	SML	HESML	SML	HESML	SML	HESML	SML	HESML	SML	HESML	SML
20000	109	141	5.45	7.05	0.156	1.203	7.8	60.15	0	285516	0	14.28	228	13894.2
30000	32	47	1.07	1.57	0.109	0.391	3.63	13.03	0	451299	0	15.04	434.4	20218.4
40000	15	47	0.38	1.18	0.094	0.422	2.35	10.55	0	619315	0	15.48	937.6	37819.4
50000	15	47	0.3	0.94	0.125	1.016	2.5	20.32	0	787282	0	15.75	1571.8	21266.2
60000	16	62	0.27	1.03	0.125	1.001	2.08	16.68	0	955252	0	15.92	–	–
70000	16	94	0.23	1.34	0.14	0.969	2	13.84	0	1123293	0	16.05	–	–
80000	15	94	0.19	1.18	0.157	1.219	1.96	15.24	0	1291300	0	16.14	–	–
90000	15	78	0.17	0.87	0.203	1.453	2.26	16.14	0	1459325	0	16.21	–	–
100000	16	110	0.16	1.1	0.187	1.812	1.87	18.12	0	1627322	0	16.27	–	–
200000	31	187	0.15	0.94	0.476	6.766	2.38	33.83	0	3496274	0	17.48	–	–
300000	47	250	0.16	0.83	0.625	15.266	2.08	50.89	0	5376342	0	17.92	–	–
400000	94	390	0.23	0.97	1.469	27.345	3.67	68.36	0	7256275	0	18.14	–	–
500000	125	437	0.25	0.87	1.565	42.001	3.13	84	0	9136247	0	18.27	–	–
600000	110	500	0.18	0.83	1.687	57.143	2.81	95.24	0	11162140	0	18.6	–	–
700000	203	563	0.29	0.8	1.922	74.752	2.75	106.79	0	13242163	0	18.92	–	–
800000	125	781	0.16	0.98	2.25	103.69	2.81	129.61	0	15322176	0	19.15	–	–
900000	125	1110	0.14	1.23	2.829	186.567	3.14	207.3	0	17402182	0	19.34	–	–

Table 21
Results of the benchmarks between the HESML V1R1 (1.1.01) and SML 0.9 semantic measures libraries.

Taxonomy size #Vertexes	AllMinDepths (msecs) HESML	AllMinDepths (msecs) SML	AvgMinDepth (μsecs) HESML	AvgMinDepth (μsecs) SML	AllMaxDepth (msecs) HESML	AllMaxDepth (msecs) SML	AvgMaxDepth (μsecs) HESML	AvgMaxDepth (μsecs) SML	AvgLCA (μsecs) HESML	AvgLCA (μsecs) SML	AvgMICA (μsecs) HESML	AvgMICA (μsecs) SML	AvgSubLea (μsecs) HESML	AvgSubLea (μsecs) SML
20000	172	93	8.6	4.65	109	63	5.45	3.15	6.3	21.9	3.1	9.4	3.2	23.5
30000	109	109	3.63	3.63	94	157	3.13	5.23	3.1	21.8	3.1	6.2	1.6	18.8
40000	94	141	2.35	3.52	78	156	1.95	3.9	3.1	18.8	3.2	12.5	1.5	32.8
50000	235	187	4.7	3.74	93	188	1.86	3.76	3.2	10.9	3.2	6.3	0	29.6
60000	109	265	1.82	4.42	94	266	1.57	4.43	3.1	14.1	3.1	6.2	1.5	56.3
70000	157	344	2.24	4.91	109	359	1.56	5.13	4.7	14.1	3.1	4.6	1.6	50
80000	125	437	1.56	5.46	140	453	1.75	5.66	3.1	11	3.1	4.7	1.6	60.9
90000	157	547	1.74	6.08	140	578	1.56	6.42	4.7	10.9	3.1	4.7	1.6	75
100000	204	703	2.04	7.03	156	719	1.56	7.19	4.7	9.3	3.1	6.3	0	90.6
200000	375	2484	1.88	12.42	375	2485	1.88	12.43	4.7	12.5	3.2	6.2	1.5	386
300000	750	7000	2.5	23.33	625	6656	2.08	22.19	4.7	12.5	7.8	7.8	3.1	785.9
400000	1031	12516	2.58	31.29	828	12250	2.07	30.62	3.1	14	4.7	9.4	3.1	1379.7
500000	1254	19328	2.51	38.66	1031	19547	2.06	39.09	4.7	14.1	3.1	9.3	1.6	2104.8
600000	1188	25203	1.98	42.01	1140	25219	1.9	42.03	4.7	12.5	3.1	12.5	1.6	2976.6
700000	1375	31844	1.96	45.49	1328	37548	1.9	53.64	4.7	14	3.1	14.1	1.6	4001.6
800000	1594	42516	1.99	53.15	1578	39860	1.97	49.83	4.7	14.1	3.1	28.1	1.6	5198.7
900000	1812	50970	2.01	56.63	1781	51095	1.98	56.77	4.7	15.7	4.7	18.7	1.5	6593.9

sures libraries without using any caching strategy by using the PosetHERep model. Likewise, our results confirm our claims in motivation 1.1 in which we state that the caching is a consequence of the use of non-intrinsic naive representation models for taxonomies.

The low performance and scalability of the shortest path algorithm in SML prevents its use in large WordNet-based benchmarks of path-based similarity measures. Looking at Table 20, you can see that SML requires more than 21 s to evaluate the length of the shortest path in a taxonomy with only 50,000 vertexes, it being approximately a half of the WordNet size. This latter fact is especially critical in any WordNet-based word similarity evaluation because the similarity is commonly defined as the maximum similarity in the cartesian product between word senses, thus, it could increase up to two orders of magnitude the latter running time for any path-based similarity measure. On the other hand, looking at Fig. 3.i, you can see the non-linear scaling of the method.

