A large reproducible benchmark of ontology-based methods and word embeddings for word similarity

Juan J. Lastra-Díaz [1a], Josu Goikoetxea [2b], Mohamed Ali Hadj Taieb [3c], Ana Garcia-Serrano [4a], Mohamed Ben Aouicha [5c], Eneko Agirre [6b]

[a] *NLP & IR Research Group, ETSI de Informática (UNED)*
Universidad Nacional de Educación a Distancia, Juan del Rosal 16, 28040 Madrid (Spain)
[b] *IXA NLP group, Faculty of Informatics, UPV/EHU*
Manuel Lardizabal 1 (20018), Donostia, Basque Country
[c] *Faculty of Sciences of Sfax, Tunisia*

Abstract

This work is a companion reproducibility paper of the experiments and results introduced by Lastra-Díaz et al. [44], which is based on the evaluation of a companion reproducibility dataset with the HESML V1R4 library and the long-term reproducibility tool called Reprozip. Human similarity and relatedness judgements between concepts underlie most of cognitive capabilities, such as categorisation, memory, decision-making and reasoning. For this reason, the research on methods for the estimation of the degree of similarity and relatedness between words and concepts has received a lot of attention in the fields of artificial intelligence and cognitive sciences. However, despite the huge research effort done, there is a lack of a self-contained, reproducible and extensible collection of benchmarks which being amenable to become a de facto standard for large scale experimentation in this line of research. In order to bridge this reproducibility gap, this work introduces a set of reproducible experiments on word similarity and relatedness by providing a detailed reproducibility protocol together with a set of software tools and a self-contained reproducibility dataset which allow that all experiments and results in our aforementioned work to be reproduced exactly. Our aforementioned primary work introduces the largest, detailed and reproducible experimental survey on word similarity and relatedness reported in the literature, which is based on the implementation of all evaluated methods into a same software platform. Our reproducible experiments evaluate most of methods in the families of ontology-based semantic similarity measures and word embedding models. Finally, we detail how to extend our experiments to evaluate other unconsidered experimental setups.

Keywords:
Ontology-based semantic similarity measures, Word embeddings, Information Content models, Reproducible benchmark, HESML, Reprozip

[1] jlastra@invi.uned.es (corresponding author)
[2] josu.goikoetxea@ehu.eus
[3] mohamedali.hadjtaieb@gmail.com
[4] agarcia@lsi.uned.es
[5] mohamed.benaouicha@fss.usf.tn
[6] e.agirre@ehu.eus

1. Introduction

Human similarity and relatedness judgements between concepts underlie most of cognitive capabilities, such as categorisation, memory, decision-making and reasoning. Thus, the proposal of methods for the estimation of the degree of similarity and relatedness between words and concepts has been a very active line of research in the fields of Artificial Intelligence (AI), Natural Language Processing (NLP) and Information Retrieval (IR). For this reason, authors has been largely involved in this line of research during the last decade. For instance, we have proposed new ontology-based semantic similarity measures [36, 39, 33, 34, 25, 24, 23, 22, 4, 3, 6], Information Content (IC) models [35, 37, 27, 26, 6], word embeddings [18, 20, 19, 21], distributional semantics measures [68, 1, 7], semantic measure libraries [43, 5], reproducibility resources [40, 45], word similarity benchmarks [1], reproducible experiments on word similarity based on WordNet [34, 41] and reproducible benchmarks between semantic measures libraries [34, 38].

Main approaches on word similarity and relatedness proposed in the literature can be categorised in two large families as follows: (1) Ontology-based semantic similarity Measures (OM), and (2) distributional measures whose most recent and successful methods are based on Word Embedding (WE) models. Precisely, our primary work [44] introduces a deep experimental study on both aforementioned families of methods encouraged by the recent advances in the family of WE models, which is based on the implementation and evaluation of all methods in a same software platform based on HESML V1R4 [42] and WordNet 3.0 [50], as well as their subsequent recording with the Reprozip long-term reproducibility tool [13]. Before the publication of this work, the only large reproducible experimental surveys on word similarity reported in the literature were those introduced by Lastra-Díaz and García-Serrano [36, 35, 37] in another reproducibility paper [43] belonging to this same reproducibility initiative [12], in which we also find other works such as those introduced by Wolke et al. [66] and Fariña et al. [14]. However, there is no neither joint reproducible benchmarks on word embeddings and ontology-based semantic similarity measures nor other ones evaluating the latest family of methods on so large count of datasets as those evaluated by our primary work [44].

Reproducibility of methods and research results in the field of NLP has become a serious problem which severely hampers any research effort and the smooth integration of newcomers in the field, which is especially hard for most graduate students who start their scientific career in this aforementioned field. This reproducibility gap was already highlighted in a pioneering work by Pedersen [54], being subsequently confirmed by Fokkens et al. [16] by evaluating several works in the same line of research tackled herein. More recently, Branco et al. [9] introduce a call for reproducibility submissions in a known NLP journal to bridge the aforementioned reproducibility gap. We subscribed to this reproducibility alarm by adopting as basic norm the detailed replication of all methods evaluated in our papers, as well as the warning on many contradictory or unreproducible results in a series of papers in this line of research, such as the works introduced by Lastra-Díaz and García-Serrano [36, 35, 37], Lastra-Díaz et al. [43] and our primary paper [44]. In a recent and valuable reproducibility study in the field of NLP, Wieling et al. [64, p.641] found that only a third part of the published works (36.2%) provided their source code; however, they found by evaluating a random sample of ten works that only a tenth part of the former group could be reproduced exactly. Thus, this later finding yields an alarming ratio of only a 3.62% of reproducible works in this aforementioend study. For all reasons above, we subscribe both the reproducible manifesto [53] for a reproducible science and reproducibility initiative lead by Information Systems [12], as well as the slow science manifesto[7] for a reflective research. Finally, we make our own the words of Pedersen [54, p.470]: "we might one day only accept for publication articles that are accompanied by working software that allows for immediate and reliable reproduction of results".

The aim of this work is to introduce a detailed experimental setup based on a collection of publicly available software tools [42] and reproducibility resources [45, 46], which are provided as supplementary material, with the aim of exactly reproducing all experiments and results reported in our primary work [44].

1.1. Main motivation

Our first motivation is to introduce a self-contained and easily reproducible set of experiments on word similarity and relatedness which allow to reproduce all experiments, results and conclusions introduced by our primary work [44] exactly.

A second motivation is the lack of a self-contained, reproducible and extensible collection of benchmarks on word similarity and relatedness which jointly evaluate the most recent methods on the families of ontology-based semantic similarity measures and word embedding models on a same software platform, and consequently, being amenable to become a de facto standard for large scale experimentation in this line of research. Despite the huge research effort done during the last decades, such as witnessed by the plethora of methods reviewed and evaluated in our primary work [44], there is still a lack of a fully automatic, reproducible and extensible collection of benchmarks which make the evaluation and development of word similarity and relatedness methods easier. In general, there is a lack of reproducibility resources in this line of research which was partially bridged by the introduction of several semantic measures libraries, such as SML [29], SISR [5] and the most recent HESML [43] which is the largest and efficient among them, in addition to provide self-contained and easily reproducible experiments by the first time. Likewise, the reproducible experiments and reproducibility datasets introduced by Lastra-Díaz et al. [43] and Lastra-Díaz and García-Serrano [40, 41, 38] respectively have allowed by the first time to reproduce a set of large experimental surveys on ontology-based semantic similarity measures based on WordNet [36, 35, 37] exactly. However, recent and fast advances in the family of word embedding models together with the active research on ontology-based methods have raised the need to carry-out joint evaluations of both families of methods in a large set of benchmarks to elucidate the state of the problem.

[7] http://slow-science.org/

1.2. Definition of the problem and contributions

This work tackles the problem of designing and implementing for the first time a self-contained and easily reproducible set of experiments on word similarity and relatedness of the families of ontology-based semantic similarity measures based on WordNet and word embeddings by providing a very detailed reproducibility protocol together with a set of software tools [42] and a companion reproducibility dataset [45] which is publicly available at [46]. In addition, we detail how our reproducible experiments could be extended for setting up and evaluating unconsidered experimental setups including other datasets, word embeddings or ontology-based semantic similarity measures.

The rest of the paper is structured as follows. Section 2 introduces HESML library [43] and Reprozip tool [12] which set the software platform originally used to run all experiments introduced herein and our long-term reproducibility platform respectively. Section 3 introduces the new reproducible experiments on word similarity whilst section 4 details how them can be extended or created new ones from scratch. Finally, we introduce our conclusions and future work.

