Appendix B. The reproducible experiments on word similarity

This appendix introduces a detailed experimental setup based on a collection of publicly available software tools and reproducibility resources with the aim of exactly replicating all experiments, results and data tables reported in our main paper [18].

All experiments carried-out in our aforementioned work were implemented in *HESML V1R4* [16] by running *HESMLclient* program with an unique reproducible experiment file in XML-based (*.exp) file format which encodes the evaluation of all semantic measures in all datasets as listed in table B.7. HESML V1R4 implements all ontology-based semantic similarity measures based on WordNet as well as the evaluation of all pre-trained word embedding models evaluated in our benchmarks [18]. In addition, execution of our experiments was recorded into a Reprozip [6] file, called '*WN_ontology_measures_vs_embeddings.rpz*', which is publicly available at the UNED Dataverse repository [19]. This later Reprozip file can be reproduced in any Reprozip compliant platform[1] as detailed in section Appendix B.4.2. Thus, all our methods, experiments and results can be reproduced using two different software platforms and methods as detailed in table B.1.

Sotware used	Supported reproducibility methods
HESMLclient [16]	You must download HESML V1R4 [16] and a complementary ZIP file containing the collection of pre-trained word embedding files (*WordEmbeddings.zip* [19]), and then running *HESMLclient* with the reproducible file as input, as detailed in section Appendix B.4.1.
ReproUnzip [6]	You must download our complementary Reprozip package [19] and setting up and running Reprounzip as detailed in Appendix B.4.2.

Table B.1: Our two methods to reproduce all experiments, results and data tables reported in our main paper [18].

These two aforementioned reproducibility methods were introduced in a previous paper [17] in which we reproduce all results, data tables and figures reported in three papers previously introduced by Lastra-Díaz and García-Serrano [13, 14, 15].

All experiments reported by Lastra-Díaz et al. [18] and detailed herein were generated on an UBUNTU 16.04 virtual computer with 8 Gb of RAM and 100 Gb of disk space. 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 and Windows-based platforms.

HESML V1R4 distribution [16] contains all source files and compiled versions of the *HESML-V1R4.jar* library and *HESMLclient.jar* Java console program. Thus, it is enough to download its official distribution from Mendeley [16] or GitHub[2] in order to run our experiments. However, for the sake of complete-

[1] https://www.reprozip.org/
[2] https://github.com/jjlastra/HESML.git

ness Appendix B.3 introduces the detailed steps to obtain and compile HESML V1R4. Finally, we provide a full reproducibility dataset at the UNED Dataverse repository [19] which contains the files detailed in table B.2.

ID	UNED Dataverse repository: files name and description
1	*appendix-reproducible-experiments.pdf* A copy of this document.
2	*benchmark_survey.exp* HESML reproducible experiment file which allows to reproduce all our experiments and results by running HESMLclient.
3	*embeddings_vs_ontomeasures_final_tables.R* A post-processing R script file which processes all raw similarity files and generates a collection of Comma Separated (CSV) files containing all data tables in our main paper [18].
4	*processed_output_benchmarks.zip* This ZIP file containing all processed CSV files generated by our post-processing R script.
5	*raw_output_benchmark_all_datasets.zip* This ZIP file contains all raw output similarity files produced by the execution of HESMLclient with our reproducible experiment file. Thus, it contains all our raw experimental data.
6	*WN_ontology_measures_vs_embeddings.rpz* Reprozip file to reproduce all our experiments in the long-term on any Reprozip compliant platform regardless the availability of the original platform used in our experiments.
7	*WordEmbeddings.zip* This ZIP file contains all pre-trained word embedding models evaluated in our experiments.
8	*Word_Similarity_Datasets.zip* This ZIP file contains all word similarity datasets evaluated in our experiments.

Table B.2: Content of our complementary reproducibility package publicly available at the UNED Dataverse repository [19].

The rest of the document is structured as follows. Appendix B.1 briefly introduces HESML and Reprozip. Appendix B.2 introduces the system requirements to reproduce our experiments, as well as the running times obtained in our testing platforms. Appendix B.3 details how to obtain and compile HESML V1R4. Appendix B.4 introduces the detailed steps of the two provided methods to run our experiments. Finally, Appendix B.5 details the post-processing steps to generate the final data tables shown in our main paper [18] from the raw

output files detailed in table B.7.

