An Integrated Platform Supporting Semantic Similarity Score Calculation and Reproducibility

Gaston K. Mazandu (gmazandu@gmail.com)
University of Cape Town
Kenneth Opap
University of Cape Town
Funmilayo Makinde
University of Cape Town
Victoria Nembaware
University of Cape Town
Francis Agamah
University of Cape Town
Christian Bope
University of Kinshasa
Emile R. Chimusa
University of Cape Town
Ambroise Wonkam, ambroise
University of Cape Town
Nicola J. Mulder
University of Cape Town

Research Article

Keywords: semantic similarity (SS), Python SS measure library (PySML)

Posted Date: August 18th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-806346/v1

Abstract

During the last decade, we witnessed an exponential rise of datasets from heterogeneous sources. Ontologies are playing an essential role in consistently describing domain concepts, data harmonization and integration to support large-scale integrative analysis and semantic interoperability in knowledge sharing. Several semantic similarity (SS) measures have been suggested to enable the integration of rich ontology structures into automated reasoning and inference. However, there is no tool that exhaustively implements these measures and existing tools are generally Gene Ontology specific, do not implement several models suggested in the WordNet context and are not equipped to properly deal with frequent ontology updates. We introduce a Python SS measure library (PySML), which tackles issues related to current SS tools, providing a portable and expandable tool to a broad computational audience. This empowers users to manipulate SS scores from several applications for any ontology version and file format. PySML is a flexible tool enabling the implementation of all existing semantic similarity models, resolving issues related to computation, reproducibility and re-usability of SS scores.

Introduction

In the current 'big data' era, we are witnessing an increase in large-scale multi-omics datasets generated from multiple sources. High-performance computing infrastructures have started to shift accordingly to enhance storing and computing power, fostering big data analytics. Integrative analyses generally yield high confidence inference with increased statistical power [1] . This elicits the need for data standards, such ontologies, to effectively harmonize and integrate these heterogeneous datasets and to provide an effective approach for making different meta-data FAIR: findable, interoperable, accessible and reusable[2]. Thus, in the past two decades, several ontologies and standards have been developed in different domains, including robotics and automation[3], biomedical research, e.g., the gene ontology (http://geneontology.org/)[4] and the human phenotype ontology (https://hpo.jax.org/app/)[5], to unambiguously describe domain terminology and concept associations in human- and machine-readable format. Online resources, including the NCBO BioPortal (https://bioportal.bioontology.org/)[6] and the Ontology Lookup Service (https://www.ebi.ac.uk/ols/)[7], have been designed to facilitate access to different ontologies in two main formats: web ontology language (OWL) and open biomedical ontology (OBO).

Semantic similarity measures allow the integration of knowledge contained in an ontology structure and contribute to the improvement of information retrieval. To optimally exploit the ontology structure, avoiding restrictions imposed by Boolean logic or exact concept match models, several semantic similarity (SS) measures [8–37] have been suggested. We refer the interested reader to[38], as well as to Supplementary File which provides an exhaustive review of these SS measures and existing tools. In the context of information retrieval, these measures enable the use of fuzzy expressions based on the degree of concept similarity, producing a satisfactory performance under uncertainty. These measures are still an active research area in various domains[39,40,41], e.g, biomedical, wordnet and artificial intelligence.

Web tools and software packages written in R, Python and Java were independently designed to facilitate the computation of SS scores [12,37,39–55] . However, existing tools are generally specific to the Gene Ontology (GO), context-dependent and consider only measures which were shown to perform well in GO[38,44]. Thus, most of these do not support edge-related measures, which may be relevant to other ontologies. Furthermore, existing tools are often context and organism dependent and only implement semantic similarity measures shown to perform well in a specific application, most of them use a defined and fixed version of the ontology, and some precompute term information content (IC) values. This indicates that existing SS tools are not well equipped to meet input requirements of current genome- and proteome-wide applications from high-throughput analysis[44], and existing tools cannot deal with frequent ontology updates.

Computing SS scores is still an issue due to the dearth of tools which consider related issues to produce consistent scores on demand and in real-time for use in related applications and for testing hypotheses. Here we implement Python SS measures library (PySML), a user-friendly and accessible package for retrieving SS scores for any ontology version and annotation datasets, resolving issues related to computation, reproducibility and reusability of SS scores. PySML provides an interface for developers to easily include source codes implementing their own SS models and facilitates the retrieval of any SS measures regardless of the ontology type and version. In addition, Python enables easy transitioning of codes between computers, rendering PySML portable and an effective framework for developing, assessing and testing existing and novel SS measures. This provides users with the freedom to choose most appropriate SS measures for specific applications.

Results

Retrieving SS scores

In a study by Mazandu et al [44] , it was observed that, for a fixed ontology (size), the running time increased linearly. However, between ontologies with varying sizes, for a fixed set of protein pairs, the running time increase was not linear considering GO cellular component (smallest size), GO molecular function (intermediate size) and GO biological process ontologies (highest size) with almost surely biological process ontology providing the worst running time. With this prior knowledge, it is valid to show how, in the worst case, i.e., using the largest ontology (GO biological process), the running time varies as reaching a limit speed event would be due to the user data input size and possibly to the number of operations induced. This model was also used in a study by Harispe et al[52], in which the ontology is fixed, and check how the library behaves in varying the user input size. Thus, using the existing largest ontology, which is the GO biological process ontology, we computed the average running time and extracted some SS scores produced in order to compare SML and PySML in terms of the running time performance and score integrity, i.e., consistency between models and resulted scores.