2. Background on HESML and Reprozip

HESML [43] is a self-contained Java software library of semantic measures based on WordNet whose latest version, called HESML V1R4 [42], also supports the evaluation of pre-trained word embedding models, such as those introduced by Mikolov et al. [49], Pennington et al. [57], Schwartz et al. [61], Wieting et al. [65], Goikoetxea et al. [20], Bojanowski et al. [8], Agirre and Soroa [2], Camacho-Collados et al. [11] and Mrkšić et al. [52]. HESML is a self-contained experimentation platform on word similarity and relatedness which is especially well suited to run large experimental surveys by supporting the execution of automatic reproducible experiment files on word similarity based on a XML-based file format (*.exp). Despite the latest version of HESML only supports WordNet, it could be easily extended to manage other ontologies by implementing the proper parsers as detailed by Lastra-Díaz et al. [43]. HESML library has been completely developed in NetBeans 8 and Java 8, being distributed with three WordNet versions whilst *HESMLclient* is a complementary Java console program whose aim is to run word similarity experiments by calling HESML functionality. For a detailed introduction to HESML, we refer the reader to its introductory paper [43]. Table 1 shows a summary of technical and legal information of the latest HESML version used in our experiments.

On the other hand, ReproZip is a virtualization tool introduced by Chirigati et al. [13], whose aim is to warrant the exact replication of experimental results onto different systems 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. Other valuable feature of Reprozip is that it allows to modify the input files of any Reprozip package with the aim of evaluating a set of experiments using originally unconsidered methods, configuration parameters or datasets. Reprozip supports main virtualization platforms as Docker and VirtualBox; however, our preferred option is Docker. For a comparison of these two types of virtualization platforms, we

HESML software library	Description
Current code version.	V1R4
Legal Code License.	Creative Commons By-NC-SA 4.0
Permanent code repository used for this version.	http://dx.doi.org/10.17632/t87s78dg78.4
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	Sample source code in the HESMLclient program.
Community forum for questions.	hesml+subscribe@googlegroups.com, hesml+unsubscribe@googlegroups.com

Table 1: Technical and legal information of the latest version of the HESML software library [43] used in our experiments.

Reprozip tool	Description
Current version	1.0.16
Web site	https://www.reprozip.org
Supported platforms	Linux, Windows and MacOS

Table 2: Technical and access information of Reprozip long-term reproducibility tool [12].

refer the reader to the survey introduced by Merkel [48], in which the author introduces Docker and compares it with classic Virtual Machines (VM) such as VirtualBox. Finally, Reprozip also simplifies the generation, packaging and execution of Docker-based experiments. For all reasons above, we encourage the research community to use Reprozip as a long-term reproducibility backup. Table 2 shows a summary of technical and access information of the Reprozip reproducibility tool.

3. The Reproducible Experiments on word similarity

The aim of this section is to introduce a set of detailed experimental setups in order to replicate the methods and experiments introduced by our primary work [44] exactly. Section 3.1 details the experimental setup for the implementation of our experiments in our primary work [44], then section 3.2 details the minimal system requirements for the testing platforms with the aim of running our reproducible experiments. Likewise, section 3.2 reports the running times obtained by the authors and reviewers in the evaluation of our reproducible experiments in different testing platforms. Section 3.3 details the procedure for obtaining and compiling HESML source code, as well as running its pre-compiled jar files. We note that it is unneeded to compile the HESML source code to run the experiments because the HESML distribution already includes pre-compiled versions of the HESMLclient program with the latest HESML version. Next, section 3.4 introduces the method to run our experiments which are based on the running of HESMLclient program whilst section 3.5 introduces our long-term reproducibility method based on Reprozip. Finally, section 3.6 introduces the automated data analysis carried-out to process the raw similarity values generated by our experiments and computing all evaluation metrics reported in our aforementioned primary work [44], as well as a report in HTML file format showing all data tables generated from our raw data.

3.1. Experimental setup in our primary paper

All experiments carried-out in our primary paper [44] were implemented in *HESML V1R4* [42] by running *HESMLclient* program with a reproducible experiment file in XML-based (*.exp) file format which encodes the evaluation of all semantic measures in all datasets as listed in table 4. Experimental setup and software platform used to implement all our experiments is detailed in [44, figure 3]. HESML V1R4 implements all ontology-based semantic similarity measures based on WordNet 3.0 as well as the evaluation of all pre-trained word embedding models evaluated in our benchmarks. In addition, execution of our experiments was recorded into a long-term reproducibility Reprozip file, called "*WN_ontology_measures_vs_embeddings.rpz*", which is part of our companion reproducibility dataset [45], being publicly available at [46]. Our aforementioned Reprozip file can be reproduced in any Reprozip compliant platform[8] as detailed in section 3.5. Thus, all our methods, experiments and results can be reproduced using two different software platforms and methods as detailed in table 3. First reproducibility method is based on the execution of *HESMLclient* program whilst second one is based on the execution of the aforementioned Reprozip file.

Sotware used	Supported reproducibility methods
HESMLclient [42]	You should download HESML V1R4 [42] and a supplementary ZIP file containing the collection of pre-trained word embedding files (*WordEmbeddings.zip* [46]), and then running *HESMLclient* with the reproducible file as input, as detailed in section 3.4.
ReproUnzip [13]	You should download our supplementary Reprozip file [46] and setting up and running Reprounzip as detailed in section 3.5.

Table 3: Our two methods to reproduce all experiments and results introduced by our primary work [44]. HESMLclient method is that originally used to run our experiments in our primary work, whilst ReproUnzip provides a long-term reproducibility method regardless the original testing platform used to run our experiments.

The two reproducibility methods cited in table 3 were introduced in the introductory HESML paper [43] which introduces a detailed protocol to reproduce all experiments, results, data tables and figures reported in three papers previously introduced by Lastra-Díaz and García-Serrano [35, 36, 37], as well as the benchmarks between semantic measures libraries reported in [43]. All experiments detailed herein were originally implemented on an UBUNTU 16.04

[8] https://www.reprozip.org/

virtual computer with 8 Gb of RAM and 100 Gb of disk space called UBUNTU-base1 as detailed in table 5. However, it could be reproduced in any Java 8 or Reprounzip compliant platform by using any of the two aforementioned methods above, which includes most Linux-based, MacOS-based and Windows-based platforms. For this reason, our experiments have been successfully reproduced using both HESMLclient and ReproUnzip methods (see figure 1) in all testing platforms detailed in table 5 with the running times reported in table 6.

Figure 1 shows the three reproducibility workflows introduced herein which are defined by the selection of one of the two reproducibility methods shown in table 3 with a specific testing platform. HESML distribution includes the pre-compiled version of HESML V1R4 and HESMLclient .jar files, thus any reader interested in reproducing our experiments can directly follow the setup instructions in tables 7 and 8, and subsequently running the experiments as detailed in tables 10 and 11. On the other hand, table 4 shows the full collection of reproducible experiments encoded by the *"benchmark_survey.exp"* file (see figure 2), as well as the corresponding raw output files that are generated during its execution whose subsequent processing allows to reproduce the results reported in our primary paper [44] exactly, as detailed in section 3.6.

Word similarity and relatedness benchmarks reproduced herein [44]

Dataset	Raw output files generated by our experiments
MC28 [51]	raw_similarity_values_MC28_dataset.csv
RG65 [60]	raw_similarity_values_RG65_dataset.csv
PS_{full} [58]	raw_similarity_values_PSfull_dataset.csv
Agirre201 [1]	raw_similarity_values_Agirre201_lowercase_dataset.csv
SimLex665 [30]	raw_similarity_values_SimLex665_dataset.csv
MTurk771 [28]	raw_similarity_values_MTurk771_dataset.csv
MTurk287/235 [59]	raw_similarity_values_MTurk287-235_dataset.csv
WS353Rel [15]	raw_similarity_values_WS353Rel_dataset.csv
Rel122 [63]	raw_similarity_values_Rel122_dataset.csv
WS353Full [15]	raw_similarity_values_WS353Full_dataset.csv
SimLex111 [30]	raw_similarity_values_SimLex111_dataset.csv
SimLex222 [30]	raw_similarity_values_SimLex222_dataset.csv
SimLex999 [30]	raw_similarity_values_SimLex999_dataset.csv
SimVerb3500 [17]	raw_similarity_values_SimVerb3500_dataset.csv
MEN [10]	raw_similarity_values_MEN_dataset.csv
YP130 [67]	raw_similarity_values_YP130_dataset.csv
RW2034 [47]	raw_similarity_values_RareWords2034_dataset.csv
RW1401 [47]	raw_similarity_values_RareWords1401_dataset.csv
SCWS [32]	raw_similarity_values_SCWS1994_dataset.csv

Table 4: Collection of raw output files generated by the execution of the "benchmark_survey.exp" reproducible experiment file by any of the two aforementioned reproducibility methods. Each raw output file contains the raw similarity or relatedness values returned for each word pair by each semantic measure. These raw output files are subsequently processed by a R-language script to produce the final data tables shown in our primary paper [44], as detailed in section 3.6. For further details on the datasets above, we refer the reader to our primary paper [44, table 3].