Appendix B.1. About HESML and Reprozip

HESML [17] is a self-contained Java software library of semantic measures based on WordNet whose latest version, called HESML V1R4 [16], also supports the evaluation of pre-trained word embedding files, such as those generated by word embedding models introduced by Mikolov et al. [22], Pennington et al. [27], Schwartz et al. [31], Wieting et al. [33], Goikoetxea et al. [9], Bojanowski et al. [3], Agirre and Soroa [2], Camacho-Collados et al. [5] and Mrkšić et al. [24].

HESML source code data	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	This work and the sample source code in the HESMLclient program.
Community forum for questions.	hesml+subscribe@googlegroups.com, hesml+unsubscribe@googlegroups.com

Table B.3: Summary of technical and legal information of the HESML software library.

HESML is a self-contained experimentation platform on word similarity 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 called (*.exp). Despite latest version of HESML only supports WordNet, it could be easily extended to manage other ontologies by implementing the proper parsers. HESML has been completely developed in NetBeans 8 and Java 8, being distributed with all 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 any reader to its introductory paper [17].

On the other hand, ReproZip is a virtualization tool introduced by Chirigati et al. [6], 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. Finally, other very 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.

Appendix B.2. System requirements and performance evaluation

Table B.4 shows the platforms in which we have successfully reproduced the experiments detailed herein, whilst table B.5 show their running times in the completion of all experiments for each aforementioned reproducibility method. 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 computer detailed in table B.4, 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 platform shown in table B.4 is based on a minimal overall disk space of 200 Gb, instead of 100 Gb as required by the aforementioned reproducibility method, with the aim of setting up UBUNTU, Docker and the resources required by our Reprozip package.

Platform	Operating system	Configuration
Ubuntu-base	Ubuntu 16.04	1 Core CPU, 8 Gb RAM, 100 Gb SSD disk
Ubuntu-Reprounzip	Ubuntu 16.04	1 Core CPU, 8 Gb RAM, 200 Gb SSD disk
Windows-base	Windows 10x64	1 Core CPU, 8 Gb RAM, 100 Gb SSD disk

Table B.4: Virtual computers successfully used to reproduce our experiments.

Platform	Method	Running time
Ubuntu-base	HESMLclient	17581 min $\approx$ 12.2 days
Ubuntu-Reprounzip	ReproUnzip	18109 $\approx$ 12.6 days
Windows-base	HESMLclient	10 days

Table B.5: Running times for the execution of all benchmarks by using HESMLclient program with the 'benchmark_survey.exp' experiment file, or by running the ReproUnzip with the 'WN_ontology_measures_vs_embeddings.rpz' file.

Appendix B.3. Obtaining and compiling HESML

Table B.3 shows the technical information required to obtain and compile the *HESML* source code and run the experiments detailed in table B.7. HESML V1R4 distribution includes compiled versions of *HESML* library and the *HESMLclient* program, thus this later program can be directly used without 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 [16]; or (2) by downloading it from its GitHub repository detailed in table B.3. Once the source code HESML ZIP package has been downloaded and extracted onto your hard drive, the project will have the folder structure shown in table B.6 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

```
HESML_Library/
├── HESML/
├── HESMLclient/
├── PedersenICmodels/
├── ReproducibleExperiments/
│   ├── AI_paper/
│   ├── EAAI_paper/
│   ├── Embeddings_vs_OntologyMeasures_paper/
│   │   ├── benchmark_survey.exp
│   │   ├── RawOutputFiles/
│   │   └── ProcessedOutputFiles/
│   ├── KBS_paper/
│   ├── Post-scripts/
│   │   └── embeddings_vs_ontomeasures_final_tables.R
│   ├── RawOutputFiles/
│   ├── All_paper_tables.exp
│   └── WordNetBasedExperiments.xsd
├── WordEmbeddings(*)/
├── WN_Datasets/
├── WordNet-2.1/
├── WordNet-3.0/
└── WordNet-3.1/
```

Table B.6: Directory structure of the HESML library once it has been extracted onto disk. The reproducible experiment file and the post-processing R script used to reproduce and generate the final data tables in [18] respectively are shown in bold. (*) WordEmbeddings folder contains the pre-trained files for all word embedding models used in our experiments [18]; however, this folder is neither included by the HESML V1R4 Lastra-Díaz and García Serrano [16] distribution nor HESML V1R4 release at GitHub repository because its large size. Thus, you must download the *WordEmbeddings.zip* file [19] and extract it onto the main *HESML_Library* directory to retrieve this folder and its content.