SML obtains the lowest average running-time in the evaluation of a classic IC-based similarity measure in a WordNet-based benchmark, although there is no a statistically significant difference as regard HESML. Looking at Table 19, you can see that SML obtains an average running-time of 71.2 ms, whilst HESML and WNetSS obtain 92.6 and 176,926.8 ms respectively. However, the p-value for the t-student test between SML and HESML is 0.147, thus, there is no a statistically significant difference between these two latter libraries. We attribute this slight advantage of SML on HESML in the WordNet-based test to the WordNet indexing approach of HESML. Despite HESML outperforming SML in the topological algorithms used by the Jiang-Conrath similarity measure, the WordNet indexing and lookup in HESML is up to three times slower than its equivalent in SML. This difference in the performance of the WordNet indexing process between HESML and SML is a consequence of the implementation of two further hashmap lookup operations in HESML, which are not needed by the WordNet indexing approach of SML.

WNetSS obtains the lowest performance in the evaluation of the WordNet-based similarity benchmark, obtaining an average running-time which is more than three orders of magnitude higher than HESML and SML. Table 19 shows that the average running-time of 176,926.8 ms obtained by WNetSS is 2,485 and 1,911 times the average running-time obtained by SML and HESML respectively. This latter fact confirms our statements in Section 1.1.1 on the impact of a software architecture based on a relational database server on the performance and scalability of WNetSS.

Finally, PosetHERep could easily extended in a straightforward way to support any type of semantic relationship, in addition to the 'is-a' taxonomical relationships. Thus, the PosetHERep model could be used as the main building block for large ontologies, and with a proper extension it could be adapted to efficiently manage other non-taxonomical semantic graphs.

7.1. The new state of the art

Our previous discussion allows us to conclude that HESML is the more efficient and scalable semantic measures library between the three libraries evaluated herein. However, there is no a statistically significant difference in the performance of HESML and SML in the evaluation of non path-based similarity measures on WordNet. Thus, SML also provides an efficient and practical solution to evaluate IC-based similarity measures and IC models based on WordNet, despite its performance prevents the evaluation of path-based similarity measures. On the other hand, WNSetSS exhibits a poor performance as consequence of its RDBMS-based caching approach, moreover, it does not provide its source files which seriously prevents its evaluation, extensibility and verification. Finally, there would be interesting to carry out a comparison and verification of the detailed values reported by each library with the aim of checking and validating their implementation.

8. Conclusions and future work

We have introduced a new and linearly scalable representation model for large taxonomies, called *PosetHERep,* and the *HESML V1R2* [60] semantic measures library based on the former. We have proven in a statistically significant manner that HESML V1R2 is the most efficient and scalable publicly available software library of ontology-based similarity measures and intrinsic IC models based on WordNet. However, there is not a statistically significant difference in the performance of HESML and SML in the evaluation of an IC-based similarity measure based on WordNet, unlike the evaluation of any path-based similarity measure in which HESML is much more efficient. On the other hand, *PosetHERep* and *HESML* have proven, conversely to common belief, that is possible to improve significantly the performance and scalability of the state-of-the-art semantic measures libraries without caching using a proper intrinsic representation model for taxonomies. The performance of WNetSS is more than three orders of magnitude lower than HESML and SML because of its caching strategy based on a relational database.

In addition, we have introduced a set of reproducible experiments based on ReproZip [64] and *HESML*, which corresponds to the experimental surveys introduced by Lastra-Díaz and García-Serrano in [57], [56] and [58], as well as the *WNSimRep v1* replication framework and dataset [63] and a benchmark of semantic measures libraries [61].

As forthcoming activities, we plan to extend *HESML* in order to support Wikidata [126] and non "is-a" relationships in the short term, whilst in the mid term, we expect to support the Gene Ontology (GO), MeSH and SNOMED-CT ontologies. In addition, we plan to include further ontology-based similarity measures and IC models reported in the literature, as well as the possibility of importing word embedding files with the aim of allowing the experimental comparison of state-of-the-art ontology-based and corpus-based similarity measures and methods.

9. Revision Comments

This reproducibility paper presents a novel software library (HESML) that implements a plethora of ontology-based semantic similarity measures and information content models. The value of such library is indubitable, since it provides a benchmark to compare existing and potentially new approaches in the field. By using and evaluating the implemented measures and models, researchers are able to thoroughly compare the available implementations and uncover which are the measures that more accurately mimic human understanding. In addition, because the source code is provided, new models and measures can more easily be built on top of the existing ones, facilitating the progress of the research on similarity measures.

While reviewing this manuscript, a few issues around reproducibility were brought into discussion. One issue was related to post-processing: ideally, for reproducibility purposes, the post-processing of output files should be as automatic as possible to facilitate the generation of the final results and figures of the paper. Evaluating performance and scalability is also key to reproducibility, since this makes the library more appealing for readers and researchers who will use it and perform experiments in potentially different computational platforms. Last, not only the instructions to run the library should be clear, but also the implemented modules and functions should be well described to make the library extendable and more useful. The authors satisfactorily took all our comments into account and significantly improved their artifact. It is worth noting that an important outcome of this submission and the reviews was the improvement in performance and scalability of the library, which will greatly benefit every researcher working in this area.

We would like to thank the authors for providing such a valuable artifact to the community, and for their great effort in making sure that all the instructions for building and using the library are clear, and all the experimental results can be reproduced effortlessly.

Acknowledgements

Ted Pedersen kindly answered all our questions and provided his WordNet-based frequency files dataset, whilst Sébastien Harispe provided the SML source code and his total support in evaluating it. Mohamed Hadj Taieb kindly offered us his total support in replicating their similarity measures exactly. Jorge Martínez-Gil, Emmanuel Pothos, Emiel Van Miltenburg and Lubomir Stanchev kindly answered our questions about their methods. Ángel Castellanos proposed us the use of HESML to manage FCA-based applications. Mark Hallett checked the proper use of English. Finally, we are grateful to the reviewers for their suggestions in order to improve significantly this work. This work has been partially supported by the Spanish Musacces (S2015/HUM3494) and VEMODALEN (TIN2015-71785-R) Projects.