Checking score integrity

For checking score integrity, we used SS sores produced from the one megabyte input size consisting of approximately 60 975 protein-pairs. Results in Fig. 3a indicate that BMA scores are higher than scores derived from the SimGIC model in both PySML and SML. Nearly 100% of scores produced by the BMA model are higher than those from the SimGIC model, with the BMA mean scores of 0.30853 and 0.55727 for PySML and SML, respectively, as compared to SimGIC mean scores of 0.07888 for PySML and 0.12326 for SML. These observations also support results shown in Fig. 3b, suggesting that SS scores produced by SML are higher than those computed using PySML. To confirm this hypothesis, we performed a paired t-test, which revealed significance mean score differences: 0.24855 (95%CI:0.24696–0.25014) and 0.04440 (95%CI:0.04390–0.04489) with P-value < 6.7e−05 for BMA and SimGIC models, respectively, under the null hypothesis that there is no difference between scores produced by the PySML and SML tools.

Average running time performance

We assessed the average running time performance for PySML and SML using SS models described above. When computing different scores, we recorded the running time of each input size to assess the average running time for each library. Figure 4a shows how this expected running time varies with protein pair file sizes and, as expected the plot shows the linearity pattern increasing as concept pair input size increases, with SML yielding lower running time than PySML. In addition, we used most commonly used functional similarity measures for Semantic based measures (Avg, BMA, ABM, BMM, Max) and IC-based (SimGIC, SimDIC, SimUIC, SimCOU, SimCOT), summarized in Table 1 (see Supplementary File 1) and described in Supplementary File 2 (see Appendix 2), using Nunivers term similarity based on the GO universal IC values. The average running time outputs in Fig. 4b show the same pattern, as aforementioned.

Discussion

PySML and other similar tools

There have been numerous tools implemented for producing concepts and entity SS scores. Most of them are context dependent, restricted to GO with a specific fixed version and only implement SS measures shown to perform well in a specific application [44] . we refer the interested reader to[38] where these tools are described in terms of SS measures and input size that each tool supports. PySML is implemented in Python, one of the most efficient coding languages with various in-built libraries, providing fast prototyping capabilities which make it easy to learn. In addition, Python enables easy transitioning of code between computers, rendering PySML portable and an effective framework for developing, assessing and testing existing and novel SS measures. As such, PySML provides an interface for developers to easily include source code implementing their own SS models. This library facilitates the retrieval of any existing or new SS measures regardless of the ontology and 7 non-ontology based SS

measures. These non-ontology SS measures are used for other types of data structures, which can be translated into Boolean vector profiles, e.g., clinical records, gene expression and population- or individual-based single nucleotide polymorphism, protein-biological pathway profiles, etc.

Among existing tools dedicated to computing SS scores, the SML library [52] is most similar to PySML, SML is implemented in Java while PySML is implemented in Python. Though the two languages are among the best languages and available for most operating systems, Java, as a compiled language, offers good computational performance in terms of running time in comparison to interpreted languages, e.g., Python. However, Python programming language benefits outweigh Java running time performance and has become one of the most popular programming languages, if not the most popular programming language. This popularity is due to several factors, including high productivity as compared to other programming languages, like C, C + + and Java, simple programming syntax, code readability and English-like commands. As such, Python is more advantageous in terms of learning, considering its various in-built libraries and more appropriate for its expansion in many machine learning and artificial intelligence tasks. This suggests that the PySML library meets a high level of acceptability.

Assessing SS score integrity

We closely analyzed PySML scores in comparison to those produced by SML. For both tools, the BMA scores are higher than the SimGIC (Jaccard index) scores almost surely (Fig. 3a), as it would be expected based on their mathematical expressions. Comparing PySML to SML scores, SML produces higher scores than PySML (Fig. 3b). The high scores produced by SML is mainly caused by the contribution of the root of the ontology in SS computation as proteins sharing only the ontology root have a score greater than 0. This suggests that SML overestimates SS scores by considering the root as an informative ontology concept, which ultimately biases these scores[38], thus negatively impacting the performance of these SS models[38,54].

Retrieving, reproducing and reusing SS scores for any ontology in any application is still challenging [39,44,54] . This mainly due to the lack of a tool that exhaustively implements existing SS models and related assumptions to produce consistent scores on demand and in real-time for use in related applications and for testing hypotheses. PySML bridges this gap, providing an effective framework for developing, assessing and testing existing and novel SS measures with the possibility to compute a customized IC-based SS approach. This framework is practical and of immediate interest for the SS end-users and developers, helping in consistently retrieving and SS scores and easing comparisons of any existing and novel SS models.

Quantifying running time performance

As observed in Fig. 4, PySML average running time increases linearly as the input size increases. Comparing PySML average running time to SML (Fig. 4a), SML takes less time than PySML to produce SS scores, as expected. This is mainly due to the programming languages used as pointed out previously. SML benefits this feature from the Java programming language, as a compiled language, which is

expected to take less running time than PySML implemented in Python as an interpreted language. Considering the Python simple programming syntax, PySML is easy to understand compared to a Java-based software, and we anticipate a great acceptance of this library in the scientific and computational audience.

Taken together with average running time for computing different SS scores, PySML allows a large audience to benefit from its functionalities, effectively producing scores in realistic timeframes (Fig. 4b). It is worth mentioning that the Python performance issues are being overcome by introducing several libraries, such as scipy, numpy, etc. implemented in C and Fortran, well known compilers in terms of performance and speed, yielding performance boosts that can range from a few percent to several orders of magnitude, depending on the task at hand. This is an area of the PySML potential future expansion, which will be explored to optimize as much as possible the PySML running time.

Summary

PySML is a flexible, easy-to-use and expandable Python open library for handling SS measures for any ontology in any application with clear benefits when compared to similar solutions. It provides a large community interested in SS measures with an interface that eases SS measure implementation, testing, evaluation and comparison, enabling reproducibility and reusability of SS scores. Moreover, the PySML library adequately supports the implementation of new SS models, enabling end-users customized IC-based SS scores by providing the term-IC value map and freely choosing the SS models to be used for producing scores. Thus, PySML provides an effective platform for the replication and independent assessment of previously reported models and results.