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 [25]. 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 [26].

ReproducibleExperiments folder contains one subfolder for each paper introduced by Lastra-Díaz and García-Serrano [13, 14, 15] and the work reproduced herein [18]. 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 also contains the *RawOutputFiles* subfolder with all the raw output files of the three aforementioned papers.

Post-scripts folder contains a set of post-processing R scripts which process the raw output files generated by all reproducible experiments with the aim of reproducing all final data tables and figures reported in the 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 B.7, 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 B.7. In addition, this folder contains a subfolder called 'RawOutputFiles' containing all raw output similarity files generated by our experiments. The R script file called 'embeddings_vs_ontomeasures_final_tables.R' generates all files in 'ProcessedOutputFiles' subfolder. However, last version of our post-processing file at our Dataverse repository [19], called *embeddings_vs_ontomeasures_final_tables.R*, generates three subfolders during its execution with the aim of arranging in a more pleasant way all final data tables reported in our main paper [18].

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.

Appendix B.4. Running the experiments

Table B.7 shows the full collection of reproducible experiments encoded by the 'benchmark_survey.exp' file, as well as the corresponding raw output files that are generated during its execution. The subsequent processing of these raw output files allows to reproduce the results reported in our main paper [18] exactly, as detailed in Appendix B.5.

Before to run the experiments, you must download the *WordEmbeddings.zip* file [19] and extract it onto the main *HESML_Library* directory. This later file contains all pre-trained word embedding files; however, because of its large size and the space limitations of GitHub and Mendeley repositories it could not be included in any of both HESML distributions.

Appendix B.4.1. Running the experiments with HESMLclient

Once you have downloaded the *HESML V1R4* library as detailed in section Appendix B.3, 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* ahown in table B.7. Thus, you only need to follow the steps below:

Dataset	Raw output files generated
MC28 [23]	raw_similarity_values_MC28_dataset.csv
RG65 [30]	raw_similarity_values_RG65_dataset.csv
PS_{full} [28]	raw_similarity_values_PSfull_dataset.csv
Agirre201 [1]	raw_similarity_values_Agirre201_lowercase_dataset.csv
SimLex665 [11]	raw_similarity_values_SimLex665_dataset.csv
MTurk771 [10]	raw_similarity_values_MTurk771_dataset.csv
MTurk287/235 [29]	raw_similarity_values_MTurk287-235_dataset.csv
WS353Rel [7]	raw_similarity_values_WS353Rel_dataset.csv
Rel122 [32]	raw_similarity_values_Rel122_dataset.csv
WS353Full [7]	raw_similarity_values_WS353Full_dataset.csv
SimLex111 [11]	raw_similarity_values_SimLex111_dataset.csv
SimLex222 [11]	raw_similarity_values_SimLex222_dataset.csv
SimLex999 [11]	raw_similarity_values_SimLex999_dataset.csv
SimVerb3500 [8]	raw_similarity_values_SimVerb3500_dataset.csv
MEN [4]	raw_similarity_values_MEN_dataset.csv
YP130 [34]	raw_similarity_values_YP130_dataset.csv
RW2034 [20]	raw_similarity_values_RareWords2034_dataset.csv
RW1401 [20]	raw_similarity_values_RareWords1401_dataset.csv
SCWS [12]	raw_similarity_values_SCWS1994_dataset.csv

Table B.7: 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 in R to produce the final data tables shown in our main paper [18].

(1) Open a Linux or Windows command prompt in the *HESML_Library/HESMLclient* directory.
(2) Run the following command with the main reproducible experiment file:

prompt:> java -jar -Xms4096m dist/HESMLclient.jar ../ReproducibleExperiments/Embeddings_vs_OntologyMeasures_paper/benchmark_survey.exp

Finally, the execution of the command in step 2 above will generate all raw output files listed in table B.7.