Appendix A. Resources in the HESML distribution

Table 22 details the resources and datasets included in the HESML V1R2 distribution.

Table 22
Collection of resources distributed as supplementary material of the present work and included the HESML V1R2 distribution package.

Reference works	Acronym	Resource type	Licensing type
This work and [60]	HESML V1R2	Java software library	CC By-NC-SA 4.0
This work and [63]	WNSimRep v1	Replication dataset	CC By-NC 3.0
Miller [87], Fellbaum [35]	WordNet 2.1	Ontology-based lexicon	Attribution
Miller [87], Fellbaum [35]	WordNet 3.0	Ontology-based lexicon	Attribution
Miller [87], Fellbaum [35]	WordNet 3.1	Ontology-based lexicon	Attribution
Rubenstein and Goodenough [111]	RG65	Word similarity benchmark	Attribution
Miller and Charles [88]	MC28	Word similarity benchmark	Attribution
Agirre et al. [2]	Agirre201	Word similarity benchmark	Attribution
Pirró [103]	$P\&S_{full}$	Word similarity benchmark	Attribution
Hill et al. [50]	SimLex665	Word similarity benchmark	Attribution
Patwardhan and Pedersen [93], Pedersen [96]	WN-IC-3.0.tar	WN-based frequency files	Attribution

References

[1] A. Adhikari, S. Singh, A. Dutta, B. Dutta, A novel information theoretic approach for finding semantic similarity in WordNet, in: Proceedings of IEEE International Technical Conference (TENCON-2015), IEEE, Macau, China, 2015, pp. 1–6, doi:10.1109/TENCON.2015.7372780.

[2] E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, A. Soroa, A study on similarity and relatedness using distributional and WordNet-based approaches, in: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, in: NAACL '09, Association for Computational Linguistics, Stroudsburg, PA, USA, 2009, pp. 19–27.

[3] H. Al-Mubaid, H.A. Nguyen, Measuring semantic similarity between biomedical concepts within multiple ontologies, IEEE Trans. Syst. Man Cybern. Part C Appl. Revi: 39 (4) (2009) 389–398, doi:10.1109/TSMCC.2009.2020689.

[4] M.B. Aouicha, M.A.H. Taieb, Computing semantic similarity between biomedical concepts using new information content approach, J. Biomed. Inf. 59 (2016) 258–275, doi:10.1016/j.jbi.2015.12.007.

[5] M. Ashburner, C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J. Michael Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, M.A. Harris, D.P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J.C. Matese, J.E. Richardson, M. Ringwald, G.M. Rubin, G. Sherlock, Gene Ontology: tool for the unification of Biology, Nat. Genet. 25 (1) (2000) 25–29, doi:10.1038/75556.

[6] M. Baker, 1,500 scientists lift the lid on reproducibility, Nature 533 (7604) (2016) 452–454, doi:10.1038/533452a.

[7] S. Banerjee, T. Pedersen, Extended gloss overlaps as a measure of semantic relatedness, in: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI), 2003, pp. 805–810. Acapulco, México.

[8] R. Banjade, N. Maharjan, N.B. Niraula, V. Rus, D. Gautam, Lemon and tea are not similar: measuring word-to-word similarity by combining different methods, in: A. Gelbukh (Ed.), Proceedings of the 16th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing), LNCS, 9041, Springer, Cairo, Egypt, 2015, pp. 335–346, doi:10.1007/978-3-319-18111-0_25.

[9] M. Batet, A study on semantic similarity and its application to clustering: enabling the classification of textual data, VDM Verlag, 2011.

[10] M. Batet, Ontology-based semantic clustering, AI Commun. Eur. J. Artif. Intell. 4 (3) (2011) 291–292, doi:10.3233/AIC-2011-0501.

[11] M. Batet, A. Erola, D. Sánchez, J. Castellà-Roca, Utility preserving query log anonymization via semantic microaggregation, Inf. Sci. 242 (2013) 49–63, doi:10.1016/j.ins.2013.04.020.

[12] M. Batet, D. Sánchez, A review on semantic similarity, in: M. Khosrow-Pour (Ed.), Encyclopedia of Information Science and Technology, third edition, ISI Global, 2015, pp. 7575–7583, doi:10.4018/978-1-4666-5888-2.ch746.

[13] M. Batet, D. Sánchez, Improving semantic relatedness assessments: ontologies meet textual corpora, Procedia Comput. Sci. 96 (2016) 365–374, doi:10.1016/j.procs.2016.08.149.

[14] M. Batet, D. Sánchez, A. Valls, An ontology-based measure to compute semantic similarity in biomedicine, J. Biomed. Inf. 44 (1) (2011) 118–125, doi:10.1016/j.jbi.2010.09.002.

[15] M. Ben Aouicha, M.A.H. Taieb, A. Ben Hamadou, SISR: system for integrating semantic relatedness and similarity measures, Soft Comput. (2016) 1–25, doi:10.1007/s00500-016-2438-x. http://dx.doi.org/10.1007/s00500-016-2438-x

[16] M. Ben Aouicha, M.A.H. Taieb, A. Ben Hamadou, Taxonomy-based information content and wordnet-wiktionary-wikipedia glosses for semantic relatedness, Appl. Intell. (2016) 1–37, doi:10.1007/s10489-015-0755-x.

[17] M. de Berg, M. Van Kreveld, M. Overmars, O. Schwarzkӧpf, Computational Geometry: Algorithms and Applications, Springer-Verlag, 1997.

[18] E. Blanchard, M. Harzallah, P. Kuntz, A generic framework for comparing semantic similarities on a subsumption hierarchy, in: M. Ghallab, C.D. Spyropoulos, N. Fakotakis, N. Avouris (Eds.), Proceedings of the ECAI, Frontiers in Artificial Intelligence and Applications, 178, IOS Press, 2008, pp. 20–24, doi:10.3233/978-1-58603-891-5-20.