Methods

Implementation of the PySML library

PySML is an extensible, effective, portable and expandable open library implemented in Python [56] version ≥ 2.7 and tested on a Linux operating system. It enables the retrieval and reproducibility of ontology concept IC and concept or entity (concept set) pairwise SS scores. PySML runs locally or on a high-performance computing cluster via an easy single command-line terminal on any computer or any operating system running Python and satisfying the PySML requirements (see Supplementary File 2, Appendix 1: Sect. 1 and Sect. 2). The only required argument is the ontology file in OWL or OBO format, the list of SS measures to retrieve or the application to process (see Fig. 1), and output results on the screen or in space separated value file according to user choice. In fact, PySML also performs concept or term enrichment analysis on a set of entities, clustering or classification of a set of entities based on SS scores and the identification of entities involved in a concept similar to a given concept target.

Different existing semantic similarity measures

As pointed out previously, several ontology semantic measures exist, including ontology concept IC and concept or entity (concept set) pairwise SS scores. The classification of different SS measures is shown

in Fig. 2 and the full description of these measures is provided in Supplementary File 2 (see Appendix 2). A systematic review by Mazandu et al[38] revealed a large collection of semantic similarity measures (refer to Supplementary File 2, Appendix 2, for more information) consisting of 9 IC approaches, yielding 930 ontology concept and 6574 entity pairwise SS measures, which can be deployed in specific applications. PySML is a repository of python modules for analyzing entity sets at the functional level based on their annotations using SS measures. It is a context-independent tool for implementing different SS measures shown in Fig. 2.

Symbols of different measures used in some existing tools and PySML are shown in Supplementary File 1 (see Table 1) and we refer readers to Supplementary File 2 (see Appendix 2) where complete descriptions and algebraic forms of all these measures are provided. Best performance often results from trading between accuracy and computational speed, PySML implements all these measures, except those which are known to be computationally unattractive with high disproportion between computational complexity and performance improvement, e.g., measures built on graph-based similarity measure (GRASM)[12,54]. However, PySML offers a platform that may be used to easily develop, test and assess any measures, so that users who are interested in these measures, for example, can process them within the platform. This makes PySML practical and easy to use, to the large community of SS users and to a broad computational audience, helping tool designers and experienced end-users, as well as non-programmers produce SS scores.

PySML parameter inputs and result outputs

In order to run PySML, the user invokes a specific module through Python alongside command arguments (see Supplementary File 2 for details, Appendix 1: Sect. 5 and Sect. 6 for details). The list of arguments is described in Supplementary File 2, Appendix 1: Sect. 5, with the model symbol argument in Table 1 for the SS model to be computed and model parameter symbol argument in Supplementary File 2 (see Table S1). Some other parameters change the operation of the program according to user preferences, for example, SS scores produced are presented in a table format, displayed on the screen or directed into an output file (space separated value format). It is worth noting that, for the ontology file, PySML allows user to provide the ontology file in OBO or OWL format or active Uniform Resource Locator (URL) to the ontology file in which case PySML retrieves the ontology online using the specified URL provided.

PySML enables the computation of more than one SS models in a single command line, which are retrieved sequentially. It stores these entity or term pairs in the output file alongside SS scores for each pair or term, which is calculated differently depending on the SS model. Upon running the PySML library, the output file, which is named after the SS measure, is created within the folder of the library by default, if no specific folder, which should contain the output file, has been provided. This output file format is as indicated above and containing all entity or term pairs and associated SS scores computed. Note that displaying output on the screen with argument (-s 1) is the default option. Different testing tasks were performed on a laptop model name Intel® Core™ i7–8665U CPU @ 1.90 GHz running x86_64 GNU/Linux.

For the testing process, the annotation file was extracted from the GOA UniProt dataset [57,58,59] at ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/UNIPROT/goa_uniprot_all.gaf.gz. We randomly selected pairs of GO biological process annotated proteins varying size up to one megabyte. We produced SS scores using SimGIC and BMA models with Lin pairwise term similarity based on the Seco IC values within the PySML and SML tools with the following arguments: (-t es -m bma:lin:seco simgic:seco -n biological_process -s 0) for PySML, (-t sm -profile GO -pm lin -ic seco -gm bma -aspect BP) in SML for the BMA model and (-t sm -profile GO -ic seco -gm gic -aspect BP) when the SimGIC model is run. For PySML running Avg, Max, BMA, ABM and BMM pairwise models, and IC-based SimGIC, SimDIC, SimUIC, SimCOU, SimCOT models with nunivers pairwise term similarity based on the GO-universal IC values, using (-t es -m m_arg -n biological_process -s 0) as argument for each model set as 'm_arg' a given model argument provided in Supplementary File 1 (Table 1).

Declarations

Data availability

All the results and data used in this study have been made available via the PySML github website at https://github.com/gkm-software-dev/post-analysis-tools located in the pysml-dev/tests folder (see PySMLRES2021A datasets) or via figshare at dx.doi.org/10.6084/m9.figshare.14599992.

Code availability

The PySML library for computing and reproducing semantic similarity scores is freely available and accessible at https://github.com/gkm-software-dev/post-analysis-tools in the pysml-dev folder and at http://web.cbio.uct.ac.za/ITGOM/post-analysis-tools/pysml-dev/, distributed under the GNU General Public License (GPL:https://www.gnu.org/licenses/gpl-3.0.en.html).

Acknowledgements

We would like to acknowledge all authors for their contributions in writing the paper, and developing the Python package. We extend our acknowledgements to everyone involved with free software, from the core developers to those who contributed to the documentation as well as those providing computational facilities, specifically the Centre for high performance computing (CHPC) at (https://www.chpc.ac.za). This study is supported by the National Institutes of Health (NIH), USA, Common Fund under H3ABioNet (U24HG006941) and SADaCC (U24HL135600), and partly by the South Africa NRF (132232/RA191016483424). The content is solely the responsibility of the authors and does not necessarily represent the official views of the funders.