HESML V1R4 distribution [42] contains all source files and pre-compiled versions of the *HESML-V1R4.jar* library and *HESMLclient.jar* Java console

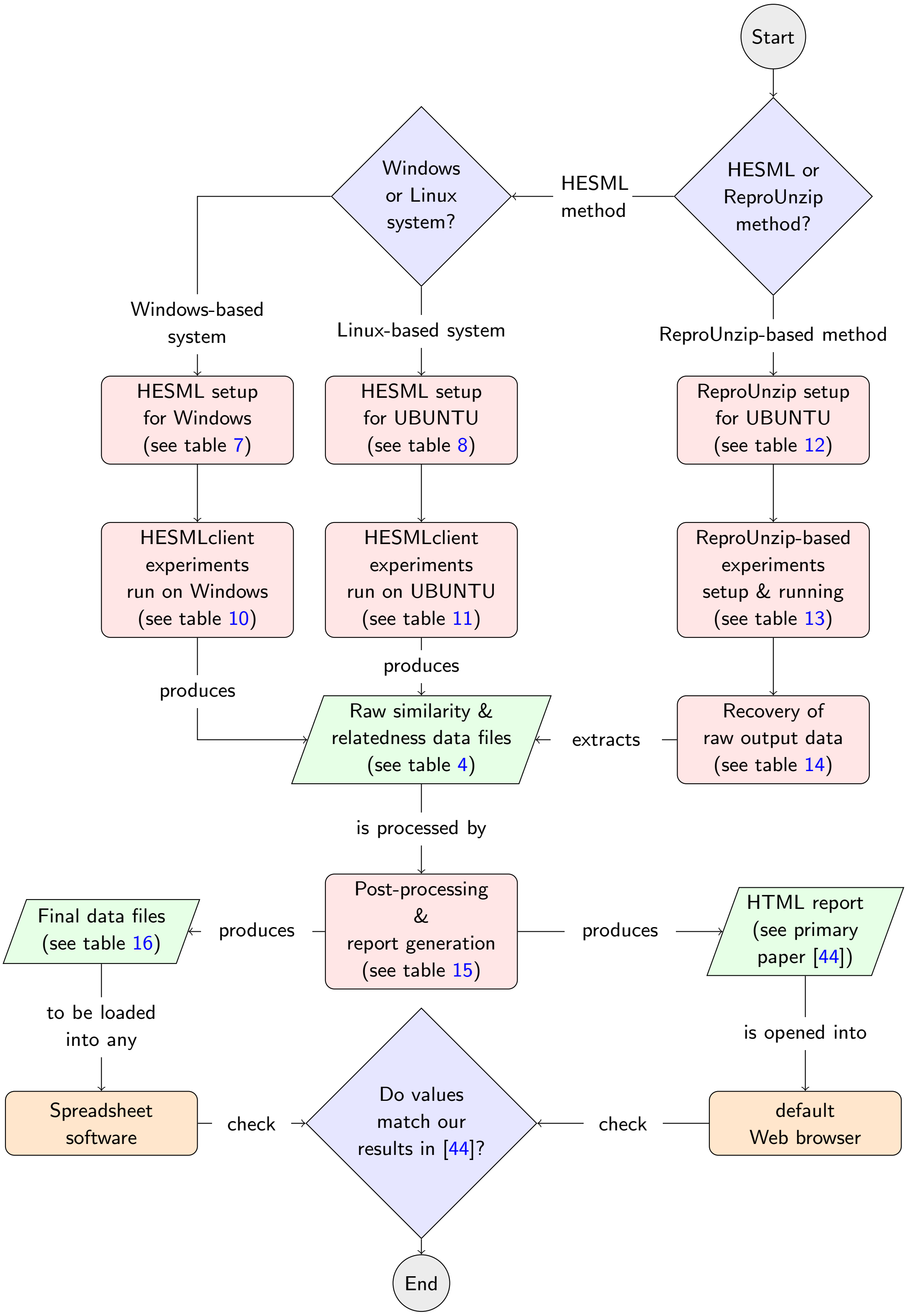

Figure 1: Reproducibility workflows using either HESMLclient or ReproUnzip programs to run the reproducible experiments introduced herein. The three workflows detailed above produce the same raw and processed data files, as well as a collection of HTML pages which reproduce all data tables reported by our primary paper [44].

program. Thus, it is enough to download its official distribution from Mendeley [42] or GitHub[9] in order to run our experiments. However, for the sake of completeness, section 3.3 introduces the detailed steps to obtain and compile HESML V1R4. Finally, we introduce a companion reproducibility dataset [45], being publicly available at [46]. This aforementioned reproducibility dataset gathers into a common repository all data files required to reproduce our experiments with the aim of providing a consolidated and permanent version of these files, and thus avoiding the tedious work of gathering all these stuff as well as any risk of alteration or unavailability of them in the future.

Testing platform	Type	Operating Sys.	Configuration	Tested by
Ubuntu-base1	Virtual	Ubuntu 16.04	1 Core Intel E5-2640-v2 CPU @2 GHz, 8 Gb RAM, 100 Gb SSD disk	Authors
Ubuntu-base2	Virtual	Ubuntu 16.04	2 Intel Core Xeon E5 2699-v4 CPU @2.2 GHz, 8 Gb RAM, 100 Gb SSD disk	Authors
Ubuntu-base3	Virtual	Ubuntu 16.04	1 Core Intel E5-2640-v2 CPU @2 GHz, 8 Gb RAM, 200 Gb SSD disk	Authors
Ubuntu-base4	Virtual	Ubuntu 16.04	2 Intel Core Xeon E5 2699-v4 CPU @2.2 GHz, 8 Gb RAM, 200 Gb SSD disk	Authors
Windows-base1	Laptop	Windows 10x64	1 Intel Core i7-5500U CPU @2.4GHz, 16 Gb RAM, 100 Gb SSD disk	Authors
XXXX-rev1	—	—	—	*Reviewer 1
XXXX-rev2	—	—	—	*Reviewer 2
XXXX-rev3	—	—	—	*Reviewer 3

Table 5: Testing platforms successfully used to reproduce our experiments. Virtual computers are cloud-based servers rented to `clouding.io` which are based on the OpenStack virtualization platform [62]. Ubuntu-base1 and Ubuntu-base2, as well as as Ubuntu-base3 and Ubuntu-base4, differ only in the disk space demanded by Reprounzip. On the other hand, Ubuntu-base1 and Ubuntu-base2 differ from Ubuntu-base3 and Ubuntu-base4 in that these two later platforms use a more modern CPU that the former ones, which were used in the implementation of our original experiments in [44]. For this reason, the experiments reproduced on Ubuntu-base3 and Ubuntu-base4 configurations report lower running times than the former ones as shown in table 6. (*) We will include the testing platforms evaluated by the reviewers.

3.2. System requirements and performance evaluation

Table 5 shows the testing platforms in which we have successfully reproduced the experiments detailed herein, whilst table 6 shows their running times in the completion of all experiments for each aforementioned reproducibility method and testing platform. The configuration of these platforms sets the minimal system requirements to reproduce our experiments. Unlike the execution of our experiments using *HESMLclient* program on the UBUNTU-based computers

[9]`https://github.com/jjlastra/HESML.git`

detailed in table 5, the execution using Reprounzip demands more disk space because it needs to setup a docker container to run the experiments. For this reason, UBUNTU-Reprounzip platforms shown in table 5 are based on a minimal overall disk space of 200 Gb with the aim of setting up UBUNTU, Docker and the resources required by our Reprozip package.

Run	Testing platform	Method	Running time	Tested by
1	Ubuntu-base1	HESMLclient	17581 min ≈ 12.2 days	Authors
2	Ubuntu-base3	ReproUnzip	18109 min ≈ 12.6 days	Authors
3	Ubuntu-base2	HESMLclient	9622 min ≈ 6.68 days	Authors
4	Ubuntu-base4	ReproUnzip	11732 min ≈ 8.15 days	Authors
5	Windows-base1	HESMLclient	10 days	Authors
6	Ubuntu-base2	HESMLclient	10201 min ≈ 7.08 days	Authors
7				*Reviewer 1
8				*Reviewer 2
9				*Reviewer 3

Table 6: Running times obtained on different testing platforms for the execution of all benchmarks by using HESMLclient program with the 'benchmark_survey.exp' experiment file, or by running ReproUnzip program with the "WN_ontology_measures_vs_embeddings.rpz" file. (*) We will include the performance reported by the reviewers.