[19] M. Botsch, S. Steinberg, S. Bischoff, L. Kobbelt, OpenMesh - a generic and efficient polygon mesh data structure, CiteseerX (2002).

[20] A. Budanitsky, G. Hirst, Evaluating WordNet-based measures of lexical semantic relatedness, Comput. Ling. 32 (1) (2006) 13–47, doi:10.1162/coli.2006.32.1.13.

[21] A. Castellanos, J. Cigarrán, A. García-Serrano, Formal concept analysis for topic detection: a clustering quality experimental analysis, Inf. Syst. (2017). http://dx.doi.org/10.1016/j.is.2017.01.008.

[22] A. Castellanos, A. García-Serrano, J. Cigarrán, Linked data-based conceptual modelling for recommendation: a FCA-based approach, in: M. Hepp, Y. Hoffner (Eds.), E-Commerce and Web Technologies, Lecture Notes in Business Information Processing, Springer International Publishing, 2014, pp. 71–76, doi:10.1007/978-3-319-10491-1_8.

[23] P. Castells, M. Fernández, D. Vallet, An adaptation of the vector-space model for ontology-based information retrieval, IEEE Trans. Knowl. Data Eng. 19 (2) (2007) 261–272.

[24] J.M. Chaves-González, J. Martínez-Gil, Evolutionary algorithm based on different semantic similarity functions for synonym recognition in the biomedical domain, Knowl.-Based Syst. 37 (2013) 62–69, doi:10.1016/j.knosys.2012.07.005.

[25] M. Chen, R.A. Chowdhury, V. Ramachandran, D.L. Roche, L. Tong, Priority Queues and Dijkstra's Algorithm, Technical Report TR-07-54, Computer Science Department, University of Texas at Austin, 2007. http://www3.cs.stonybrook.edu/~rezaul/papers/TR-07-54.pdf.

[26] F. Chirigati, R. Capone, R. Rampin, J. Freire, D. Shasha, A collaborative approach to computational reproducibility, Inf. Syst. 59 (2016) 95–97, doi:10.1016/j.is.2016.03.002.

[27] F. Chirigati, R. Rampin, D. Shasha, J. Freire, ReproZip: computational reproducibility with ease, in: Proceedings of the 2016 ACM SIGMOD International Conference on Management of Data (SIGMOD), 16, bigdata.poly.edu, 2016, pp. 2085–2088.

[28] F.M. Couto, H.S. Pinto, The next generation of similarity measures that fully explore the semantics in biomedical ontologies, J. Bioinf. Biol. 11 (5) (2013) 1371001, doi:10.1142/S0219720013710017.

[29] F.M. Couto, M.J. Silva, P.M. Coutinho, Measuring semantic similarity between Gene Ontology terms, Data . Knowl. Eng. 61 (1) (2007) 137–152, doi:10.1016/j.datak.2006.05.003.

[30] V. Cross, X. Hu, Using semantic similarity in Ontology alignment, in: Proceedings of the Sixth International Workshop on Ontology Matching (OM), 10th Int. Semantic Web Conference (ISWC 2011), 2011, pp. 61–72. Bonn Germany.

[31] G.G. Dagher, B.C.M. Fung, Subject-based semantic document clustering for digital forensic investigations, Data . Knowl. Eng. 86 (2013) 224–241, doi:10.1016/j.datak.2013.03.005.

[32] R. Dijkman, M. Dumas, B. van Dongen, R. Käärik, J. Mendling, Similarity of business process models: metrics and evaluation, Inf. Syst. 36 (2) (2011) 498–516, doi:10.1016/j.is.2010.09.006.

[33] Editorial, Reality check on reproducibility, Nature 533 (7604) (2016) 437, doi:10.1038/533437a.

[34] J. Fähndrich, S. Weber, S. Ahrndt, Design and use of a semantic similarity measure for interoperability among agents, in: M. Klusch, R. Unland, O. Shehory, A. Pokahr, S. Ahrndt (Eds.), Multiagent System Technologies, Lecture Notes in Computer Science, Springer International Publishing, 2016, pp. 41–57, doi:10.1007/978-3-319-45889-2_4.

[35] , WordNet: An Electronic Lexical Database, in: C. Fellbaum (Ed.), MIT Press, Cambridge, MA, 1998.

[36] S. Fernando, M. Stevenson, A semantic similarity approach to paraphrase detection, in: Proceedings of the 11th Annual Research Colloquium of the UK Special-interest group for Computational Lingusitics, 2008, pp. 45–52. Oxford, UK.

[37] A. Fokkens, M. Van Erp, M. Postma, T. Pedersen, P. Vossen, N. Freire, Offspring from reproduction problems: what replication failure teaches us, in: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL, Sofia, Bulgaria, 2013, pp. 1691–1701.

[38] J.B. Gao, B.W. Zhang, X.H. Chen, A WordNet-based semantic similarity measurement combining edge-counting and information content theory, Eng. Appl. Artif. Intell. 39 (2015) 80–88, doi:10.1016/j.engappai.2014.11.009.

[39] V.N. Garla, C. Brandt, Semantic similarity in the biomedical domain: an evaluation across knowledge sources, BMC Bioinf. 13:261 (2012), doi:10.1186/1471-2105-13-261.

[40] T. Grego, F.M. Couto, Enhancement of chemical entity identification in text using semantic similarity validation, PloS One 8 (5) (2013) e62984, doi:10.1371/journal.pone.0062984.

[41] P.H. Guzzi, M. Mina, C. Guerra, M. Cannataro, Semantic similarity analysis of protein data: assessment with biological features and issues, Briefings Bioinf. 13 (5) (2012) 569–585, doi:10.1093/bib/bbr066.

[42] M.A. Hadj Taieb, M. Ben Aouicha, A. Ben Hamadou, A new semantic relatedness measurement using WordNet features, Knowl. Inf. Syst. 41 (2) (2014) 467–497, doi:10.1007/s10115-013-0672-4.