Author Information

Affiliations

Division of Human Genetics, Department of Pathology, University of Cape Town, Health Sciences Campus, Anzio Rd, Observatory, 7925, South Africa

Gaston K. Mazandu, Victoria Nembaware, Francis Agamah, Emile R. Chimusa & Ambroise Wonkam

Computational Biology Division, Department of Integrative Biomedical Sciences, IDM, CIDRI-Africa WT Centre, University of Cape Town, Health Sciences Campus. Anzio Rd, Observatory, 7925, South Africa

Gaston K. Mazandu, Kenneth Opap, Funmilayo Makinde & Nicola J. Mulder

African Institute for Mathematical Sciences, 5-7 Melrose Road, Muizenberg, 7945, Cape Town, South Africa

Gaston K. Mazandu & Funmilayo Makinde

University of Kinshasa (UNIKIN), Faculty of Sciences, Department of Mathematics and Informatics, Kinshasa, Democratic Republic of Congo

Christian Bope

Contributions

G.K.M. developed and tested the library and performed the benchmarking analyses and comparisons with input from K.O, V.N. & E.R.C, drafted the manuscript with input from all co-authors who contributed to the final version of the manuscript.

Corresponding authors

Correspondence to be addressed to Gaston K. Mazandu.

Competing Interests

The authors declare no competing interests.

References

1. Mazandu, G. K. *et al.*. Designing data-driven learning algorithms: A necessity to ensure effective post-genomic medicine and biomedical research. In: Artificial Intelligence - Applications in Medicine and Biology. 5 Princes Gate Court, London, UK:IntechOpen Publisher,3–18(2019).
2. Wilkinson, M. D. *et al.* The FAIR Guiding Principles for scientific data management and stewardship. *Sci data*, **3**, 160018 (2016).
3. Prestes, E. *et al.* Towards a core ontology for robotics and automation. *Robotics and Autonomous Systems*, **61** (11), 1193–1204 (2013).

4. The Gene Ontology Consortium. The Gene Ontology resource: enriching a GOld mine. *Nucleic Acids Res*, **49** (D1), D325–D334 (2021).
5. Köhler, S. *et al.* The Human Phenotype Ontology in 2021. *Nucleic Acids Res*, **4**, D1207–D1217 (2021).
6. Martínez-Romero, M. *et al.* NCBO Ontology Recommender 2.0: an enhanced approach for biomedical ontology recommendation. J Biomed Semantics 8(1), 21(2017).
7. Côté, R. *et al.* The ontology lookup service: bigger and better. *Nucleic Acids Res*, **38**, W155–W160 (2010).
8. Lord, P. W. *et al.* Investigating semantic similarity measures across the gene ontology: the relationship between sequence and annotation., **19** (10), 1275–1283 (2003).
9. Sevilla, J. L. *et al.* Correlation between gene expression and GO semantic similarity. *IEEE/ACM Trans Comput Biol Bioinform*, **2** (4), 330–338 (2005).
10. Mazandu, G. K. & Mulder, N. J. DaGO-Fun: tool for Gene Ontology-based functional analysis using term information content measures. *BMC Bioinformatics*, **14**, 284 (2013).
11. Zhang, P. *et al.* Gene functional similarity search tool (GFSST). *BMC Bioinformatics*, **7**, 135 (2006).
12. Couto, F. M., Silva, M. J. & Coutinho, P. M. Measuring semantic similarity between Gene Ontology terms. *Data Knowledge Eng*, **61** (1), 137–152 (2007).
13. Schlicker, A. *et al.* A new measure for functional similarity of gene products based onGene Ontology. *BMC Bioinformatics*, **7**, 302 (2006).
14. Mazandu, G. K. & Mulder, N. J. Information content-based Gene Ontology semantic similarity approaches: toward a unified framework theory. BioMed Res Int 2013, 11 (2013).
15. Wang, J. Z. *et al.* A new method to measure the semantic similarity of GO terms., **23** (10), 1274–1281 (2007).
16. Pesquita, C. *et al.* Metrics for GO based protein semantic similarity: a systematic evaluation. *BMC Bioinformatics*, **9** (Suppl 5), S4 (2008).
17. Mistry, M. & Pavlidis, P. Gene Ontology term overlap as a measure of gene functional similarity. *BMC Bioinformatics*, **9**, 327 (2008).
18. del Pozo, A., Pazos, F. & Valencia, A. Defining functional distances over Gene Ontology. *BMC Bioinformatics*, **9**, 50 (2008).
19. Jain, S. & Bader, G. D. An improved method for scoring protein-protein interactions using semantic similarity within the gene ontology. *BMC Bioinformatics*, **11**, 562 (2010).
20. Mazandu, G. K. & Mulder, N. J. A topology-based metric for measuring term similarity in the Gene Ontology. Adv Bioinformatics 2012, 17 (2012).
21. Pesaranghader, A. *et al.* simDEF:definition-based semantic similarity measure of gene ontology terms for functional similarity analysis of genes., **32**, 1380–1387 (2016).
22. Peng, J. *et al.* Measuring semantic similarities by combining gene ontology annotations and gene co-function networks. *BMC Bioinformatics*, **6**, 44 (2015).