3.3. Obtaining and compiling HESML

Table 1 shows the technical information required to obtain and compile the *HESML* source code and run the experiments detailed in table 4. HESML V1R4 distribution includes compiled versions of *HESML* library and the *HESMLclient* program, thus this later program could be directly used without the need of compiling the source code in NetBeans. There are two different ways of obtaining the *HESML* source code as follows: (1) by downloading the latest HESML version from the permanent Mendeley Data link [42]; or (2) by downloading it from its GitHub repository detailed in table 1. Once the HESML source code has been downloaded and extracted onto your hard drive, the project will have the folder structure shown in figure 2 and detailed below:

HESML 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-V1R4.jar* distribution file generated during the compilation, whilst *HESMLclient* 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 any (*.exp) reproducible experiment file.

PedersenICmodels folder contains the full WordNet-InfoContent-3.0 collection of WordNet-based frequency files created by Ted Pedersen [55]. 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 [56].

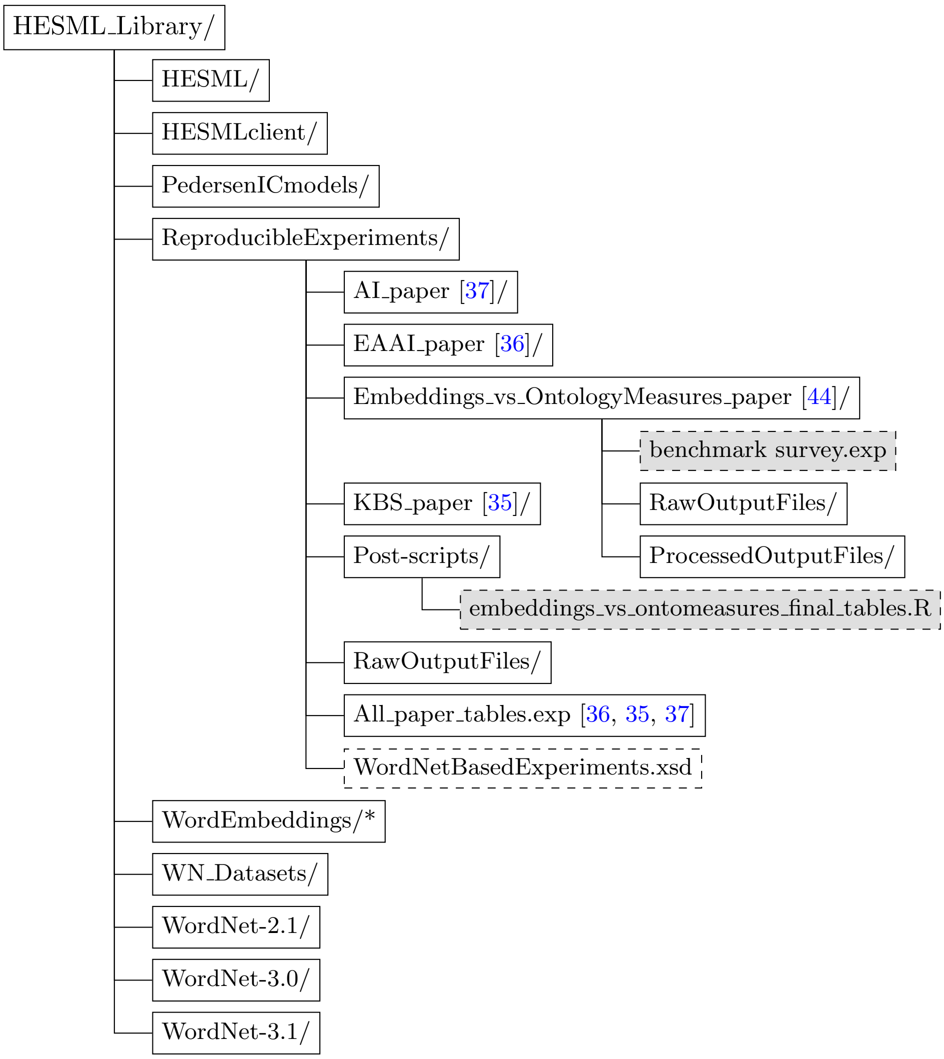

Figure 2: Directory structure of the HESML library once it has been extracted onto disk. The reproducible experiment file and the post-processing R-language script used to reproduce and generate our final data tables respectively are shown in dashed-line boxes in grey, whilst XML-based experiment file format is detailed by XML-schema file shown in unfilled dashed-line box. (*) WordEmbeddings folder contains the pre-trained files for all word embedding models used in our experiments; however, this folder is neither included by the HESML V1R4 [42] distribution nor HESML V1R4 release at GitHub repository because its large size. Thus, you must download the "*WordEmbeddings.zip*" file [46] and extract it onto the main *HESML_Library* directory to retrieve this folder and its content.

ReproducibleExperiments folder contains one subfolder for each paper introduced by Lastra-Díaz and García-Serrano [35, 36, 37] and our primary paper [44] reproduced herein. Likewise, the aforementioned folder also contains a XML-schema file called "*WordNetBasedExperiments.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 corresponding to the three aforementioned papers of Lastra-Díaz and García-Serrano. All (*.exp) files have been created with the XML Spy editor. In addition, this folder contains the *RawOutputFiles* subfolder with all the raw output files of the three aforementioned papers [35, 36, 37].

Post-scripts folder contains a set of post-processing R scripts which process the raw output files generated by all reproducible experiments to generate all final data tables and figures reported in our papers exactly.

WN_datasets folder contains a collection of (*.csv) data files with fields separated by semicolon which correspond to the word similarity benchmarks shown in table 4, whilst *WordNet-2.1*, *WordNet-3.0* and *WordNet-3.1* contain the database files of three different versions of WordNet.

Embeddings_vs_OntologyMeasures_paper folder contains the reproducible experiment file "benchmark_survey.exp" encoding all benchmarks introduced herein and detailed in table 4. In addition, this folder contains a subfolder called "RawOutputFiles" containing all raw output similarity files generated by our experiments. The R-language script file called "embeddings_vs_ontomeasures_final_tables.R" generates all files in "ProcessedOutputFiles" subfolder.

Tables 7 and 8 show a detailed step-by-step procedure to set up our reproducible experiments based on HESML on any Windows or Linux-based system respectively. HESML distribution includes pre-compiled versions of HESML-client program and HESML library; thus, you could skip the compilation step for running our experiments. However, for the sake of completeness, we briefly detail the compilation steps in table 9.

Step	Windows-based setup instructions for HESMLclient experiments
(1)	Install Java 8 runtime or higher in your workstation.
(2)	Open a PowerShell console (Windows 7 and higher) in any directory.

(3)	Create a working directory and move to it as follows:
	$ mkdir REPRODIR
	$ cd REPRODIR

(4)	Download and extract latest HESML version from its GitHub repository (see URL below) using either any Web browser or PowerShell as detailed below: $ powershell -command "& { iwr `https://github.com/jjlastra/HESML/archive/master.zip` }" $ Expand-Archive ./master.zip .

(5)	Download the *WordEmbeddings.zip* file from our Dataverse repository [46, see URL below] and extract it onto HESML root directory using either any Web browser or PowerShell as detailed below:
	$ cd HESMLmaster/HESML_LIBRARY
	$ mkdir WordEmbeddings
	$ cd WordEmbeddings
	$ powershell -command "& { iwr `https://doi.org/10.21950/wordembeddings.zip` }"
	$ Expand-Archive ./wordembeddings.zip .

Table 7: Detailed instructions for downloading HESML V1R4 onto a Windows-based system from its GitHub repository.

Step	Linux-based setup instructions for HESMLclient experiments

(1)	Install Java 8 and Java SE Dev Kit 8 or higher as follows:
	$ sudo apt-get update
	$ sudo apt-get -y install default-jdk

(2)	Install UNZIP program as follows:
	$ sudo apt-get update
	$ sudo apt-get -y install unzip

(3)	Create a working directory and move to it as follows:
	$ mkdir REPRODIR
	$ cd REPRODIR

(4)	Download the latest HESML version from GitHub (see URL below) as follows: $ wget `https://github.com/jjlastra/HESML/archive/master.zip`
(5)	Extract *master.zip* file onto your working directory as follows: $ unzip master.zip

(6)	Download the *WordEmbeddings.zip* file from our Dataverse repository [46, see URL below] and extract it onto HESML root directory as detailed below.
	$ cd HESMLmaster/HESML_LIBRARY
	$ wget `https://doi.org/10.21950/wordembeddings.zip`
	$ unzip wordembeddings.zip -d WordEmbeddings

Table 8: Detailed instructions for downloading HESML V1R4 from its GitHub repository onto a Linux-based system.