[43] M.A. Hadj Taieb, M. Ben Aouicha, A. Ben Hamadou, Ontology-based approach for measuring semantic similarity, Eng. Appl. Artif. Intell. 36 (2014) 238–261, doi:10.1016/j.engappai.2014.07.015.

[44] M.A. Hadj Taieb, M. Ben Aouicha, Y. Bourouis, FM3S: features-based measure of sentences semantic similarity, in: E. Onieva, I. Santos, E. Osaba, H. Quintián, E. Corchado (Eds.), Proceedings of the 10th International Conference on Hybrid Artificial Intelligent Systems (HAIS 2015), LNCS, 9121, Springer, Bilbao, Spain, 2015, pp. 515–529, doi:10.1007/978-3-319-19644-2_43.

[45] D. Hao, W. Zuo, T. Peng, F. He, An approach for calculating semantic similarity between words using WordNet, in: Proceedings of the Second International Conference on Digital Manufacturing Automation, IEEE, 2011, pp. 177–180, doi:10.1109/ICDMA.2011.50.

[46] S. Harispe, A. Imoussaten, F. Trousset, J. Montmain, On the consideration of a bring-to-mind model for computing the information content of concepts defined into ontologies, in: Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2015), IEEE, Istanbul, Turkey, 2015, pp. 1–8, doi:10.1109/FUZZ-IEEE.2015.7337964.

[47] S. Harispe, S. Ranwez, S. Janaqi, J. Montmain, The semantic measures library and toolkit: fast computation of semantic similarity and relatedness using biomedical ontologies, Bioinf. 30 (5) (2014) 740–742, doi:10.1093/bioinformatics/btt581.

[48] S. Harispe, S. Ranwez, S. Janaqi, J. Montmain, The semantic measures library: assessing semantic similarity from knowledge representation analysis, in: E. Métais, M. Roche, M. Teisseire (Eds.), Proceedings of the 19th International Conference on Applications of Natural Language to Information Systems (NLDB 2014), LNCS, 8455, Springer, Montpelier, France, 2014, pp. 254–257, doi:10.1007/978-3-319-07983-7_37.