23. Bandyopadhyay, S. & et Mallick, K. A new path based hybrid measure for gene ontology similarity. *IEEE/ACM Trans Comput Biol Bioinform (TCBB)*, **11** (1), 116–127 (2016).
24. Schlicker, A., Lengauer, T. & Albrecht, M. Improving disease gene prioritization using the semantic similarity of Gene Ontology terms., **26** (18), i561–7 (2010).
25. Chabalier, J., Mosser, J. & Burgun, A. A transversal approach to predict gene product networks from ontology-based similarity. *BMC Bioinformatics*, **2**, 235 (2007).
26. Guo, X. *et al.* Assessing semantic similarity measures for the characterization of human regulatory pathways., **22** (8), 967–973 (2006).
27. Meng, J., Liu, D. & Luan, Y. Inferring plant microrna functional similarity using a weighted protein-protein interaction network. *BMC Bioinformatics*, **16**, 360 (2015).
28. Na, D., Son, H. & Gsponer, J. Categorizer: a tool that categorizes genes into user-defined biological groups based on semantic similarity. *BMC Genomics*, **15**, 1091 (2014).
29. Mazandu, G. K. & Mulder, N. J. The use of semantic similarity measures for integrating heterogeneous Gene Ontology annotation pipelines. *Front Genet*, **5**, 264 (2014).
30. Pesquita, C. *et al.* Semantic similarity in biomedical ontologies. *PLoS Comput Biol*, **5** (7), e1000443 (2009).
31. Mazandu, G. K. & Mulder, N. J. Information content-based gene ontology functional similarity measures: which one to use for a given biological data type? *PLoS ONE*, **9** (12), (2014). e113859
32. Seco, N., Veale, T. & Hayes, J. An intrinsic information content metric for semantic similarity in wordnet. In:16th European Conference on Artificial Intelligence, ECAI 2004, IOS Press,Valencia, Spain, 1089–90(2004).
33. Sanchez, D., Batet, M. & Isern, D. Ontology-based information content computation. *Knowledge-Based Syst*, **24**, 297–303 (2011).
34. Zhou, Z., Wang, Y. & Gu, J. A new model of information content for semantic similarity in WordNet. In:Second International Conference on Future Generation Communication and Networking Symposia, FGCNS2008, IEEE Computer, Washington, USA, 3, 85–9(2008).
35. Seddiqui, M. H. & Aono, M. Metric of intrinsic information content for measuring semantic similarity in an ontology. In:Proceedings of 7th Asia-Pacific Conference on Conceptual Modeling, Brisbane, Australia, 110, 89–96(2010).
36. Meng, L., Gu, J. & Zhou, Z. A new model of information content based on concept's topology for measuring semantic similarity in WordNet. *Int J Grid Distrib Comput*, **5** (3), 81–94 (2012).
37. Jeong, J. C. & Chen, X. W. A new semantic functional similarity over Gene Ontology. *IEEE/ACM Trans Comput Biol Bioinform*, **12** (2), 322–334 (2014).
38. Mazandu, G. K., Chimusa, E. R. & Mulder, N. J. Gene Ontology semantic similarity tools: survey on features and challenges for biological knowledge discovery. *Brief Bioinform*, **18** (5), 886–901 (2017).
39. Lara-Clares, A., Lastra-Díaz, J. J. & Garcia-Serrano, A. Protocol for a reproducible experimental survey on biomedical sentence similarity. *PLoS ONE*, **16** (3), (2021). e0248663

40. Le, D. H. UFO: A tool for unifying biomedical ontology-based semantic similarity calculation, enrichment analysis and visualization. *PLoS ONE*, **15** (7), e0235670 (2020).
41. Nguyen, Q. H. & Le, D. H. Similarity Calculation, Enrichment Analysis, and Ontology Visualization of Biomedical Ontologies using UFO. *Curr Protoc*, **1** (4), (2021). e115
42. Fröhlich, H., Speer, N., Poustka, A. & Beißbarth, T. GOSim–an R-package for computation of information theoretic GO similarities between terms and gene products. *BMC Bioinformatics*, **8**, 166 (2007).
43. Du, Z. *et al.* G-SESAME: web tools for GO-term-based gene similarity analysis and knowledge discovery. *Nucleic Acids Res*, **37** (2), D345–9 (2009).
44. Mazandu, G. K. *et al.* A-DaGO-Fun: an adaptable Gene Ontology semantic similarity based func-tional analysis tool., **32**, 477–479 (2016).
45. Lastra-Díaz, J. J. *et al.* HESML: a scalable ontology-based semantic similarity measures library with a set of reproducible experiments and a replication dataset. *Information Systems*, **66**, 97–118 (2017).
46. Faria, D. *et al.* ProteInOn: a web tool forprotein semantic similarity. https://webpages.ciencias.ulisboa.pt/~fjcouto/files/report%20dfaria-tr2007.pdf (2007).
47. Ovaska, K., Laakso, M. & Hautaniemi, S. Fast gene ontology based clustering for microarray experiments. *BioData Min*, **1**, 11 (2008).
48. Caniza, H. *et al.* GOssTo: a user-friendly stand-alone and web tool for calculating semantic similarities on the Gene Ontology., **30**, 2235–2236 (2014).
49. Schlicker, A. & Albrecht, M. FunSimMat: a comprehensive func-tional similarity database. *Nucleic Acids Res*, **36**, D434–9 (2008).
50. Martin, D. *et al.* GOToolBox: functional analysis of gene data sets based on Gene Ontology. *Genome Biol*, **5** (12), 1901–1908 (2004).
51. Yu, G. *et al.* GOSemSim: an R package for measuring semantic similarity among GO terms and gene products., **26** (7), 976–978 (2010).
52. Harispe, S. *et al.* The semantic measures library and toolkit: fast computation of semantic similarity and relatedness using biomedical ontologies., **30**, 740–742 (2014).
53. Gentleman, R. & Visualizing and distances using GO. http://bioconductor.org/packages/2.6/bioc/vignettes/Gostats/inst/doc/Govis.pdf (2005).
54. Bible, P. W. *et al.* The effects of shared information on semantic calculations in the gene ontology. *Computational and Structural Biotechnology Journal*, **15**, 195–211 (2017).
55. McInnes, B. T., Pedersen, T. & Pakhomov, S. V. UMLS-Interface and UMLS-Similarity: open source software for measuring paths and semantic similarity. AMIA Annu Symp Proc 2009, 431-5 (2009).
56. Python Software Foundation. Python Language Reference, version 2.7. Available at http://www.python.org (2021).