Appendix B.4.2. Running the ReproZip experiments

The ReproZip[3] 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, generating a Reprozip package called *WN_ontology_measures_vs_embeddings.rpz* file which is publicly available at [19]. This later ReproZip file was generated by running Reprozip on the Ubuntu-base workstation detailed in table B.4; however, in order to run ReproUnzip based on Docker as detailed below is needed to set up a Ubuntu-Reprounzip platform (see table B.4). 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.

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

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 apt-get install python-pip
9	sudo apt-get install screen
10	sudo pip install reprounzip[all]
11	sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
12	curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
13	sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
14	sudo apt-get update
15	sudo apt-get install docker-ce

Table B.8: Detailed instructions on installing ReproUnzip with Docker for Ubuntu.

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 [21], in which the author introduces Docker and compares it with classic Virtual Machines (VM) such as VirtualBox.

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 B.8, despite several steps possibly being unnecessary depending on your starting configuration. Once ReproUnzip and Docker have been successfully installed, table B.9 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[4].

The running of the reproducible experiments based on Docker for Ubuntu took around 13 days on the aforementioned Ubuntu-base workstation. Once the running has finished, you should follow the instructions shown in table B.10 to recover the raw output files from the Docker container, as listed in table B.7. Finally, table B.5 summarizes the software platforms in which the reproducible experiments [19] have been successfully reproduced.

Appendix B.5. Processing of the result files

The running of the *benchmark_survey.exp* experiment file generates the collection of comma-separated files (*.csv) listed in table B.7, whose values are

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

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 B.8.
2	Download the *WN_ontology_measures_vs_embeddings.rpz* from its UNED Dataverse repository [19]. For instance, you can execute the following command: wget https://edatos.consorciomadrono.es/api/access/datafile/1845?gbrecs=true&key=6f853fd4-9545-4fd3-867d-5437345cd2a7 -O WN_ontology_measures_vs_embeddings.rpz
3	Open a command console in the directory containing the 'WN_ontology_measures_vs_embeddings.rpz' file and executes the six commands below. - First, we must setup the docker container as detailed below. Step (1) could take up to 45 minutes depending of your platform). (1) user@server:> reprounzip docker setup WN_ontology_measures_vs_embeddings.rpz docker_folder - We create a 'screen' session and run Reprounzip in background (2) user@server:> screen -S REPROUNZIP (3) user@screenconsole:> reprounzip docker run docker_folder - We detach from 'screen' before to close the server main console (4) user@screenconsole:> CTRL+a, d - We reattach to the screen console to check the completion of Reprounzip (5) user@server:> screen -r REPROUNZIP - We destroy the 'screen' console once finished Reprounzip execution (6) user@server:> screen -X -S REPROUNZIP quit

Table B.9: 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.

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 main paper [18]. In order to automate this post-processing, we provide the '*embeddings_vs_ontomeasures_final_tables.R*' R script file as shown in table B.6. In order to run the aforementioned script, you need to setup the well-known R statistical program[5] 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 the R-script file called 'embeddings_vs_ontomeasures_final_tables.R' contained in the folder shown in table B.6.

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

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

Table B.10: 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. You should obtain all raw output files listed in table B.7.

3. 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 main paper [18].
4. Select the menu option '*Edit->Run all*'. The final assembled tables will be saved in the output directories defined above, as detailed in table B.11.

Table B.11 shows the output files generated by the aforementioned R script from the raw output files listed in B.7, as well as their corresponding tables in our main paper [18].

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: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics. NAACL '09. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 19–27.

