SoftwareX 35 (2026) 102954

Contents lists available at ScienceDirect

SoftwareX

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

CSO Classifier 4.0: Software for efficient ontology-based topic annotation in computer science

Faisal Ramzan [a], Angelo Salatino [b], Francesco Osborne [b], Diego Reforgiato Recupero [a,*], Enrico Motta [b]

[a] Department of Mathematics and Computer Science (DMI), University of Cagliari, Via Ospedale 72, Cagliari, 09124, Italy
[b] Knowledge Media Institute (KMi), The Open University, Walton Hall, Milton Keynes, MK7 6AA, United Kingdom

ARTICLE INFO

Keywords:
Ontology-based topic annotation
Topic extraction
Scholarly metadata
Ontology-based classification
Word embeddings
Scalable preprocessing

ABSTRACT

CSO Classifier 4.0 is an open-source software framework for automatically annotating scholarly documents with research topics from the Computer Science Ontology. The software maps document metadata, including titles, abstracts, and keywords, to ontology concepts through syntactic matching, semantic expansion, and ontology-aware post-processing. This new version 4.0 introduces an updated semantic backbone trained on recent scholarly metadata, soft classification with topic weighting, domain-aware filtering, compatibility with modern Python environments, and improved metadata generation. The package supports scalable, reproducible, and interpretable topic annotation for digital libraries, knowledge graph construction, recommender systems, trend analysis, and science-of-science studies.

Metadata

Code metadata.

Nr	Code metadata description	Metadata
C1	Current code version	*4.0.0*
C2	Permanent link to code/repository used for this code version	*https://github.com/angelosalatino/cso-classifier*
C3	Legal code license	*Apache 2.0 License*
C4	Code versioning system used	*Git*
C5	Software code languages, tools and services used	*Python; PyPI; GitHub; Word2Vec-based semantic resources; Computer Science Ontology*
C6	Compilation requirements, operating environments and dependencies	*Python 3.11/3.12; platform independent; dependencies specified in* `requirements.txt`
C7	If available, link to developer documentation/manual	*https://github.com/angelosalatino/cso-classifier*
C8	Support email for questions	*kmi-cso@open.ac.uk*

1. Motivation and significance

The rapid growth of scholarly publications has significantly increased the difficulty of organising, searching, and analysing scientific knowledge at scale. Modern digital libraries have evolved beyond traditional metadata-based retrieval systems to support semantic enrichment and AI-driven content analysis workflows [1]. As a result, research infrastructures require topic metadata that is both interpretable and machine-actionable, enabling reliable indexing, recommendation, trend analysis, and large-scale analytics across scholarly artefacts such as papers, patents, and technical reports.

Automatically identifying meaningful research topics from scholarly documents remains challenging. Traditional topic modelling approaches infer latent themes directly from text. Probabilistic models such as Latent Dirichlet Allocation (LDA) provide a principled unsupervised framework, but their outputs are often difficult to interpret, compare across collections, and align with fine-grained research areas without manual intervention [2–4]. Extensions of these models incorporate contextual

* Corresponding author.
Email address: diego.reforgiato@unica.it (D. Reforgiato Recupero).

https://doi.org/10.1016/j.softx.2026.102954
Received 13 May 2026; Received in revised form 27 July 2026; Accepted 12 August 2026
Available online 17 August 2026
2352-7110/© 2026 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

signals such as authorship information [5]. Supervised text categorisation methods assign documents to predefined labels using annotated training data [6,7]; however, such datasets are difficult to obtain for fine-grained scholarly topics because of the high number of labels and substantial class imbalance.

Ontology-driven methods address these limitations by grounding annotations in structured vocabularies. In the biomedical domain, automatic Medical Subject Headings (MeSH) indexing has transitioned from hybrid rule-based systems such as the Medical Text Indexer [8,9] to neural approaches including DeepMeSH, AttentionMeSH, and BERTMeSH [10–13], demonstrating the immense utility of controlled vocabularies for large-scale indexing. Parallel to this, within the computing landscape, the Computer Science Ontology (CSO) provides a large-scale representation of research topics and semantic relations for consistent, human-readable annotations [14,15], operationalised primarily through CSO Classifier to execute automatic document annotations [16].

Large Language Models (LLMs) and hybrid neural approaches have recently been explored for scholarly knowledge mining, including question answering, summarisation, reasoning over scientific articles, PDF understanding, and topic extraction [17–23]. However, their use in high-throughput scholarly analytics remains constrained by computational cost, latency, operational complexity, and reproducibility concerns [24,25]. Overall, scalable topic annotation still requires a practical balance between interpretability, semantic coverage, stability, and efficiency.