Step	Detailed instructions to compile HESML
(1)	Follow the step-by-step procedures to download HESML source code as detailed in tables 7 and 8 for Windows or Linux-based systems respectively.
(2)	Install Java SE Dev Kit 8 and NetBeans 8.0.2 or higher in your workstation.
(3)	Launch NetBeans IDE and open the *HESML* and *HESMLclient* projects contained in the HESML root folder as shown in table 2. NetBeans automatically detects the presence of a *nbproject* subfolder with the project files.
(4)	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.

Table 9: Detailed instructions for compiling HESML onto any Windows or Linux-based system. We recall that the compilation of HESML is unneeded to run all experiments introduced herein.

3.4. Running the experiments with HESMLclient

Once you have downloaded and extracted the *HESML V1R4* library onto your hard drive as detailed in section 3.3, you are ready to run the reproducible experiments by following the steps detailed in tables 10 and 11 for testing platforms based on Windows and Linux respectively. However, before to run the experiments, you must download the *WordEmbeddings.zip* file [46] and extract it onto the main *HESML_Library* directory as detailed in step 5 of table 7 for Windows, and step 6 of table 8 for the Linux-based case. This later ZIP file contains all pre-trained word embedding files; however, it is not included in current HESML distribution because of its large size and the space limitations of GitHub and Mendeley repositories. We note that the original *HESMLclient* source code is defined to fetch the required input files from the folder structure of *HESML* as shown in figure 2.

Step	HESMLclient running instructions on any Windows-based system
(1)	Open a command console in the *HESMLclient* directory as shown in figure 2. $ cd REPRODIR/HESML_Library/HESMLclient
(2)	Run the following command with the reproducible experiment file: $ java -jar -Xms4096m dist/HESMLclient.jar *../ReproducibleExperiments/Embeddings_vs_OntologyMeasures_paper/benchmark_survey.exp*
(3)	Command in step 2 above will generate all raw output files listed in table 4 onto *../ReproducibleExperiments/Embeddings_vs_OntologyMeasures_paper* folder (see figure 2).

Table 10: Detailed instructions for running our experiments with the *HESMLClient* program on any testing platform based on Windows.

Step	HESMLclient running instructions on any Linux-based system
(1)	Open a Linux command console in the *HESMLclient* directory (see figure 2). user@server$ cd REPRODIR/HESML_Library/HESMLclient
(2)	We create a "screen" session and run HESMLclient in background. Note that HESMLclient execution could take up to two weeks (see table 6). user@server$ screen -S REPROEXPS user@screen$ java -jar -Xms4096m dist/HESMLclient.jar *../ReproducibleExperiments/Embeddings_vs_OntologyMeasures_paper/benchmark_survey.exp*
(3)	We detach from "screen" before to close the server main console user@screen$ CTRL+a, d
(4)	We reattach to the screen console to check the completion of HESMLclient user@server$ screen -r REPROEXPS
(5)	We destroy the "screen" console once finished HESMLclient execution user@server$ screen -X -S REPROEXPS quit
(6)	Second command in step (2) above will generate all raw output files listed in table 4 onto *../ReproducibleExperiments/Embeddings_vs_OntologyMeasures_paper* folder (see figure 2).

Table 11: Detailed instructions for running our experiments with the *HESMLClient* program on any testing platform based on Linux.

3.5. Running the ReproZip experiments

The ReproZip[10] program was used for recording and packaging the running of the *HESMLclient* program with all the reproducible experiments defined by the "*benchmark_survey.exp*" file into the "*WN_ontology_measures_vs_embeddings.rpz*" file, which is publicly available at our UNED Dataverse repository [46]. This later Reprozip file was generated by running Reprozip on the Ubuntu-base1 workstation detailed in table 5; however, in order to run ReproUnzip based on Docker as detailed below is needed to set up an Ubuntu-Reprounzip platform (see table 5). Because the execution of the experiments takes long time, and Reprounzip with Docker cannot be executed in background mode without any output console, we will setup and use "screen" program on Linux.

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. However, because of its simple setup and computational efficiency, 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 12, despite several steps possibly being unnecessary depending on your starting configuration. Once ReproUnzip and Docker have been successfully installed, table 13 shows the detailed instructions to set up and run the reproducible experiments. Those readers who prefer to use ReproUnzip with VirtualBox instead of Docker can consult the ReproZip installation page[11].

The running of the reproducible experiments based on Docker for Ubuntu tooks approximately one week in a modern virtual computer as detailed in table 6. Once the running is completed, you should follow the instructions

[10] https://www.reprozip.org/

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

Step	Detailed setup instructions for ReproUnzip on any Linux-based system
	Steps (1-4) below install Reprounzip and all its dependencies.
(1)	$ sudo apt-get update
(2)	$ sudo apt-get -y install libffi-dev libssl-dev openssl openssh-server
(3)	$ sudo apt-get -y install libsqlite3-dev python-dev python-pip screen
(4)	$ sudo pip install reprounzip[all]
	Steps (5-11) below install latest version of Docker CE whilst step 11 checks its installation. For further details, we refer the reader to the official Docker setup page: https://docs.docker.com/install/linux/docker-ce/ubuntu/
(5)	$ sudo apt-get -y install apt-transport-https ca-certificates
(6)	$ sudo apt-get -y install curl gnupg-agent software-properties-common
(7)	$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
(8)	$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
(9)	$ sudo apt-get update
(10)	$ sudo apt-get -y install docker-ce docker-ce-cli containerd.io
(11)	$ sudo docker run hello-world

Table 12: Detailed instructions on installing ReproUnzip with Docker for Ubuntu. Despite that steps above could look tedious, we prefer that readers are aware of all packages being installed instead of running a single setup script hiding this information.

shown in table 14 to retrieve the raw output files from the Docker container, as listed in table 4. Finally, table 6 reports a sample of software platforms in which the Reprozip-based experiments introduced herein have been successfully reproduced.

3.6. Processing of the raw output files

The running of the "*benchmark_survey.exp*" experiment file generates the collection of comma-separated files (*.csv) listed in table 4, whose values are separated by a semicolon. All raw output files are saved in the same folder as their corresponding input reproducible experiment file.

Raw output similarity files generated by our experiments must be processed in order to compute the Pearson, Spearman and Harmonic score metrics matching the tables shown in our primary paper [44]. We provide a R-language script called "*embeddings_vs_ontomeasures_final_tables.R*" with the aim of automating this post-processing. Latest version of the aforementioned post-processing script should be obtained from HESML GitHub distribution, as detailed in tables 7 and 8, or from our companion reproducibility dataset [46]. This aforementioned script includes the source code of the *mat.sort* function provided by the BioPhysConnectoR package [31], which is no longer available in CRAN server.

In order to carry-out the aforementioned post-processing, you should setup the well-known R statistical program[12] in your workstation and follow the steps

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

Step	Detailed setup and running instructions for our Reprozip-based experiments
(1)	Setup the Reprounzip program onto any supported platform (Linux, Windows and MacOS) by following the step-by-step guide detailed in table 12 (see ReproZip installation page for further information).

(2)	Move to the home directory and create a working directory as follows
	$ cd /home
	$ mkdir REPROEXPS
	$ cd REPROEXPS

(3)	Download the *WN_ontology_measures_vs_embeddings.rpz* (12.4 Gb) from its repository [46]. For instance, you can execute the command below. The download of this file could takes several minutes. $ wget `https://doi.org/10.21950/wn_ontology_measures_vs_embeddings.rpz`
(4)	Next, we must setup the docker container as detailed below which could take up to 45 minutes depending of your testing platform. Thus, we recommend to create a "screen" session to run in background both setup and running of the Reprozip-based experiment. You can detach from "screen" console by pressing "Ctrl+a,d". user@server$ screen -S REPROUNZIP user@server$ reprounzip docker setup wn_ontology_measures_vs_embeddings.rpz docker_folder
(5)	Next, we will run the Reprozip-based experiment. Note that Reprounzip execution could take up to two weeks depending on your hardware setup (see table 6). We strongly recommend to keep open the *screen* console to run the experiment in background as detailed below. user@screen$ reprounzip docker run docker_folder - We detach from "screen" before to close the server main console user@screen$ CTRL+a, d - We reattach to the screen console to check the completion of Reprounzip user@server$ screen -r REPROUNZIP - We destroy the "screen" console once finished Reprounzip execution user@server$ screen -X -S REPROUNZIP quit

Table 13: Detailed instructions on how to reproduce the packaged experiments once Reprounzip has been installed. We use *screen* program with the aim of allowing the execution of Reprounzip in background whilst main program console is detached and closed.

detailed in table 15. Then, you need to install the "knitr" and "readr" packages using the functionality provided for this task by R program. Table 16 shows the output files which are generated from the raw output files listed in table 4 by running our aforementioned post-processing script, as well as their corresponding data tables in our primary paper [44]. In addition, our post-processing script generates a collection of HTML files which contain all data tables reported in our primary paper [44].