57. Barrell, D. *et al.* The GOA database in 2009-an integrated Gene Ontology Annotation resource. *Nucleic Acids Res*, **37** (Database issue), D396–403 (2009).
58. Mutowo-Meullenet, P. *et al.* Use of Gene Ontology Annotation to understand the peroxisome proteome in humans. Database (Oxford) 2013, bas062 (2013).
59. Huntley, R. P. *et al.* The GOA database: gene Ontology annotation updates for 2015. *Nucleic Acids Res*, **43** (Database issue), D1057–63 (2015).

Figures

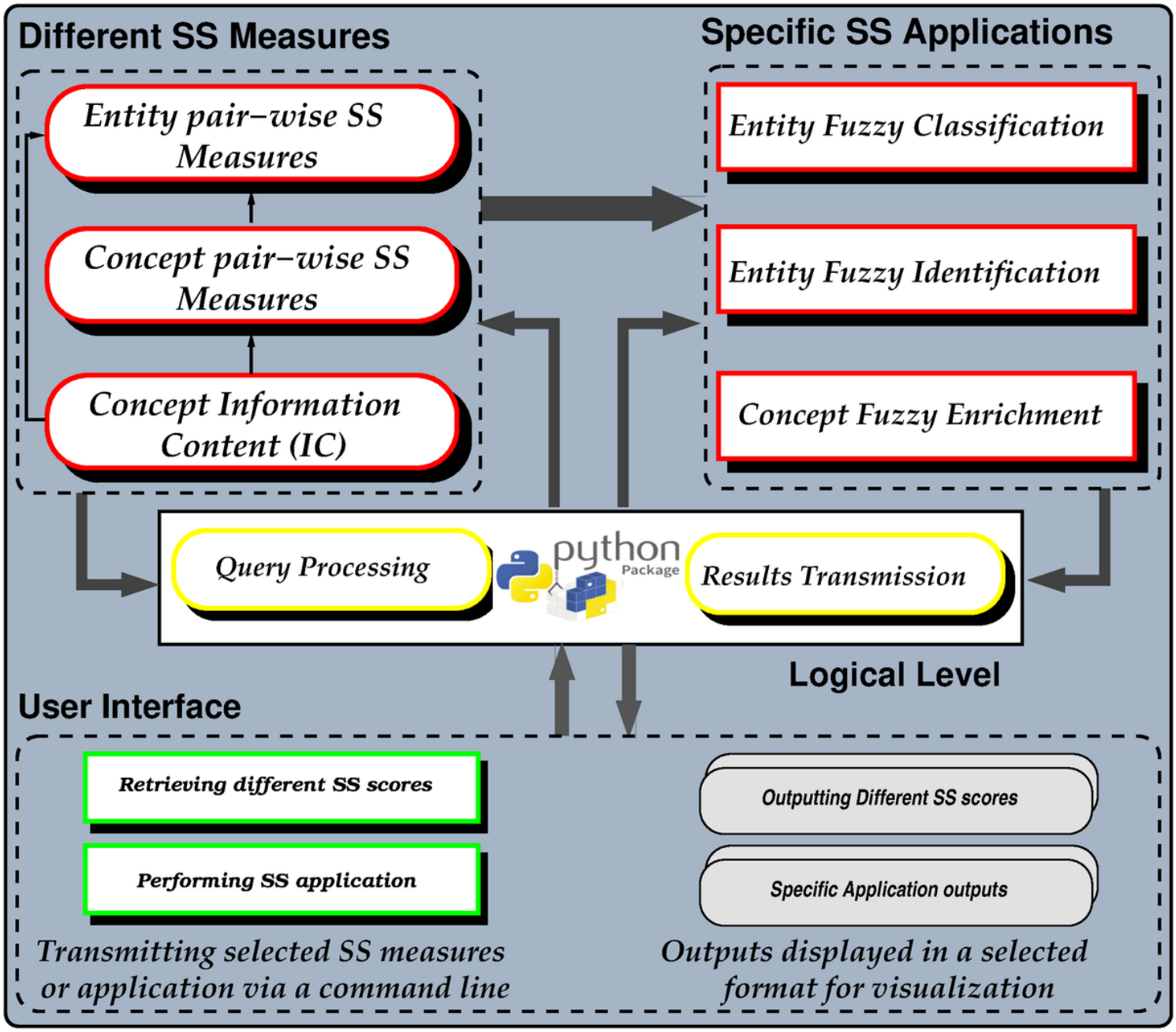

Figure 1

Overall workflow of the PySML tool. The scheme has two main steps: User interface and input processing. Inputs are parsed via a single command-line terminal and the query is run, producing SS scores/application result in the user format choice.