Building on Computer Science Ontology, CSO Classifier was introduced as an ontology-based classification framework that maps scholarly metadata, such as titles, abstracts, and keywords, to research concepts and expands them through semantic relatedness [16,26]. It achieves this via controlled-vocabulary annotation using lightweight Word2Vec-style embeddings, offering a practical trade-off between scalability and semantic generalisation whilst preserving interpretable, grounded labels.

In this software paper, we present `CSO Classifier 4.0`, the latest version of this ontology-driven software framework, designed for contemporary scholarly corpora and large-scale analytics pipelines.[1] Users provide document metadata as input, and the software maps it to CSO concepts through syntactic matching, semantic expansion, and ontology-aware post-processing. The output consists of ontology-aligned topic annotations and, when required, topic weights that can be directly used in downstream workflows such as semantic indexing, recommendation, knowledge graph construction, trend analysis, and scholarly analytics.

CSO Classifier 4.0 builds upon the architecture introduced in version 3.0 while improving both the semantic model and software maintainability. It incorporates an updated semantic backbone trained on larger and more recent scholarly metadata, enabling better coverage of contemporary research terminology. It also introduces soft classification with topic weighting, domain-aware filtering, support for modern Python environments, and improved interoperability through standardised metadata generation.

In this paper, we primarily focus on the implementation, functionality, and evolution of CSO Classifier 4.0. We refer the readers to Salatino et al. [16,26] for a detailed description of the underlying methodological approach of the original CSO Classifier. Therefore, our main contributions are:

- We introduce CSO Classifier 4.0, an updated open-source software package for ontology-guided topic annotation of scholarly documents based on the Computer Science Ontology.
- We enhance the semantic model and extend the functionality of the classifier by leveraging updated large-scale scholarly metadata and introducing soft classification with topic weighting and domain-specific filtering.

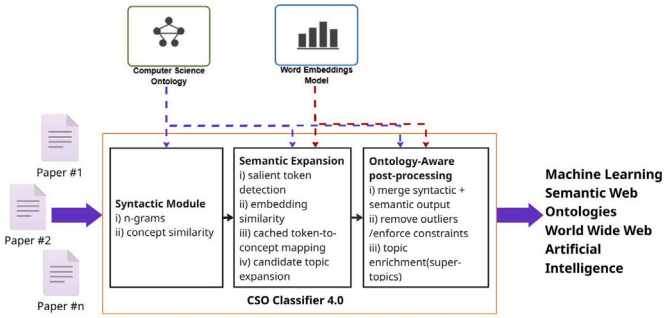

Fig. 1. Workflow of CSO Classifier.

- We provide the complete dataset construction and retraining pipeline to support full ecosystem reproducibility and custom community deployments.

The remainder of this paper is organised as follows. Section 2 describes the software architecture and main functionalities of the classifier. Section 3 describes the classifier in practice with an illustrative example. Section 4 discusses the scientific and practical impact of the software. Finally, Section 5 concludes the paper and outlines future developments.

2. CSO classifier 4.0

In this section, we describe the architecture, and the new functionalities of the classifier, alongside its evaluation.

2.1. Software architecture

Fig. 1 summarises the overall architecture. At inference time, the classifier takes as input a document's title, abstract, and optionally keywords, and returns a set of CSO topics. The system operates through three main components: (i) a *syntactic module* that identifies concepts explicitly mentioned in the text, (ii) a *semantic module* that infers related topics using distributional semantics, and (iii) an *ontology-aware post-processing module* that consolidates and refines the results.

The *syntactic module* identifies topics explicitly mentioned in the input text by mapping candidate n-grams to CSO concepts. After removing English stop words, it generates unigrams, bigrams, and trigrams and compares each candidate with CSO topic labels using normalized Levenshtein string similarity, retaining the matched topics with a similarity score above 0.9. For computing the Levenshtein similarity, we employ RapidFuzz.[2]

The *semantic module* identifies additional topics that are not explicitly mentioned in the input text. It extracts relevant tokens using part-of-speech tagging and associates them with semantically related CSO concepts using a precomputed token-to-concept cache derived from a bespoke phrase-aware Word2Vec embeddings model. During inference, the cache enables efficient retrieval of candidate CSO concepts without repeatedly querying the complete embedding model. Details of the embedding training and cache-generation process are provided in Section 2.3.

The *ontology-aware post-processing module* integrates the outputs of the syntactic and semantic components and evaluates the consistency of the resulting candidate topics. Candidate concepts are compared with the remaining topic set using pairwise semantic similarities derived from the embedding space and structural relationships encoded in CSO. Topics that are weakly connected to the dominant topic set are removed as outliers, while closely related concepts are retained. The module then

[1] CSO Classifier 4.0—https://pypi.org/project/cso-classifier/4.0.0/.

[2] RapidFuzz—https://rapidfuzz.github.io/RapidFuzz/.

enriches the final result by adding relevant higher-level CSO concepts, producing a coherent and hierarchically structured topic representation.