Finally, raw data files and processed data files shown in tables 4 and 16 respectively could be loaded into any spreadsheet software to carry-out any further data analysis or confirming the reproducibility of the experiments and results reported by our primary paper [44].

Step	Recovering the output files generated by our Reprozip-based experiments
1	user@server$ reprounzip showfiles docker_folder
2	user@server$ sudo reprounzip docker download −−all docker_folder

Table 14: Detailed instruction to recover the output files generated by our Reprozip-based experiments. The first instruction shows a list with the output files generated by the experiments, whilst the second one extracts all the output files from the container and downloads them onto the current folder. You should obtain all raw output files listed in table 4.

Step	Detailed post-processing instructions based on a R-language script
(1)	Launch the R statistical program and install *knitr* and *readr* packages.
(2)	Launch the R statistical program (you could also use R-Studio).
(3)	Select the menu option "*File->Open script*". Then, load the R-language script file called *embeddings_vs_ontomeasures_final_tables.R* contained in the folder shown in figure 2. Latest version of the aforementioned script should be obtained from HESML GitHub distribution or our companion reproducibility dataset [46].
(4)	Edit the *rootDir*, *inputDir* and *outputDir* variables at the beginning of the script in order to set the directory which contains the raw output files onto your hard drive, as well as the directory in which will be saved the final assembled data tables as reported in our primary paper [44]. IMPORTANT NOTE: *inputDir* and *outputDir* variables should end with slash '/' symbol.
(5)	Select the menu option "*Edit->Run all*". The final assembled data tables will be saved in the output directories defined above, as detailed in table 16. In addition, the aforementioned R script creates a and opens a collection of HTML files which show all data tables in our primary paper [44] and detailed in table 16.

Table 15: Detailed instructions for the post-processing of the raw output files generated by our experiments. R-language script computes all Pearson, Spearman and Harmonic score metrics and generates a HTML report file reproducing all data tables reported by our primary paper Lastra-Díaz et al. [44].

4. Extending and reusing our reproducible experiments

Our reproducible experiments are based on the XML-based HESML experiment file format (*.exp) whose specification is detailed by the "*WordNet-BasedExperiments.xsd*" schema file distributed with HESML library as shown in figure 2. Both *.exp experiment files and *.xsd schema file were created with XML Spy editor. Next paragraphs provide a detailed description of the main objects encoded by the HESML XML-based experiment file format and how they could be used to create new experiments from scratch as those introduced herein.

HESML XML-based experiment file format. Figure 3 shows a sample file which has been extracted from the "benchmark_survey.exp" file encoding all reproducible experiments introduced herein. *WordNetBasedExperiments* is the root node which contains the collection of word similarity or relatedness benchmarks to be evaluated, whilst *SingleDatasetSimilarityValuesExperiment* encodes a specific word similarity benchmark defined by a dataset, an output directory and a collection of WordNet-based similarity measures and pre-trained word embed-

File#	Post-processing output files saved at “outputDir” directory	In primary paper [44]
1	table_Pearson_SimDatasets.csv	table 4 (full precision)
2	table_Pearson_SimDatasets_rounded.csv	table 4
3	table_Spearman_SimDatasets.csv	table 5 (full precision)
4	table_Spearman_SimDatasets_rounded.csv	table 5
5	table_Pearson_RelDatasets.csv	table 6 (full precision)
6	table_Pearson_RelDatasets_rounded.csv	table 6
7	table_Spearman_RelDatasets.csv	table 7 (full precision)
8	table_Spearman_RelDatasets_rounded.csv	table 7
9	table_joined_allEmbeddings_similarity.csv	table 8 (full precision)
10	table_joined_allEmbeddings_similarity_rounded.csv	table 8
11	table_joined_allEmbeddings_relatedness.csv	table 9 (full precision)
12	table_joined_allEmbeddings_relatedness_rounded.csv	table 9
13	table_pvalues_AttractReppel_allembeddings_similarity.csv	table A.1
14	table_pvalues_Paragramws_allembeddings_relatedness.csv	table A.2
15	table_AvgMeasures_Pearson_SimDatasets.csv	table A.3 (full precision)
16	table_AvgMeasures_Pearson_SimDatasets_rounded.csv	table A.3
17	table_AvgMeasures_Spearman_SimDatasets.csv	table A.4 (full precision)
18	table_AvgMeasures_Spearman_SimDatasets_rounded.csv	table A.4
19	table_AvgMeasures_Pearson_RelDatasets.csv	table A.5 (full precision)
20	table_AvgMeasures_Pearson_RelDatasets_rounded.csv	table A.5
21	table_AvgMeasures_Spearman_RelDatasets.csv	table A.6 (full precision)
22	table_AvgMeasures_Spearman_RelDatasets_rounded.csv	table A.6

Table 16: Collection of processed output files generated by the execution of the “embeddings_vs_ontomeasures_final_tables.R” script file onto the *outputDir* directory and their corresponding tables in our primary work [44].

ding models. *SimilarityMeasures* nodes encode ontology-based semantic similarity measures based on WordNet which could require a further Information Content (IC) model for its implementation, being declared below the *WordNetMeasures* node. Likewise, *RawWordVectorFiles* encodes the collection of pre-trained word embedding files to be evaluated in the same dataset. Both *SimilarityMeasures* and *RawWordVectorFiles* could be declared independently, and they could contain an unlimited number of methods to be evaluated. Latest HESML version supports three different pre-trained word embeddings file formats which are defined by the *EmbVectorFiles*, *UKBVectorFiles* and *NasariVectorFiles* nodes. Raw output files which are generated by *SingleDatasetSimilarityValuesExperiment* procedures contain a matrix of values encoding the raw similarity value reported by each method for each word pair in the similarity dataset being evaluated.

XML-based HESML experiment file sample

```
<WordNetBasedExperiments>
    <SingleDatasetSimilarityValuesExperiment>
        <OutputFileName>raw_similarity_values_MC28_dataset.csv</OutputFileName>
        <DatasetDirectory>../WN\_Datasets</DatasetDirectory>
        <DatasetFileName>Miller_Charles_28_dataset.csv</DatasetFileName>
        <WordNetMeasures>
            <WordNetDatabaseFileName>data.noun</WordNetDatabaseFileName>
            <WordNetDatabaseDirectory>../Wordnet-3.0/dict</WordNetDatabaseDirectory>
            <SimilarityMeasures>
                <SpecificSimilarityMeasure>
                    <MeasureType>JiangConrath</MeasureType>
                    <IntrinsicICModel>Sanchez2011</IntrinsicICModel>
                </SpecificSimilarityMeasure>
                <SpecificSimilarityMeasure>
                    <MeasureType>Rada</MeasureType>
                </SpecificSimilarityMeasure>
            </SimilarityMeasures>
        </WordNetMeasures>
        <RawWordVectorFiles>
            <EmbVectorFiles>
                <VectorFile>../WordEmbeddings/attract-reppel.emb</VectorFile>
            </EmbVectorFiles>
            <UKBVectorFiles>
                <VectorFile>../WordEmbeddings/wordnet-ukb.ppv</VectorFile>
            </UKBVectorFiles>
            <NasariVectorFiles>
                <NasariVectorFile>
                    <SensesFile>../WordEmbeddings/nasari/en_wordsenses_BN.txt</SensesFile>
                    <VectorFile>../WordEmbeddings/nasari/nasari-unified</VectorFile>
                </NasariVectorFile>
            </NasariVectorFiles>
        </RawWordVectorFiles>
    </SingleDatasetSimilarityValuesExperiment>
</WordNetBasedExperiments>
```

Figure 3: XML source code above shows an example of a HESML reproducible experiment on word similarity and relatedness. Source code above has been extracted from the "benchmark_survey.exp" file which encodes all experiments reported in our primary paper [44].

Extending or modifying our experiments. Anyone could use our main aforementioned experiment file as a template to create new experiments from scratch by evaluating other sets of available ontology-based semantic similarity measures based on WordNet, pre-trained word embedding models or word similarity datasets not considered herein. For instance, any reader could evaluate any ontology-based methods by declaring it in any *SimilarityMeasures* node whenever this method have been previously implemented in HESML and its keyname being specified by the *SimilarityMeasureType* enumeration in the "*WordNet-BasedExperiments.xsd*" schema file. Likewise, currently supported IC models are specified by the *IntrinsicICModelType* enumeration in the aforementioned XML schema file. On the other hand, any reader could evaluate any unconsidered pre-trained word embedding model by declaring a new method in the *RawWordVectorFiles*, whenever its corresponding pre-trained model being provided in any of the three file formats which are currently supported, otherwise there would be needed to extend HESML to support a new pre-trained word embedding file format. Finally, any reader could define any new benchmark considering a different set of word similarity datasets by declaring further *SingleDatasetSimilarityValuesExperiment* nodes with their corresponding dataset files in comma-separated file format. For a detailed list of the methods currently implemented by HESML V1R4, we refer the readers to its release notes [42].