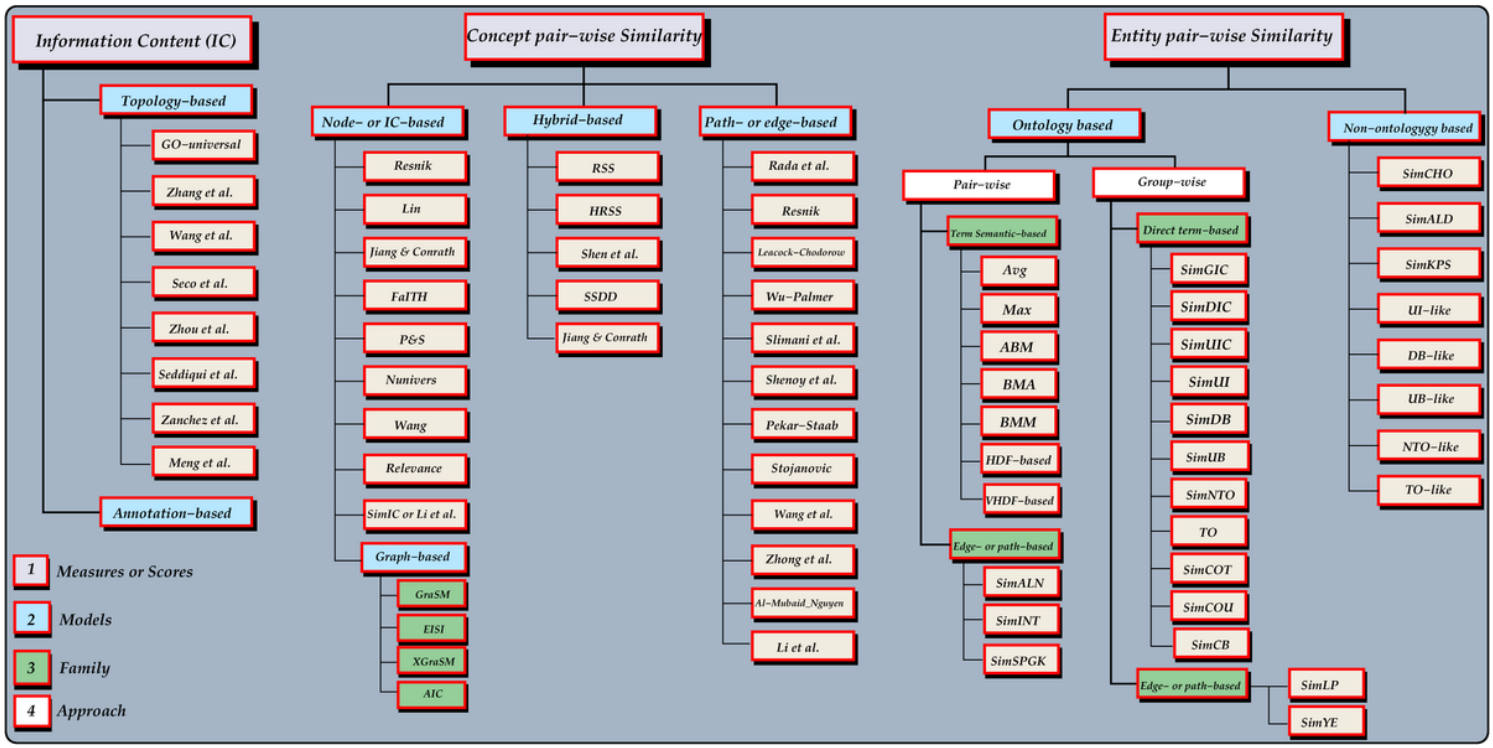

Figure 2

Flowchart adapted from38, classifying existing SS measures. Flowchart of different measures classified according to their conception. IC values are produced using topology or annotation-based models. Using term IC values or path length/distance (shortest path length) between two concepts (nodes) and depth of concepts in the structure lead to different concept SS and entity SS models based on the structure of the ontology. There also exist entity SS measures, which do not take into account the structure of the ontology, referred to as non-ontology structure-based measures.