2.2. Software functionalities

Compared with its previous version, CSO Classifier 4.0 improves both the semantic model and the software infrastructure. It adopts an updated semantic backbone trained on a substantially larger and more recent corpus of scholarly metadata (approximately 132 million papers), including publications up to 2025, thereby improving coverage of emerging research topics.

Further, the classifier introduces new functionalities for more flexible topic annotation. When `get_weights = True`, the classifier returns a weight for each identified topic. For syntactic topics, the weight corresponds to the normalized Levenshtein similarity between the extracted expression and the matched CSO label. For semantic topics, the relevance score considers both the number of times a concept is identified and the number of distinct input n-grams that trigger it.

The `filter_by` parameter can further restrict the output to topics within user-specified CSO branches. Finally, this new version has been written to improve compatibility with Python 3.11 and 3.12, features expanded documentation, and supports Croissant-compliant metadata generation.

2.3. Semantic resource generation

CSO Classifier 4.0 relies on semantic resources generated offline and subsequently used during inference. To accommodate the continuous evolution of scholarly language, we developed a companion dataset construction and retraining pipeline, which we have made openly available in a public repository.[3] This pipeline retrieves the latest CSO release[4] and constructs an English-language training corpus from Semantic Scholar[5] titles and abstracts. During preprocessing, multiword CSO concepts are represented as underscore-connected tokens, and frequent bigrams and trigrams are preserved as single semantic units. A custom Word2Vec model [4] is then trained using Gensim 4.3.3,[6] with the embeddings learned from the distributional contexts of terms and phrases in the corpus.

To generate the token-to-concept cache, the pipeline retrieves semantically related terms from the embedding space for each token in the model, retaining candidates with a cosine similarity of at least 0.70. It then computes the normalised Levenshtein similarity between these candidates and all CSO labels, keeping only the matches with a threshold above 0.90. The resulting associations are stored and reused during inference.

The full retraining pipeline was executed on a Dell PowerEdge R730xd server equipped with two Intel Xeon E5-2630 v3 processors (16 physical CPU cores in total) and 128 GiB of RAM, requiring approximately two days to complete. To improve memory efficiency, multiprocessing was used, and the dataset was divided into four equally sized partitions. No GPU was required for the Word2Vec training stage. These requirements apply only to regenerating the semantic resources; standard users can use the precomputed resources distributed with CSO Classifier 4.0.

2.4. Evaluation and computational benchmarking

We evaluated CSO Classifier 4.0 through a regression testing approach on 10,000 matched papers against CSO Classifier 3.0, treating the latter's outputs as a stable reference. The comparison uses Jaccard similarity over predicted semantic topic sets. The results show strong agreement between the two versions, with a mean Jaccard similarity of 0.768 and a median of 0.800. CSO Classifier 4.0 also produces broader annotations, increasing the average number of topics per paper from 6.890 to 7.376 and the number of unique topics from 6811 to 7200.

This evaluation measures continuity and changes in topic coverage relative to CSO Classifier 3.0, whose high classification quality was already validated against human experts in the original publication [26]. Because the standard Jaccard coefficient was calculated over exact topic labels without accounting for hierarchical relationships, these results represent a strict measure of version-to-version continuity. Using a large-scale sample of 10,000 papers ensures that the transition to the new retraining pipeline preserves the core behavior of the validated framework while successfully expanding topic coverage.

Furthermore, we assessed the computational efficiency of CSO Classifier 4.0, by comparing it against its predecessor and two additional baselines: a BERT-based Named Entity Recognition model [27] and Mistral-7B-Instruct-v0.3. This evaluation was conducted on the same ordered sample of 100 papers, using only titles and abstracts as input.

To reduce the influence of transient variations in execution time, memory utilisation, and energy estimation, the inference benchmark on the same ordered sample of 100 papers was repeated 100 times for each system. The models were not retrained during these repetitions. Table 1 reports the comparison based on an array of indicators, and values are expressed as mean ± standard deviation. All systems were evaluated on the same physical workstation, equipped with an Intel Core i7-12700H processor, 14 physical cores, 20 logical processors, and 16 GB of RAM, using the same document inputs.

Operational energy consumption was estimated using the CodeCarbon `OfflineEmissionsTracker`.[7] To ensure comparability, all CO_2-equivalent values were recalculated from the estimated energy consumption using a common carbon-intensity factor of 330.718 gCO_2/kWh.

Although the inference benchmark was repeated 100 times, it contained only 100 unique papers. Moreover, CodeCarbon provides software-based energy estimates rather than direct measurements from an external power meter. The reported values should therefore be interpreted as workload-specific operational estimates and not as a general environmental ranking or lifecycle assessment of the evaluated systems.