5. Conclusions and future work

This work introduces for the first time a large set of reproducible experiments on word similarity and relatedness including most methods in the families of ontology-based semantic similarity measures based on WordNet and word embedding models. This aforementioned set of experiments allow that all experiments and results introduced by Lastra-Díaz et al. [44] to be reproduced exactly. Likewise, our reproducible experiments could be easily extended or modified to create new benchmarks from scratch which evaluate a different set of methods and word similarity and relatedness datasets from those considered herein. For this reason, we hope that this set of reproducible benchmarks to become into a de facto standard experimentation platform for any future research on word similarity and relatedness.

As forthcoming activities, we plan the study and proposal of new distributional similarity and relatedness measures, as well as their use in the definition of sentence and short-text similarity measures.

Acknowledgements

We are grateful of Fernando González and Juan Corrales for setting up our UNED Dataverse dataset, Yuanyuan Cai for answering kindly our questions to replicate their IC-based similarity measures and IC models in HESML, and http://clouding.io for their technical support to set up our experimental platform. We are also very thankful to José Camacho-Collados for providing the weighting overlap source code which we have integrated into HESML for measuring the similarity between the NASARI vectors. This work has been partially supported by the Spanish project VEMODALEN (TIN2015-71785-R), the Basque Government (type A IT1343-19), BBVA BigKnowledge bigknowledge project and the Spanish Research Agency LIHLITH project (PCIN-2017-118 / AEI) in the framework of EU ERA-Net CHIST-ERA.

References

[1] Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., Soroa, A., 2009. A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In: Proc. of Human Language Technologies: The 2009 Annual Conf. of the North American Chapter of the Association for Computational Linguistics. NAACL '09. ACL, Stroudsburg, PA, USA, pp. 19–27.

[2] Agirre, E., Soroa, A., 2009. Personalizing pagerank for word sense disambiguation. In: Proc. of the 12th Conf. of the European Chapter of the Association for Computational Linguistics. ACL, pp. 33–41.

[3] Ben Aouicha, M., Hadj Taieb, M. A., Nov. 2015. G2WS: Gloss-based WordNet and Wiktionary semantic Similarity measure. In: 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA). pp. 1–7.

[4] Ben Aouicha, M., Hadj Taieb, M. A., 2016. Computing semantic similarity between biomedical concepts using new information content approach. Journal of Biomedical Informatics 59, 258–275.

[5] Ben Aouicha, M., Hadj Taieb, M. A., Ben Hamadou, A., Nov. 2016. SISR: System for integrating semantic relatedness and similarity measures. Soft Computing, 1–25.

[6] Ben Aouicha, M., Hadj Taieb, M. A., Ben Hamadou, A., Mar. 2016. Taxonomy-based information content and wordnet-wiktionary-wikipedia glosses for semantic relatedness. Applied Intelligence, 1–37.

[7] Ben Aouicha, M., Hadj Taieb, M. A., Beyaoui, S., Aug. 2016. Distributional semantics study using the co-occurrence computed from collaborative resources and WordNet. In: 2016 International Symposium on INnovations in Intelligent SysTems and Applications (INISTA). pp. 1–8.

[8] Bojanowski, P., Grave, E., Joulin, A., Mikolov, T., Jul. 2016. Enriching Word Vectors with Subword Information. arXiv:1607.04606.

[9] Branco, A., Cohen, K. B., Vossen, P., Ide, N., Calzolari, N., Mar. 2017. Replicability and reproducibility of research results for human language technology: introducing an LRE special section. Language Resources and Evaluation 51 (1), 1–5.

[10] Bruni, E., Tran, N.-K., Baroni, M., 2014. Multimodal Distributional Semantics. Journal of Artificial Intelligence Research 49 (1), 1–47.

[11] Camacho-Collados, J., Pilehvar, M. T., Navigli, R., 2016. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. Artif. Intell. 240, 36–64.

[12] Chirigati, F., Capone, R., Rampin, R., Freire, J., Shasha, D., Mar. 2016. A collaborative approach to computational reproducibility. Information Systems 59, 95–97.

[13] Chirigati, F., Rampin, R., Shasha, D., Freire, J., 2016. ReproZip: computational reproducibility with ease. In: Proc. of the ACM Intl. Conf. on Management of Data (SIGMOD). Vol. 16. pp. 2085–2088.

[14] Fariña, A., Martínez-Prieto, M. A., Claude, F., Navarro, G., Lastra-Díaz, J. J., Prezza, N., Seco, D., Apr. 2019. On the reproducibility of experiments of indexing repetitive document collections. Information systems 83, 181–194.

[15] Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppin., E., 2002. Placing search in context: the concept revisited. ACM Trans. on Information Systems 20 (1), 116–131.

[16] Fokkens, A., Van Erp, M., Postma, M., Pedersen, T., Vossen, P., Freire, N., Aug. 2013. 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, pp. 1691–1701.

[17] Gerz, D., Vulić, I., Hill, F., Reichart, R., Korhonen, A., Nov. 2016. SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity. In: Proc. of EMNLP. Austin, Texas, pp. 2173–2182.

[18] Goikoetxea, J., Agirre, E., Soroa, A., 2014. Exploring the Use of Word Embeddings and Random Walks on Wikipedia for the CogAlex Shared Task. Proc. of the 4th Workshop on Cognitive Aspects of the Lexicon, 31–34.

[19] Goikoetxea, J., Agirre, E., Soroa, A., 2016. Single or Multiple? Combining Word Representations Independently Learned from Text and WordNet. In: Proc. of AAAI. pp. 2608–2614.

[20] Goikoetxea, J., Soroa, A., Agirre, E., 2015. Random Walks and Neural Network Language Models on Knowledge Bases. In: Proc. of HLT-NAACL. pp. 1434–1439.

[21] Goikoetxea, J., Soroa, A., Agirre, E., Jul. 2018. Bilingual Embeddings with Random Walks over Multilingual Wordnets. Knowledge-Based Systems 150 (15), 218–230.

[22] Hadj Taieb, M. A., Ben Aouicha, M., Ben Hamadou, A., Nov. 2014. A new semantic relatedness measurement using WordNet features. Knowledge and Information Systems 41 (2), 467–497.

[23] Hadj Taieb, M. A., Ben Aouicha, M., Ben Hamadou, A., Nov. 2014. Ontology-based approach for measuring semantic similarity. Engineering Applications of Artificial Intelligence 36, 238–261.

[24] Hadj Taieb, M. A., Ben Aouicha, M., Bourouis, Y., Jun. 2015. FM3S: Features-Based Measure of Sentences Semantic Similarity. In: Onieva, E., Santos, I., Osaba, E., Quintián, H., Corchado, E. (Eds.), Proceedings of the 10th International Conference on Hybrid Artificial Intelligent Systems (HAIS 2015). Vol. 9121 of LNCS. Springer, Bilbao, Spain, pp. 515–529.

[25] Hadj Taieb, M. A., Ben Aouicha, M., Tmar, M., Ben Hamadou, A., 2012. New WordNet-based semantic relatedness measurement. In: GWC 2012 6th International Global Wordnet Conference. academia.edu, p. 126.

[26] Hadj Taieb, M. A., Ben Aouicha, M., Tmar, M., Ben Hamadou, A., Nov. 2012. Wikipedia Category Graph and New Intrinsic Information Content Metric for Word Semantic Relatedness Measuring. In: Proc. of the Third International Conference on Data and Knowledge Engineering (ICDKE). Lecture Notes in Computer Science. Springer Berlin Heidelberg, Wuyishan, Fujian, China, pp. 128–140.

[27] Hadj Taieb, M. A., Ben Aouicha, M., Tmar, M., Hamadou, A. B., 2011. New information content metric and nominalization relation for a new WordNet-based method to measure the semantic relatedness. In: Cybernetic Intelligent Systems (CIS), 2011 IEEE 10th International Conference on. pp. 51–58.

[28] Halawi, G., Dror, G., Gabrilovich, E., Koren, Y., 2012. Large-scale Learning of Word Relatedness with Constraints. In: Proc. of ACM SIGKDD. ACM, New York, NY, USA, pp. 1406–1414.