[49] S. Harispe, S. Ranwez, S. Janaqi, J. Montmain, Semantic similarity from natural language and ontology analysis, Synthesis Lectures on Human Language Technologies, 8, Morgan & Claypool publishing, 2015, doi:10.2200/S00639ED1V01Y201504HLT027.
[50] F. Hill, R. Reichart, A. Korhonen, SimLex-999: evaluating semantic models with (Genuine) similarity estimation, Comput. Ling. 41 (4) (2015) 665–695, doi:10.1162/COLI_a_00237.
[51] G. Hirst, D. St-Onge, Lexical chains as representations of context for the detection and correction of malapropisms, in: C. Fellbaum (Ed.), WordNet: An electronic lexical database, Massachusetts Institute of Technology, 1998, pp. 305–332.
[52] J.J. Jiang, D.W. Conrath, Semantic similarity based on corpus statistics and lexical taxonomy, in: Proceedings of International Conference Research on Computational Linguistics (ROCLING X), 1997, pp. 19–33.
[53] Y. Jiang, W. Bai, X. Zhang, J. Hu, Wikipedia-based information content and semantic similarity computation, Inf. Process. Manage. 53 (1) (2017) 248–265, doi:10.1016/j.ipm.2016.09.001.
[54] R. Kyogoku, R. Fujimoto, T. Ozaki, T. Ohkawa, A method for supporting retrieval of articles on protein structure analysis considering users' intention, BMC Bioinf. 12 Suppl 1 (2011) S42, doi:10.1186/1471-2105-12-S1-S42.
[55] J.J. Lastra-Díaz, Intrinsic Semantic Spaces for the Representation of Documents and Semantic Annotated Data, Universidad Nacional de Educación a Distancia (UNED). Department of Computer Languages and Systems, 2014 Master's thesis. http://e-spacio.uned.es/fez/view/bibliuned:master-ETSIInformatica-LSI-Jlastra.
[56] J.J. Lastra-Díaz, A. García-Serrano, A new family of information content models with an experimental survey on WordNet, Knowl.-Based Syst. 89 (2015) 509–526, doi:10.1016/j.knosys.2015.08.019.
[57] J.J. Lastra-Díaz, A. García-Serrano, A novel family of IC-based similarity measures with a detailed experimental survey on WordNet, Eng. Appl. Artif Intell. J. 46 (2015) 140–153, doi:10.1016/j.engappai.2015.09.006.
[58] J.J. Lastra-Díaz, A. García-Serrano, A Refinement of the Well-Founded Information Content Models with a very Detailed Experimental Survey on WordNet, Technical Report, NLP and IR Research Group. ETSI Informática. Universidad Nacional de Educación a Distancia (UNED), 2016. http://e-spacio.uned.es/fez/view/bibliuned:DptoLSI-ETSI-Informes-Jlastra-refinement.
[59] J.J. Lastra-Díaz, A. García-Serrano, in: HESML V1R1 Java software library of ontology-based semantic similarity measures and information content models, 2016. (Mendeley Data, v1). http://dx.doi.org/10.17632/t87s78dg78.1.
[60] J.J. Lastra-Díaz, A. García-Serrano, in: HESML V1R2 Java software library of ontology-based semantic similarity measures and information content models, 2016, doi:10.17632/t87s78dg78.2. (Mendeley Data, v2). http://dx.doi.org/10.17632/t87s78dg78.2.
[61] J.J. Lastra-Díaz, A. García-Serrano, in: HESML_vs_SML: scalability and performance benchmarks between the HESML V1R2 and SML 0.9 sematic measures libraries, 2016, doi:10.17632/5hg3z85wf4.1. (Mendeley Data, v1). http://dx.doi.org/10.17632/5hg3z85wf4.1.
[62] J.J. Lastra Díaz, A. García Serrano, System and method for the indexing and retrieval of semantically annotated data using an ontology-based information retrieval model, United States Patent and Trademark Office (USPTO) Application,US2016/0179945 A1(2016).
[63] J.J. Lastra-Díaz, A. García-Serrano, in: WNSimRep: a framework and replication dataset for ontology-based semantic similarity measures and information content models, 2016. (Mendeley Data v1). http://dx.doi.org/10.17632/mpr2m8pycs.1.
[64] J.J. Lastra-Díaz, A. García-Serrano, in: WordNet-based word similarity reproducible experiments based on HESML V1R1 and ReproZip, 2016. (Mendeley Data, v1). http://dx.doi.org/10.17632/65pxgskhz9.1.
[65] C. Leacock, M. Chodorow, Combining local context and WordNet similarity for word sense identification, in: C. Fellbaum (Ed.), WordNet: An electronic lexical database, MIT Press, 1998, pp. 265–283.
[66] M.C. Lee, A novel sentence similarity measure for semantic-based expert systems, Expert Syst. Appl. 38 (5) (2011) 6392–6399, doi:10.1016/j.eswa.2010.10.043.
[67] H. Leopold, J. Mendling, H.A. Reijers, M. La Rosa, Simplifying process model abstraction: techniques for generating model names, Inf. Syst. 39 (2014) 134–151, doi:10.1016/j.is.2013.06.007.
[68] H. Leopold, S. Smirnov, J. Mendling, On the refactoring of activity labels in business process models, Inf. Syst. 37 (5) (2012) 443–459, doi:10.1016/j.is.2012.01.004.
[69] Y. Li, Z.A. Bandar, D. McLean, An approach for measuring semantic similarity between words using multiple information sources, IEEE Trans. Knowl. Data Eng. 15 (4) (2003) 871–882, doi:10.1109/TKDE.2003.1209005.
[70] D. Lin, An information-theoretic definition of similarity, in: Proceedings of the 15th International Conference on Machine Learning, 98, 1998, pp. 296–304. Madison, WI.
[71] X.Y. Liu, Y.M. Zhou, R.S. Zheng, Measuring semantic similarity in Wordnet, in: Proceedings of the 2007 International Conference on Machine Learning and Cybernetics, 6, IEEE, 2007, pp. 3431–3435, doi:10.1109/ICMLC.2007.4370741.
[72] P.W. Lord, R.D. Stevens, A. Brass, C.A. Goble, Investigating semantic similarity measures across the Gene Ontology: the relationship between sequence and annotation, Bioinf. 19 (10) (2003) 1275–1283, doi:10.1093/bioinformatics/btg153.
[73] F. Mandreoli, R. Martoglia, Knowledge-based sense disambiguation (almost) for all structures, Inf. Syst. 36 (2) (2011) 406–430, doi:10.1016/j.is.2010.08.004.
[74] S. Martínez, D. Sánchez, A. Valls, Ontology-based anonymization of categorical values, in: Modeling Decisions for Artificial Intelligence, in: LNCS, 6408, Springer Berlin Heidelberg, 2010, pp. 243–254, doi:10.1007/978-3-642-16292-3_24.
[75] J. Martinez-Gil, CoTO: a novel approach for fuzzy aggregation of semantic similarity measures, Cognit. Syst. Res. 40 (2016) 8–17, doi:10.1016/j.cogsys.2016.01.001.
[76] G.K. Mazandu, E.R. Chimusa, N.J. Mulder, Gene Ontology semantic similarity tools: survey on features and challenges for biological knowledge discovery, Briefings . Bioinf. (2016). http://dx.doi.org/10.1093/bib/bbw067.
[77] B.T. McInnes, T. Pedersen, Evaluating measures of semantic similarity and relatedness to disambiguate terms in biomedical text, J. Biomed. Inf. 46 (6) (2013) 1116–1124, doi:10.1016/j.jbi.2013.08.008.
[78] B.T. McInnes, T. Pedersen, S.V.S. Pakhomov, UMLS-interface and UMLS-similarity : open source software for measuring paths and semantic similarity, in: Proceedings of the Annual Symposium of the American Medical Informatics Association, 2009, ncbi.nlm.nih.gov, San Francisco, CA, 2009, pp. 431–435.
[79] K. Mehlhorn, P. Sanders, Algorithms and Data Structures: The Basic Toolbox, SpringerLink: Springer e-Books, Springer, 2008.
[80] J. Mendling, H.A. Reijers, J. Recker, Activity labeling in process modeling: empirical insights and recommendations, Inf. Syst. 35 (4) (2010) 467–482, doi:10.1016/j.is.2009.03.009.
[81] L. Meng, J. Gu, A new model for measuring word sense similarity in WordNet, in: Proceedings of the 4th International Conference on Advanced Communication and Networking, ASTL, 14, 2012, pp. 18–23.
[82] L. Meng, J. Gu, Z. Zhou, A new model of information content based on concept's topology for measuring semantic similarity in WordNet, Int. J. Grid Distrib. Comput. 5 (3) (2012) 81–93.
[83] L. Meng, R. Huang, J. Gu, Measuringsemantic similarity of word pairs using path and information content, Int. J. Future Gener. Commun. Netw. 7 (3) (2014) 183–194, doi:10.14257/ijfgcn.2014.7.3.17.
[84] D. Merkel, Docker: lightweight linux containers for consistent development and deployment, Linux J. 2014 (239) (2014). Article No. 2.
[85] R. Meymandpour, J.G. Davis, A semantic similarity measure for linked data: an information content-based approach, Knowl.-Based Syst. 109 (2016) 276–293, doi:10.1016/j.knosys.2016.07.012.
[86] R. Mihalcea, C. Corley, C. Strapparava, Corpus-based and knowledge-based measures of text semantic similarity, in: Proceedings of the AAAI Conference on Artificial Intelligence, 1, AAAI Press, 2006, pp. 775–780.
[87] G.A. Miller, WordNet:a lexical database for English, Commun. ACM 38 (11) (1995) 39–41.
[88] G.A. Miller, W.G. Charles, Contextual correlates of semantic similarity, Lang. Cognit. Processes 6 (1) (1991) 1–28, doi:10.1080/01690969108406936.
[89] S. Montani, G. Leonardi, Retrieval and clustering for supporting business process adjustment and analysis, Inf. Syst. 40 (2014) 128–141, doi:10.1016/j.is.2012.11.006.
[90] M.R. Munafò, B.A. Nosek, D.V.M. Bishop, K.S. Button, C.D. Chambers, N.P. du Sert, U. Simonsohn, E.-J. Wagenmakers, J.J. Ware, J.P.A. Ioannidis, A manifesto for reproducible science, Nat. Hum. Behav. 1 (2017) 0021, doi:10.1038/s41562-016-0021.
[91] J. Oliva, J.I. Serrano, M.D. del Castillo, A. Iglesias, SyMSS: A syntax-based measure for short-text semantic similarity, Data Knowl. Eng. 70 (4) (2011) 390–405, doi:10.1016/j.datak.2011.01.002.
[92] S. Patwardhan, S. Banerjee, T. Pedersen, Using measures of semantic relatedness for word sense disambiguation, in: A. Gelbukh (Ed.), Proceedings of the 4th International Conference on Computational Linguistics and Intelligent Text Processing (CICLING 2003), LNCS, 2588, Springer, Mexico D.F., 2003, pp. 241–257, doi:10.1007/3-540-36456-0_24.
[93] S. Patwardhan, T. Pedersen, Using WordNet-based context vectors to estimate the semantic relatedness of concepts, in: Proceedings of the EACL 2006 Workshop Making Sense of Sense-Bringing Computational Linguistics and Psycholinguistics Together, 1501, 2006, pp. 1–8. Trento, Italy.
[94] T. Pedersen, Empiricism is not a matter of faith, Comput. Ling. 34 (3) (2008) 465–470, doi:10.1162/coli.2008.34.3.465.
[95] T. Pedersen, in: WordNet-InfoContent-3.0.tar dataset repository, 2008. (https://www.researchgate.net/publication/273885902_WordNet-InfoContent-3.0.tar).
[96] T. Pedersen, Information Content Measures of Semantic Similarity Perform Better Without Sense-tagged Text, in: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, in: HLT '10, Association for Computational Linguistics, Stroudsburg, PA, USA, 2010, pp. 329–332.
[97] T. Pedersen, in: Measuring the similarity and relatedness of concepts: a MICAI 2013 Tutorial, 2013, doi:10.13140/RG.2.1.3025.6164.
[98] T. Pedersen, S.V.S. Pakhomov, S. Patwardhan, C.G. Chute, Measures of semantic similarity and relatedness in the biomedical domain, J. Biomed. Inf. 40 (3) (2007) 288–299, doi:10.1016/j.jbi.2006.06.004.
[99] T. Pedersen, S. Patwardhan, J. Michelizzi, WordNet::similarity: measuring the relatedness of concepts, in: Demonstration Papers at HLT-NAACL 2004, in: HLT-NAACL–Demonstrations '04, Association for Computational Linguistics, Stroudsburg, PA, USA, 2004, pp. 38–41.