The principal like-for-like comparison is between CSO Classifier 3.0 and CSO Classifier 4.0, since both systems perform ontology-based topic classification. CSO Classifier 4.0 reduced the average inference time from 48.81 ± 6.90s to 39.65 ± 3.93s and increased throughput from 125.52 ± 19.48 to 152.42 ± 12.04 papers per minute. It also reduced the estimated energy consumption from 0.71 ± 0.10Wh to 0.30 ± 0.05Wh and the normalized CO_2-equivalent emissions from 0.23 ± 0.03g to 0.10 ± 0.01g.

CSO Classifier 4.0 produced slightly broader ontology coverage, with the average number of topics per paper increasing from 6.02 to 6.27 and the number of unique topics increasing from 484 to 504. The principal trade-off was memory consumption, which increased from 1665.35 ± 21.47 MB to 3124.44 ± 12.14 MB.

BERT NER and Mistral-7B are included only as task-oriented computational baselines, not as direct alternatives for ontology classification. Their entity- and prompt-based outputs are not directly comparable with ontology-linked CSO topic annotations. Moreover, because they were evaluated using model-specific benchmark implementations, their energy and CO_2e values are reported only as indicative computational context rather than as strictly equivalent inference measurements.

[3] Dataset Construction and Retraining Pipeline—https://github.com/faisalramzan3725/w2v_model_for_cso_classifier/.

[4] CSO Concepts—https://cso.kmi.open.ac.uk/downloads.

[5] Semantic Scholar—https://www.semanticscholar.org/.

[6] Gensim—https://radimrehurek.com/gensim/.

[7] CodeCarbon—https://docs.codecarbon.io/.

Table 1
Computational comparison on the same benchmark of 100 scientific papers. Values are reported as mean ± standard deviation over 100 inference runs.

Metric	CSO 3.0	CSO 4.0	BERT NER [27]	Mistral-7B `Q4_K_M`
Primary task	Ontology classification	Ontology classification	Entity extraction	Prompt-based extraction
Ontology-linked output	Yes	Yes	No	No
Inference time (s)	48.81 ± 6.90	39.65 ± 3.93	20.46 ± 5.36	3279.24 ± 460.01
Time per paper (s)	0.48 ± 0.06	0.39 ± 0.03	0.20 ± 0.05	32.79 ± 4.60
Throughput (papers/min)	125.52 ± 19.48	152.42 ± 12.04	309.51 ± 65.70	1.85 ± 0.18
Incremental peak memory (MB)	1665.35 ± 21.47	3124.44 ± 12.14	1138.44 ± 7.60	6418.95 ± 383.96
Estimated energy consumption (Wh)	0.71 ± 0.10	0.30 ± 0.05	0.42 ± 0.22	29.22 ± 22.68
Normalized estimated CO_2-equivalent emissions (g)	0.23 ± 0.03	0.10 ± 0.01	0.14 ± 0.07	9.66 ± 7.50
Average topics/entities per paper	6.02	6.27	50.72	4.96
Unique topics/entities	484	504	3296	493

3. CSO classifier in practice

3.1. Demonstration example and classifier output

To demonstrate the main functions of CSO Classifier 4.0, we consider a sample document [28] with the following metadata:

Title: "A deep learning approach for medical image classification using XAI and convolutional neural networks"

Abstract: "The classification of medical images is a crucial task for medical diagnosis and treatment, as it provides vital information about the patient's condition and aids in making medical decisions quickly. The justification for using deep learning for the classification of medical images…"

Keywords: Convolutional neural networks, Explainable artificial intelligence, Local interpretable model-agnostic explanations, Medical image classification, SHapley Additive exPlanations

Running the code in Listing 1 produces a structured dictionary with the fields `syntactic`, `semantic`, `union`, `enhanced`, `explanation`, `syntactic_weights`, and `semantic_weights`. To keep the manuscript concise, the complete JSON output is provided in the companion document *CSO Classifier Output*.[8]

The syntactic module first extracts n-grams from the input text and matches them against CSO labels. In this example, it identifies explicit concepts such as *deep learning*, *convolutional neural networks*, *medical images*, *image classification*, *image segmentation*, *artificial intelligence*, and *parallel computing*. These matches provide high-precision topic candidates directly grounded in the document's terminology.

The semantic module then expands the candidate set by leveraging distributional similarity. Using the precomputed token-to-concept cache, the system associates salient tokens, like *classification*, *neural*, *image*, and *parallel*, with related CSO concepts, and introduces additional relevant topics, including *machine learning* and *computer vision*, even though they are not explicitly mentioned in the input text.

Finally, the ontology-aware post-processing module merges syntactic and semantic candidates, removes outliers, and enriches the results using the CSO structure. In this example, higher-level or related concepts, including *image processing* and *graphics processing unit*, are added, resulting in a coherent, hierarchical topic profile.