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

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

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

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

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

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

#	Post-processing output files saved at 'outputDir' directory	In paper [18]
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
7	table_Pearson_RelDatasets.csv	table 6 (full precision)
8	table_Pearson_RelDatasets_rounded.csv	table 6
9	table_Spearman_RelDatasets.csv	table 7 (full precision)
10	table_Spearman_RelDatasets_rounded.csv	table 7
13	table_joined_allEmbeddings_similarity.csv	table 8 (full precision)
14	table_joined_allEmbeddings_similarity_rounded.csv	table 8
15	table_joined_allEmbeddings_relatedness.csv	table 9 (full precision)
16	table_joined_allEmbeddings_relatedness_rounded.csv	table 9
17	table_pvalues_AttractReppel_allembeddings_similarity.csv	table A.1 (appendix A)
19	table_pvalues_Paragramws_allembeddings_relatedness.csv	table A.2 (appendix A)
19	table_AvgMeasures_Pearson_SimDatasets.csv	table A.3 (full precision)
19	table_AvgMeasures_Pearson_SimDatasets_rounded.csv	table A.3 (appendix A)
19	table_AvgMeasures_Spearman_SimDatasets.csv	table A.4 (full precision)
19	table_AvgMeasures_Spearman_SimDatasets_rounded.csv	table A.4 (appendix A)
19	table_AvgMeasures_Pearson_RelDatasets.csv	table A.5 (full precision)
19	table_AvgMeasures_Pearson_RelDatasets_rounded.csv	table A.5 (appendix A)
19	table_AvgMeasures_Spearman_RelDatasets.csv	table A.6 (full precision)
19	table_AvgMeasures_Spearman_RelDatasets_rounded.csv	table A.6 (appendix A)

Table B.11: Collection of processed output files generated by the execution of the 'embeddings_vs_ontomeasures_final_tables.R' script file onto the *outputDir* directory and its corresponding tables in our main paper [18] and appendix A.

[8] Gerz, D., Vulić, I., Hill, F., Reichart, R., Korhonen, A., Nov. 2016. SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP). Austin, Texas, pp. 2173–2182.

[9] Goikoetxea, J., Soroa, A., Agirre, E., Donostia, B. C., 2015. Random Walks and Neural Network Language Models on Knowledge Bases. In: HLT-NAACL. aclweb.org, pp. 1434–1439.

[10] Halawi, G., Dror, G., Gabrilovich, E., Koren, Y., 2012. Large-scale Learning of Word Relatedness with Constraints. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '12. ACM, New York, NY, USA, pp. 1406–1414.

[11] Hill, F., Reichart, R., Korhonen, A., Dec. 2015. SimLex-999: Evaluating

Semantic Models with (Genuine) Similarity Estimation. Computational Linguistics 41 (4), 665–695.

[12] Huang, E. H., Socher, R., Manning, C. D., Ng, A. Y., 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1. ACL '12. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 873–882.

[13] 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.

[14] Lastra-Díaz, J. J., García-Serrano, A., Nov. 2015. A novel family of IC-based similarity measures with a detailed experimental survey on WordNet. Engineering Applications of Artificial Intelligence Journal 46, 140–153.

[15] 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, NLP and IR Research Group. ETSI Informática. Universidad Nacional de Educación a Distancia (UNED).

[16] 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.

[17] 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.

[18] Lastra-Díaz, J. J., Goikoetxea, J., Hadj Taieb, M., García-Serrano, A., Ben Aouicha, M., Agirre, E., 2019. A reproducible survey on distributional and ontology-based methods for word similarity: simple combinations outperform the state of the art. Submitted for publication.

[19] Lastra-Díaz, J. J., Goikoetxea, J., Hadj Taieb, M. A., Agirre, E., García-Serrano, A., Ben Aouicha, M., 2018. Word similarity benchmarks of recent word embedding models and ontology-based semantic similarity measures. e-cienciaDatos, v1, http://dx.doi.org/10.21950/AQ1CVX.

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

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

[22] Mikolov, T., Chen, K., Corrado, G., Dean, J., May 2013. Efficient Estimation of Word Representations in Vector Space. In: Proceedigns of the International Conference on Learning Representations (ICLR). Scottsdale, AZ, USA.

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

[24] 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. Transactions of the Association for Computational Linguistics 5, 309–324.

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

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

[27] Pennington, J., Socher, R., Manning, C. D., 2014. Glove: Global vectors for word representation. Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014) 12, 1532–1543.

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

[29] Radinsky, K., Agichtein, E., Gabrilovich, E., Markovitch, S., Mar. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In: Proceedings of the 20th international conference on World wide web. ACM, pp. 337–346.

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

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

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

[33] Wieting, J., Bansal, M., Gimpel, K., Livescu, K., Roth, D., Jun. 2015. From Paraphrase Database to Compositional Paraphrase Model and Back.

[34] Yang, D., Powers, D. M., 2006. Verb similarity on the taxonomy of WordNet. In: The Third International WordNet Conference: GWC 2006. dspace2.flinders.edu.au.