[100] V. Pekar, S. Staab, Taxonomy learning: factoring the structure of a taxonomy into a semantic classification decision, in: Proceedings of the 19th International Conference on Computational Linguistics, in: COLING '02, 1, Association for Computational Linguistics, Stroudsburg, PA, USA, 2002, pp. 1–7, doi:10.3115/1072228.1072318.

[101] C. Pesquita, D. Faria, A.O. Falcao, P. Lord, F.M. Couto, Semantic similarity in biomedical ontologies, PLoS Comput. Biol. 5 (7) (2009) e1000443, doi:10.1371/journal.pcbi.1000443.

[102] E. Petrakis, G. Varelas, A. Hliaoutakis, P. Raftopoulou, X-similarity: computing semantic similarity between concepts from different ontologies, J. Digital Inf. Manage. 4 (4) (2006) 233–237.

[103] G. Pirró, A semantic similarity metric combining features and intrinsic information content, Data . Knowl. Eng. 68 (11) (2009) 1289–1308, doi:10.1016/j.datak.2009.06.008.

[104] G. Pirró, J. Euzenat, A feature and information theoretic framework for semantic similarity and relatedness, in: P.F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J.Z. Pan, I. Horrocks, B. Glimm (Eds.), Proceedings of the 9th International Semantic Web Conference, ISWC 2010, LNCS, 6496, Springer, Shanghai, China, 2010, pp. 615–630, doi:10.1007/978-3-642-17746-0_39.

[105] G. Pirró, N. Seco, Design, implementation and evaluation of a new semantic similarity metric combining features and intrinsic information content, in: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems: OTM 2008, LNCS, 5332, Springer, 2008, pp. 1271–1288, doi:10.1007/978-3-540-88873-4_25.

[106] E.M. Pothos, J.R. Busemeyer, J.S. Trueblood, A quantum geometric model of similarity, Psychol. Rev. 120 (3) (2013) 679–696, doi:10.1037/a0033142.

[107] R. Rada, H. Mili, E. Bicknell, M. Blettner, Development and application of a metric on semantic nets, IEEE Trans. Syst. Man. Cybern. 19 (1) (1989) 17–30, doi:10.1109/21.24528.

[108] P. Resnik, Using information content to evaluate semantic similarity in a taxonomy, in: Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI 1995), 1, 1995, pp. 448–453. Montreal, Canada.

[109] P. Resnik, Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language, J. Artif. Intell. Res. 11 (1999) 95–130.

[110] M.A. Rodríguez, M.J. Egenhofer, Determining semantic similarity among entity classes from different ontologies, IEEE Trans. Knowl. Data Eng. 15 (2) (2003) 442–456, doi:10.1109/TKDE.2003.1185844.