The final output is the following set of CSO topics:

```
Deep Learning, Convolutional Neural Networks, Medical
Images, Image Classification, Image Segmentation,
Artificial Intelligence, Machine Learning, Computer
Vision, Image Processing, Parallel Computing, Graphics
Processing Unit
```

Overall, this example illustrates how CSO Classifier 4.0 combines explicit concept detection, semantic expansion, and ontology-aware enrichment to generate interpretable, ontology-aligned topic annotations from standard scholarly metadata.

[8] CSO Classifier Sample Output—https://doi.org/10.5281/zenodo.21442400.

```python
from cso_classifier import CSOClassifier
import json

paper = {
    "title": "A deep learning approach for medical image
        classification using XAI and convolutional neural networks",
    "abstract": "The classification of medical images is a crucial
        task for medical diagnosis and treatment, as it provides...",
    "keywords": "Convolutional neural networks, Explainable
        artificial intelligence, Local interpretable model-agnostic
        explanations, Medical image classification, SHapley Additive
        exPlanations"
}

cc = CSOClassifier(explanation=True, get_weights=True)

result = cc.run(paper)

print(result)

with open("output.json", "w") as outfile:
    json.dump(result, outfile, indent=4)
```

Listing 1. Programmatic use of CSO Classifier 4.0 and JSON export.

3.2. Programmatic usage and toolchain integration

CSO Classifier can be installed in any virtual environment running Python 3.11 or later using `pip install cso-classifier` and can be readily embedded in external Python-based workflows as a metadata-enrichment component. The classifier accepts a structured record comprising a paper title, abstract, and, where available, author keywords. This input format can represent a single publication or a batch of multiple papers. These outputs can be exported as JSON and reused in external software pipelines, including digital libraries, search indexes, scholarly knowledge graphs, recommender systems, and large-scale science-of-science analytics platforms.

Listing 1 provides a minimal example in which the classifier is applied to the sample document discussed in Section 3.1. The code instantiates the classifier, processes the input metadata, prints the resulting annotations, and saves them as a JSON file for reuse by downstream tools.

3.3. Cross-domain adaptation

CSO Classifier is adaptable to domains beyond Computer Science. Users can transition to a new field by swapping CSO with an alternative knowledge organisation system [29] and training a word-embedding model on a domain-specific corpus. This is facilitated by the retraining pipeline code released with this manuscript.

To integrate an alternative ontology, a pre-processing stage must first map it into the classifier's internal representation. This schema encodes concept labels, broader and narrower relations, equivalent concepts, and preferred labels. In the current implementation, these structures are populated from the CSO predicates `superTopicOf`, `relatedEquivalent`,

`preferentialEquivalent`, and `rdfs:label`. Detailed guidance on this adaptation process is provided in the accompanying documentation [30].

4. Impact

CSO Classifier 4.0 provides standardised, interpretable, and scalable topic annotations that support scholarly knowledge management and research analytics. By grounding its outputs in CSO, the classifier generates stable and human-readable labels for semantic indexing, information retrieval, recommendation, knowledge graph construction, and metascience analyses. In practical workflows, it reduces the need for manual topic assignment by automatically producing consistent CSO-based annotations from commonly available metadata such as titles, abstracts, and keywords.

CSO-based technologies have been used in several scholarly analytics systems, including knowledge graph construction platforms such as the AIDA Knowledge Graph and CS-KG, digital libraries such as Rexplore, and editorial workflows such as Smart Topic Miner at Springer Nature [14,15,30–34]. These deployments demonstrate the relevance of ontology-based topic annotation beyond academic prototypes and in real-world scientific publishing processes.

The impact of CSO Classifier extends to several research questions in metascience and scholarly analytics [26]. It has been used to explore and analyse scholarly data [33], identify research trends and map scientific domains [14], track the evolution of topics over time [35], and analyse the diversity of expertise within scientific teams [36]. It also supports higher-level analytical tasks such as detecting research communities [35,37], modelling and recommending domain experts [38,39], and studying collaboration patterns and their relationship to research impact [26].

Beyond its original target domain, the CSO framework has enabled cross-domain applications, including the annotation of non-scholarly artefacts such as job advertisements [40], multimedia content [41], and industrial datasets [15]. These applications show that ontology-based topic annotation can bridge academic and industrial contexts, supporting the analysis of knowledge flows between research and practice [42].

A further impact of CSO Classifier 4.0 lies in its support for predictive and forward-looking analytics. By leveraging the semantic structure of CSO, related systems have investigated the forecasting of emerging research topics, academic impact, and technology adoption trends, as demonstrated by Augur and related frameworks [43–45]. These capabilities are relevant for funding agencies, research institutions, and policy-oriented analyses, where the early identification of emerging and impactful research directions can support strategic decision-making.

Finally, CSO Classifier 4.0 contributes reusable semantic resources, including embeddings and concept mappings, which can be used in downstream applications such as clustering [35], recommendation systems [39], and query expansion [46].