[29] Harispe, S., Ranwez, S., Janaqi, S., Montmain, J., Mar. 2014. The semantic measures library and toolkit: fast computation of semantic similarity and relatedness using biomedical ontologies. Bioinformatics 30 (5), 740–742.

[30] Hill, F., Reichart, R., Korhonen, A., Dec. 2015. SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. Computational Linguistics 41 (4), 665–695.

[31] Hoffgaard, F., Weil, P., Hamacher, K., Apr. 2010. BioPhysConnectoR: Connecting sequence information and biophysical models. BMC bioinformatics 11, 199.

[32] Huang, E. H., Socher, R., Manning, C. D., Ng, A. Y., 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In: Proc. of the Annual Meeting of the ACL. Vol. 1. pp. 873–882.

[33] Lastra-Díaz, J. J., Sep. 2014. Intrinsic Semantic Spaces for the representation of documents and semantic annotated data. Master's thesis, Universidad Nacional de Educación a Distancia (UNED). Department of Computer Languages and Systems, http://e-spacio.uned.es/fez/view/bibliuned:master-ETSIInformatica-LSI-Jlastra.

[34] Lastra-Díaz, J. J., Sep. 2017. Recent Advances in Ontology-based Semantic Similarity Measures and Information Content Models based

on WordNet. Ph.D. thesis, Universidad Nacional de Educación a Distancia (UNED), http://e-spacio.uned.es/fez/view/tesisuned:ED-Pg-SisInt-Jjlastra.

[35] Lastra-Díaz, J. J., García-Serrano, A., Nov. 2015. A new family of information content models with an experimental survey on WordNet. Knowledge-Based Systems 89, 509–526.

[36] Lastra-Díaz, J. J., García-Serrano, A., Nov. 2015. A novel family of IC-based similarity measures with a detailed experimental survey on WordNet. Eng. App. of Artif. Intell. 46, 140–153.

[37] Lastra-Díaz, J. J., García-Serrano, A., Jul. 2016. A refinement of the well-founded Information Content models with a very detailed experimental survey on WordNet. Tech. Rep. TR-2016-01, UNED, http://e-spacio.uned.es/fez/view/bibliuned:DptoLSI-ETSI-Informes-Jlastra-refinement.

[38] Lastra-Díaz, J. J., García-Serrano, A., Nov. 2016. HESML_vs_SML: scalability and performance benchmarks between the HESML V1R2 and SML 0.9 semantic measures libraries. Mendeley Data, v1, http://doi.org/10.17632/5hg3z85wf4.1.

[39] Lastra Díaz, J. J., García Serrano, A., Jun. 2016. 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.

[40] Lastra-Díaz, J. J., García-Serrano, A., 2016. WNSimRep: a framework and replication dataset for ontology-based semantic similarity measures and information content models. Mendeley Data v1, http://doi.org/10.17632/mpr2m8pycs.1.

[41] Lastra-Díaz, J. J., García-Serrano, A., 2016. WordNet-based word similarity reproducible experiments based on HESML V1R1 and ReproZip. Mendeley Data, v1, http://doi.org/10.17632/65pxgskhz9.1.

[42] Lastra-Díaz, J. J., García Serrano, A., 2018. HESML V1R4 Java software library of ontology-based semantic similarity measures and information content models. Mendeley Data, v4, http://dx.doi.org/10.17632/t87s78dg78.4.

[43] Lastra-Díaz, J. J., García-Serrano, A., Batet, M., Fernández, M., Chirigati, F., Jun. 2017. HESML: a scalable ontology-based semantic similarity measures library with a set of reproducible experiments and a replication dataset. Information Systems 66, 97–118.

[44] Lastra-Díaz, J. J., Goikoetxea, J., Hadj Taieb, M. A., García-Serrano, A., Ben Aouicha, M., Agirre, E., Oct. 2019. A reproducible survey on word embeddings and ontology-based methods for word similarity: linear combinations outperform the state of the art. Engineering Applications of Artificial Intelligence 85, 645–665.

[45] Lastra-Díaz, J. J., Goikoetxea, J., Hadj Taieb, M. A., García-Serrano, A., Ben Aouicha, M., Agirre, E., Aug. 2019. Reproducibility dataset for a large experimental survey on word embeddings and ontology-based methods for word similarity. Data in Brief.

[46] Lastra-Díaz, J. J., Goikoetxea, J., Hadj Taieb, M. A., García-Serrano, A., Ben Aouicha, M., Agirre, E., 2019. Word similarity benchmarks of recent word embedding models and ontology-based semantic similarity measures. e-cienciaDatos, v1, http://dx.doi.org/10.21950/AQ1CVX.

[47] Luong, T., Socher, R., Manning, C. D., 2013. Better word representations with recursive neural networks for morphology. In: Proc. of CoNLL. pp. 104–113.

[48] Merkel, D., Mar. 2014. Docker: Lightweight Linux Containers for Consistent Development and Deployment. Linux Journal 2014 (239), Article No. 2.

[49] Mikolov, T., Chen, K., Corrado, G., Dean, J., May 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781.

[50] Miller, G. A., 1995. WordNet: A Lexical Database for English. Communications of the ACM 38 (11), 39–41.

[51] Miller, G. A., Charles, W. G., 1991. Contextual correlates of semantic similarity. Language and cognitive processes 6 (1), 1–28.

[52] Mrkšić, N., Vulić, I., Séaghdha, D. Ó., Leviant, I., Reichart, R., Gašić, M., Korhonen, A., Young, S., 2017. Semantic Specialisation of Distributional Word Vector Spaces using Monolingual and Cross-Lingual Constraints. Trans. of the ACL 5, 309–324.

[53] Munafò, M. R., Nosek, B. A., Bishop, D. V. M., Button, K. S., Chambers, C. D., du Sert, N. P., Simonsohn, U., Wagenmakers, E.-J., Ware, J. J., Ioannidis, J. P. A., Jan. 2017. A manifesto for reproducible science. Nature Human Behaviour 1, 0021.

[54] Pedersen, T., 2008. Empiricism Is Not a Matter of Faith. Computational Linguistics 34 (3), 465–470.

[55] Pedersen, T., 2008. WordNet-InfoContent-3.0.tar dataset repository. https://www.researchgate.net/publication/273885902_WordNet-InfoContent-3.0.tar.

[56] Pedersen, T., Nov. 2013. Measuring the Similarity and Relatedness of Concepts: a MICAI 2013 Tutorial.

[57] Pennington, J., Socher, R., Manning, C. D., 2014. Glove: Global vectors for word representation. Proc. of EMNLP 12, 1532–1543.

[58] Pirró, G., Nov. 2009. A semantic similarity metric combining features and intrinsic information content. Data & Knowledge Engineering 68 (11), 1289–1308.

[59] Radinsky, K., Agichtein, E., Gabrilovich, E., Markovitch, S., Mar. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In: Proc. of the Intl. Conf. on WWW. ACM, pp. 337–346.

[60] Rubenstein, H., Goodenough, J. B., Oct. 1965. Contextual Correlates of Synonymy. Communications of the ACM 8 (10), 627–633.

[61] Schwartz, R., Reichart, R., Rappoport, A., 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In: Proc. of the Conf. on Computational Natural Language Learning. pp. 258–267.

[62] Sefraoui, O., Aissaoui, M., Eleuldj, M., 2012. OpenStack: toward an open-source solution for cloud computing. International Journal of Computer Applications in Technology 55 (3), 38–42.

[63] Szumlanski, S. R., Gomez, F., Sims, V. K., Aug. 2013. A New Set of Norms for Semantic Relatedness Measures. In: Proc. of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'2013). Vol. 2. aclweb.org, Sofia, Bulgaria, pp. 890–895.

[64] Wieling, M., Rawee, J., van Noord, G., Dec. 2018. Reproducibility in Computational Linguistics: Are We Willing to Share? Computational Linguistics 44 (4), 641–649.

[65] Wieting, J., Bansal, M., Gimpel, K., Livescu, K., Roth, D., 2015. From Paraphrase Database to Compositional Paraphrase Model and Back. Trans. of the ACL 3, 345–358.

[66] Wolke, A., Bichler, M., Chirigati, F., Steeves, V., Jul. 2016. Reproducible experiments on dynamic resource allocation in cloud data centers. Information Systems 59, 98–101.

[67] Yang, D., Powers, D. M., 2006. Verb similarity on the taxonomy of WordNet. In: Proc. of the 3th Intl. WordNet Conf. (GWC). pp. 121–128.

[68] Yeh, E., Ramage, D., Manning, C. D., Agirre, E., Soroa, A., 2009. WikiWalk: Random Walks on Wikipedia for Semantic Relatedness. In: Proc. of the 2009 Workshop on Graph-based Methods for Natural Language Processing. TextGraphs-4. ACL, Stroudsburg, PA, USA, pp. 41–49.