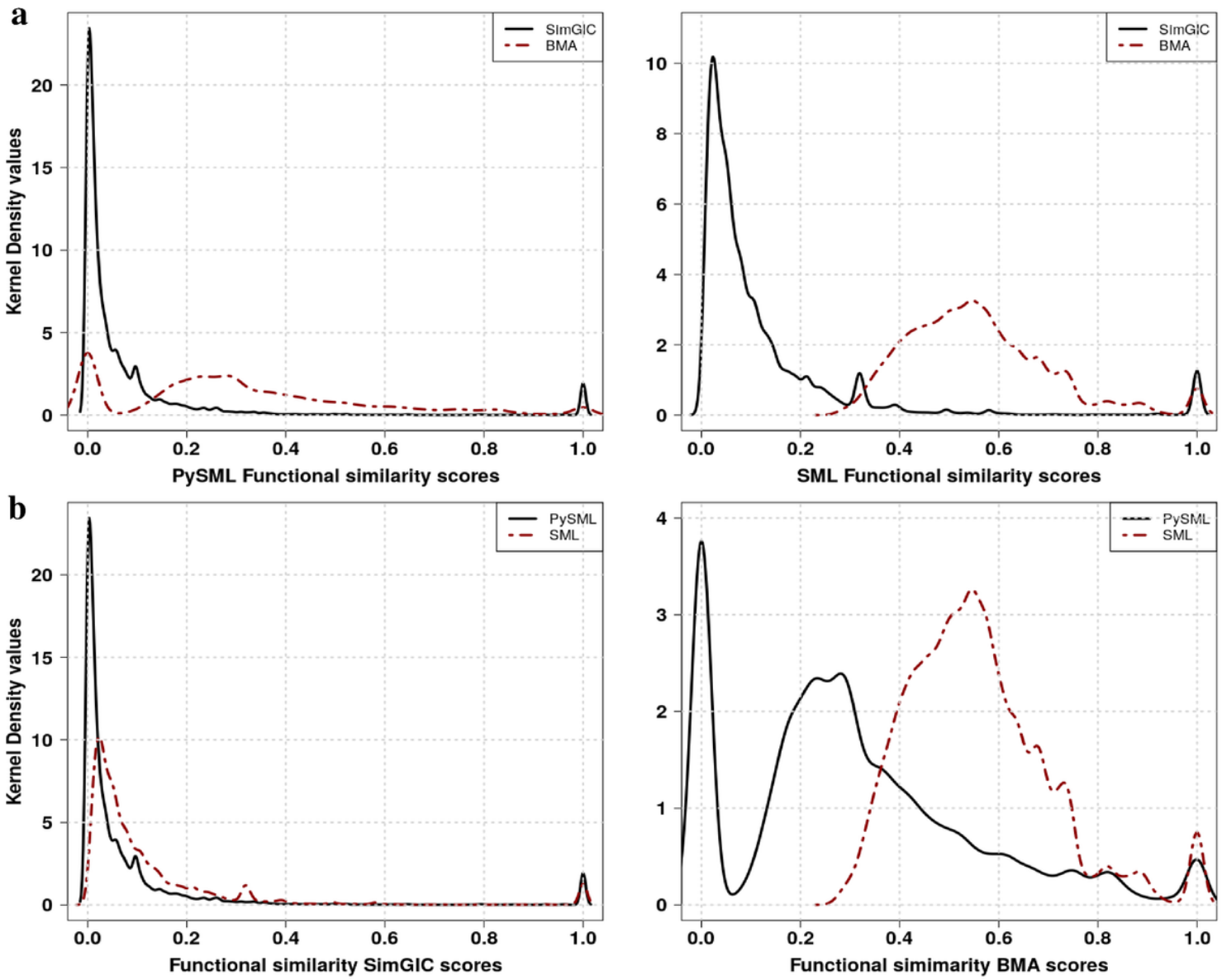

Figure 3

Distribution of BMA and SimGIC scores produced by PySML and SML tools. (a) Comparing scores computed using pairwise based BMA and IC-based SimGIC models within each tool. (b) Comparing scores computed using pairwise based BMA and IC-based SimGIC models between PySML and SML tools.

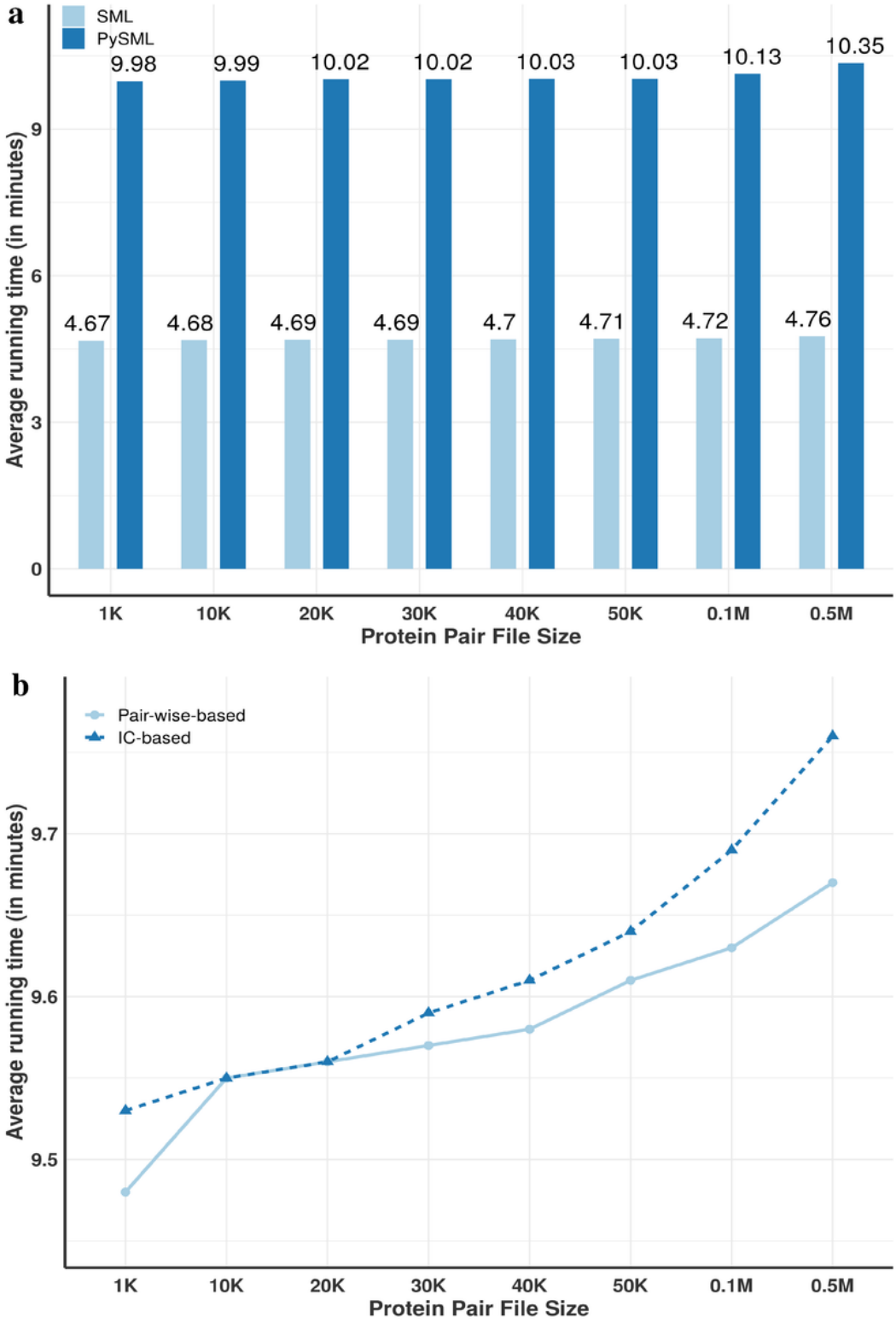

Figure 4

Running time performance of PySML and SML based on BMA and SimGIC models. (a) Entity (protein) pairwise SS score retrieval versus file size in kilobytes (K) or megabytes (M), approximating linear time complexity, with average running time of BMA and SimGIC models with Lin pairwise term similarity based on the Seco IC values. (b) Expected running time of PySML using Avg, Max, BMA, ABM and BMM models,

and that of IC-based SimGIC, SimDIC, SimUIC, SimCOU, SimCOT models with nunivers pairwise term similarity based on the GO-universal IC values.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- Table1.docx
- PySMLManual2021.pdf