5. Conclusion

We presented CSO Classifier 4.0, an open-source software framework for scalable and interpretable topic annotation of scholarly documents using the Computer Science Ontology. The software maps standard scholarly metadata, such as titles, abstracts, and keywords, to CSO concepts through a pipeline combining syntactic matching, embedding-based semantic expansion, and ontology-aware post-processing. In this way, it produces controlled, human-readable topic annotations that can support semantic indexing, knowledge graph construction, recommendation, trend analysis, and large-scale scholarly analytics. The main advancements of CSO Classifier 4.0 include an updated semantic backbone trained on large-scale and recent scholarly metadata, soft classification with topic weighting, domain-aware filtering, compatibility with modern Python environments, and a reproducible pipeline for regenerating semantic resources. These improvements enable the system to better capture evolving research terminology while maintaining consistency with previous versions.

Future work will focus on advancing towards multi-field classification, enabling the system to operate across diverse research domains beyond Computer Science. This direction involves the integration of multi-field knowledge organisation systems and the development of AI approaches capable of processing and analysing heterogeneous linguistic styles across disciplines. A key future extension is an ontology loader that accepts both the target ontology and its mapping schema and can load SKOS- or OWL-based vocabularies into the classifier's internal representation. This would remove dependence on ontology-specific predicates and substantially reduce manual preprocessing.

CRediT authorship contribution statement

Faisal Ramzan: Writing – original draft, Software, Methodology, Data curation. **Angelo Salatino:** Writing – review & editing, Supervision, Methodology, Investigation, Formal analysis, Data curation. **Francesco Osborne:** Writing – review & editing, Supervision, Methodology, Conceptualization. **Diego Reforgiato Recupero:** Writing – review & editing, Project administration, Funding acquisition, Conceptualization. **Enrico Motta:** Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Thanks to Springer Nature for funding this research. Moreover, this article is based on work from COST Action CA24121 KGELL and CA23147 GOBLIN, supported by COST (European Cooperation in Science and Technology).

References

[1] Fortunato S, Bergstrom CT, Börner K, Evans JA, Helbing D, Milojević S, Petersen AM, Radicchi F, Sinatra R, Uzzi B, Vespignani A, Waltman L, Wang D, Barabási A-L. Science of science. Science 2018;359(6379):eaao0185. https://doi.org/10.1126/science.aao0185

[2] Blei DM, Ng AY, Jordan MI. Latent dirichlet allocation. J Mach Learn Res 2003;3:993–1022. https://jmlr.org/papers/v3/blei03a.html.

[3] Jelodar H, Wang Y, Yuan C, Feng X, Jiang X, Li Y, Zhao L. Latent dirichlet allocation (LDA) and topic modeling: models, applications, a survey. Multimed Tools Appl 2019;78(11):15169–211. https://doi.org/10.1007/s11042-018-6894-4

[4] Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems (NeurIPS); 2013. https://arxiv.org/abs/1310.4546.

[5] Rosen-Zvi M, Griffiths TL, Steyvers M, Smyth P. The author-topic model for authors and documents. arXiv:1207.4169 [ArXiv]. 2004. https://api.semanticscholar.org/CorpusID:1997763.

[6] Sebastiani F. Machine learning in automated text categorization. ACM Comput Surv 2002;34(1):1–47. https://doi.org/10.1145/505282.505283

[7] Li X, Ouyang J, Zhou X. Supervised topic models for multi-label classification. Neurocomputing 2015;149:811–9. https://doi.org/10.1016/j.neucom.2014.07.053

[8] Aronson A, Mork J, Gay C, Humphrey S, Rogers W. The NLM indexing initiative's medical text indexer. Stud Health Technol Inform 2004;107:268–72. https://doi.org/10.3233/978-1-60750-949-3-268

[9] Mork J, Aronson A, Demner-Fushman D. 12 years on – is the NLM medical text indexer still useful and relevant? J Biomed Semant Feb 2017;8. https://doi.org/10.1186/s13326-017-0113-5

[10] Peng S, You R, Wang H, Zhai C, Mamitsuka H, Zhu S. DeepMeSH: deep semantic representation for improving large-scale MeSH indexing. Bioinformatics 2016;32:70–9. https://doi.org/10.1093/bioinformatics/btw294

[11] Jin Q, Dhingra B, Cohen W, Lu X. AttentionMeSH: simple, effective and interpretable automatic MeSH indexer. In: Proceedings of the 6th BioASQ workshop a challenge on large-scale biomedical semantic indexing and question answering; 2018. p. 47–56. https://doi.org/10.18653/v1/W18-5306

[12] You R, Liu Y, Mamitsuka H, Zhu S. BERTMeSH: deep contextual representation learning for large-scale high-performance MeSH indexing with full text. Bioinformatics Jul 2020. https://doi.org/10.1101/2020.07.04.187674