[111] H. Rubenstein, J.B. Goodenough, Contextual correlates of synonymy, Commun. ACM 8 (10) (1965) 627–633, doi:10.1145/365628.365657.

[112] D. Sánchez, M. Batet, Semantic similarity estimation in the biomedical domain: an ontology-based information-theoretic perspective, J. Biomed. Inf. 44 (5) (2011) 749–759, doi:10.1016/j.jbi.2011.03.013.

[113] D. Sánchez, M. Batet, A new model to compute the information content of concepts from taxonomic knowledge, Int. J. Seman. Web Inf. Syst. (ISWIS) 8 (2) (2012) 34–50, doi:10.4018/jswis.2012040102.

[114] D. Sánchez, M. Batet, D. Isern, Ontology-based information content computation, Knowl.-Based Syst. 24 (2) (2011) 297–303, doi:10.1016/j.knosys.2010.10.001.

[115] D. Sánchez, M. Batet, D. Isern, A. Valls, Ontology-based semantic similarity: a new feature-based approach, Expert Syst. Appl. 39 (9) (2012) 7718–7728, doi:10.1016/j.eswa.2012.01.082.

[116] A. Schlicker, F.S. Domingues, J. Rahnenführer, T. Lengauer, A new measure for functional similarity of gene products based on Gene Ontology, BMC Bioinf. 7 (2006) 302, doi:10.1186/1471-2105-7-302.

[117] A. Schlicker, T. Lengauer, M. Albrecht, Improving disease gene prioritization using the semantic similarity of Gene Ontology terms, Bioinformatics 26 (18) (2010). i561–7 10.1093/bioinformatics/btq384

[118] A. Sebti, A.A. Barfroush, A new word sense similarity measure in WordNet, in: Proceedings of the International Multiconference on Computer Science and Information Technology, IMCSIT 2008, IEEE, 2008, pp. 369–373, doi:10.1109/IMCSIT.2008.4747267.

[119] N. Seco, T. Veale, J. Hayes, An intrinsic information content metric for semantic similarity in WordNet, in: R. López de Mántaras, L. Saitta (Eds.), Proceedings of the 16th European Conference on Artificial Intelligence (ECAI), 16, IOS Press, Valencia, Spain, 2004, pp. 1089–1094.

[120] M.H. Seddiqui, M. Aono, Metric of intrinsic information content for measuring semantic similarity in an ontology, in: Proceedings of the Seventh Asia-Pacific Conference on Conceptual Modelling, in: APCCM '10, 110, Australian Computer Society, Inc., Darlinghurst, Australia,, 2010, pp. 89–96.

[121] H. Shima, in: WS4J home page, 2011. (https://code.google.com/p/ws4j/).

[122] L. Stanchev, Creating a similarity graph from WordNet, in: Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS'14). Article No. 36, ACM, 2014, doi:10.1145/2611040.2611055.

[123] N. Stojanovic, A. Maedche, S. Staab, R. Studer, Y. Sure, SEAL: a framework for developing sEmantic portALs, in: Proceedings of the 1st International Conference on Knowledge Capture, in: K-CAP '01, ACM, New York, NY, USA, 2001, pp. 155–162, doi:10.1145/500737.500762.

[124] A. Tversky, Features of similarity, Psychol. Rev. 84 (4) (1977) 327–352, doi:10.1037/0033-295X.84.4.327.

[125] E. Van Miltenburg, WordNet-based similarity metrics for adjectives, in: Proceedings of the 8th Global WordNet Conference, Global WordNet Association, Bucharest, Romania, 2016, pp. 414–418.

[126] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledge base, Commun. ACM 57 (10) (2014) 78–85, doi:10.1145/2629489.

[127] A. Wolke, M. Bichler, F. Chirigati, V. Steeves, Reproducible experiments on dynamic resource allocation in cloud data centers, Inf. Syst. 59 (2016) 98–101, doi:10.1016/j.is.2015.12.004.

[128] A. Wolke, B. Tsend-Ayush, C. Pfeiffer, M. Bichler, More than bin packing: dynamic resource allocation strategies in cloud data centers, Inf. Syst. 52 (2015) 83–95, doi:10.1016/j.is.2015.03.003.

[129] X. Wu, E. Pang, K. Lin, Z.-M. Pei, Improving the measurement of semantic similarity between gene ontology terms and gene products: insights from an edge- and IC-based hybrid method, PloS One 8 (5) (2013) e66745, doi:10.1371/journal.pone.0066745.

[130] Z. Wu, M. Palmer, Verbs semantics and lexical selection, in: Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics, in: ACL '94, Association for Computational Linguistics, Stroudsburg, PA, USA, 1994, pp. 133–138, doi:10.3115/981732.981751.

[131] Q. Yuan, Z. Yu, K. Wang, A new model of information content for measuring the semantic similarity between concepts, in: Proceedings of the International Conference on Cloud Computing and Big Data (CloudCom-Asia 2013), IEEE Computer Society, 2013, pp. 141–146, doi:10.1109/CLOUDCOM-ASIA.2013.25.

[132] S.-B. Zhang, J.-H. Lai, Exploring information from the topology beneath the Gene Ontology terms to improve semantic similarity measures, Gene 586 (1) (2016) 148–157, doi:10.1016/j.gene.2016.04.024.

[133] Z. Zhou, Y. Wang, J. Gu, A new model of information content for semantic similarity in WordNet, in: Proceedings of the Second International Conference on Future Generation Communication and Networking Symposia (FGCNS'08), 3, IEEE, 2008, pp. 85–89, doi:10.1109/FGCNS.2008.16.

[134] Z. Zhou, Y. Wang, J. Gu, New model of semantic similarity measuring in WordNet, in: Proceedings of the 3rd International Conference on Intelligent System and Knowledge Engineering (ISKE 2008), 1, IEEE, 2008, pp. 256–261, doi:10.1109/ISKE.2008.4730937.