[13] Devlin J, Chang M-W, Lee K, Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. 2018. https://doi.org/10.48550/arXiv.1810.04805
[14] Salatino A, Thanapalasingam T, Mannocci A, Osborne F, Motta E. The computer science ontology: a large-scale taxonomy of research areas. In: The semantic web – ISWC; 2018. p. 187–205. https://doi.org/10.1007/978-3-030-00668-6_12
[15] Salatino A, Thanapalasingam T, Mannocci A, Birukou A, Osborne F, Motta E. The computer science ontology: a comprehensive automatically-generated taxonomy of research areas. Data Intell 2020;2:379–416. https://doi.org/10.1162/dint_a_00055
[16] Salatino AA, Osborne F, Thanapalasingam T, Motta E. The CSO classifier: ontology-driven detection of research topics in scholarly articles. In: International conference on theory and practice of digital libraries. Springer; 2019. p. 296–311.
[17] Çelikten T, Onan A. Topic modeling through rank-based aggregation and LLMs: an approach for AI and human-generated scientific texts. Knowl-Based Syst 2025;314:113219. https://doi.org/10.1016/j.knosys.2025.113219
[18] Taylor R, Kardas M, Cucurull G, Scialom T, Hartshorn AS, Saravia E, Poulton A, Kerkez V, Stojnic R. Galactica: a large language model for science. arXiv:2211.09085 [ArXiv]. 2022. https://api.semanticscholar.org/CorpusID:253553203.
[19] Pramanick S, Chellappa R, Venugopalan S. SPIQA: a dataset for multimodal question answering on scientific papers. arXiv:2407.09413 [ArXiv]. 2024. https://api.semanticscholar.org/CorpusID:271161780.
[20] Gao Y, Xiong Y, Gao X, Jia K, Pan J, Bi Y, Dai Y, Sun J, Guo Q, Wang M, Wang H. Retrieval-augmented generation for large language models: a survey. 2023. https://doi.org/10.48550/arXiv.2312.10997
[21] Blecher L, Cucurull G, Scialom T, Stojnic R. Nougat: neural optical understanding for academic documents, [arXiv preprint]. 2023. https://doi.org/10.48550/arXiv.2308.13418
[22] Theocharopoulos PC, Anagnostou P, Georgakopoulos SV, Tasoulis SK, Plagianakos VP. Large language models for efficient topic modeling. Neural Comput Appl 2025;37(29):24421–39. https://doi.org/10.1007/s00521-025-11593-9
[23] Ruan X, Salatino A, Filgueira R, Moraw K, Marcoci A, Derrick G, Callaghan S. Research entity extraction and topic detection from UKRI grant proposals. arXiv:2606.30304. 2026.
[24] Samsi S, Zhao D, McDonald J, Li B, Michaleas A, Jones M, Bergeron W, Kepner J, Tiwari D, Gadepally V. From words to watts: benchmarking the energy costs of large language model inference. In: IEEE conference on high performance extreme computing (HPEC); 2023. p. 1–9. https://doi.org/10.1109/HPEC58863.2023.10363447
[25] Li A, Lin Z, Berg B, Ponnuru S, Wan X, Wang H, Gustafson S, Long DDE, Wang H. ServerlessLLM: locality-enhanced serverless inference for large language models, CoRR abs/2401.14351. 2024. https://doi.org/10.48550/arXiv.2401.14351
[26] Salatino A, Osborne F, Motta E. Cso classifier 3.0: a scalable unsupervised method for classifying documents in terms of research topics. Int J Digit Libr 2022;23(1):91–110.
[27] Pisu A, Pompianu L, Osborne F, Recupero DR, Riboni D, Salatino A. A hybrid AI methodology for generating ontologies of research topics from scientific paper corpora. arXiv:2508.04213. 2025.
[28] Mochurad L, Ilkiv A, Mochurad Y. Chapter 8 - a deep learning approach for medical image classification using XAI and convolutional neural networks. In: Singh A, Singh KK, Izonin I, editors. Responsible and explainable artificial intelligence in healthcare. Academic Press; 2025. p. 183–220. https://doi.org/10.1016/B978-0-443-24788-0.00008-X, https://www.sciencedirect.com/science/article/pii/B978044324788000008X.
[29] Salatino A, Aggarwal T, Mannocci A, Osborne F, Motta E. A survey of knowledge organization systems of research fields: resources and challenges. Quant Sci Stud 2025;6:567–610. https://doi.org/10.1162/qss_a_00363. https://direct.mit.edu/qss/article-pdf/doi/10.1162/qss_a_00363/2509880/qss_a_00363.pdf.
[30] Salatino A, Osborne F. How to use the CSO classifier in other domains; Sep 2019. https://doi.org/10.5281/zenodo.3459286
[31] Angioni S, Salatino A, Osborne F, Reforgiato Recupero D, Motta E. AIDA: a knowledge graph about research dynamics in academia and industry. Quant Sci Stud 2021;2:1–43. https://doi.org/10.1162/qss_a_00162
[32] Dessì D, Osborne F, Reforgiato Recupero D, Buscaldi D, Motta E. CS-KG: a large-scale knowledge graph of research entities and claims in computer science. In: The semantic web – ISWC; 2022. https://doi.org/10.1007/978-3-031-19433-7_39
[33] Osborne F, Motta E, Mulholland P. Exploring scholarly data with rexplore. In: The semantic web – ISWC; 2013. https://doi.org/10.1007/978-3-642-41335-3_29
[34] Osborne F, Salatino A, Birukou A, Motta E. Automatic classification of Springer nature proceedings with smart topic miner. In: The semantic web – ISWC; 2016. p. 383–99. https://doi.org/10.1007/978-3-319-46547-0_33
[35] Osborne F, Scavo G, Motta E. Identifying diachronic topic-based research communities by clustering shared research trajectories. In: Presutti V, d'Amato C, Gandon F, d'Aquin M, Staab S, Tordai A, editors. The semantic web: trends and challenges. Cham: Springer International Publishing; 2014. p. 114–29. https://doi.org/10.1007/978-3-319-07443-6_9
[36] Salatino A, Osborne F, Recupero DR, Angioni S, Motta E. Does diversity of expertise drive citation impact? Evidence from computer science. Scientometrics 2026;1–28. https://doi.org/10.1007/s11192-026-05560-x
[37] Osborne F, Scavo G, Motta E. A hybrid semantic approach to building dynamic maps of research communities. In: Knowledge engineering and knowledge management, vol. 8876; 2014. p. 356–72. https://doi.org/10.1007/978-3-319-13704-9_28
[38] Vergoulis T, Chatzopoulos S, Dalamagas T, Tryfonopoulos C. VeTo: expert set expansion in academia. In: Digital libraries for open knowledge; 2020. p. 48–61. https://doi.org/10.1007/978-3-030-54956-5_4
[39] Rahdari B, Brusilovsky P, Javadian Sabet A. Connecting students with research advisors through user-controlled recommendation. In: Proceedings of the 15th ACM conference on recommender systems, RecSys '21. New York, NY, USA: Association for Computing Machinery; 2021. p. 745–8. https://doi.org/10.1145/3460231.3478879
[40] Derksen F, Dörpinghaus J. Digitalization and sustainability in German continuing education. In: Informatik 2023 - designing futures: Zukünfte gestalten. Bonn: Gesellschaft Für Informatik E.V.; 2023. p. 1945–53. https://doi.org/10.18420/inf2023_194
[41] Ajwani P, Arolkar H. Classification of domains in computer science using random forest algorithm for YouTube dataset. In: Second international conference on image processing and capsule networks. Cham, Switzerland: Springer; 2022. p. 662–70. https://doi.org/10.1007/978-3-030-84760-9_56
[42] Salatino A, Osborne F, Motta E. ResearchFlow: understanding the knowledge flow between academia and industry. In: Knowledge engineering and knowledge managemen; 2020. p. 219–36. https://doi.org/10.1007/978-3-030-61244-3_16
[43] Salatino A, Osborne F, Motta E. AUGUR: forecasting the emergence of new research topics. In: The 18th ACM/IEEE; 2018. p. 303–12. https://doi.org/10.1145/3197026.3197052
[44] Chatzopoulos S, Vergoulis T, Kanellos I, Dalamagas T, Tryfonopoulos C. ArtSim: improved estimation of current impact for recent articles. In: Bellatreche L, Bieliková M, Boussaïd O, Catania B, Darmont J, Demidova E, Duchateau F, Hall M, Merčun T, Novikov B, Papatheodorou C, Risse T, Romero O, Sautot L, Talens G, Wrembel R, Žumer M, editors. ADBIS, TPDL and EDA 2020 common workshops and doctoral consortium. Cham: Springer International Publishing; 2020. p. 323–34. https://doi.org/10.1007/978-3-030-55814-7_27
[45] Osborne F, Motta E. Pragmatic ontology evolution: reconciling user requirements and application performance. In: 17th international semantic web conference, Monterey, CA, USA, October 8–12, 2018, proceedings, Part I; 2018. p. 495–512. https://doi.org/10.1007/978-3-030-00671-6_29
[46] Beck M, Rizvi STR, Dengel A, Ahmed S. From automatic keyword detection to ontology-based topic modeling. In: Document analysis systems: 14th IAPR international workshop, DAS 2020, Wuhan, China, July 26–29, 2020, proceedings. Berlin, Heidelberg: Springer-Verlag; 2020. p. 451–65. https://doi.org/10.1007/978-3-030-57058-3_32