International Journal on Digital Libraries
https://doi.org/10.1007/s00799-021-00305-y

CSO Classifier 3.0: a scalable unsupervised method for classifying documents in terms of research topics

Angelo Salatino[1] · Francesco Osborne[1] · Enrico Motta[1]

Received: 11 February 2021 / Revised: 30 June 2021 / Accepted: 11 July 2021

Abstract

Classifying scientific articles, patents, and other documents according to the relevant research topics is an important task, which enables a variety of functionalities, such as categorising documents in digital libraries, monitoring and predicting research trends, and recommending papers relevant to one or more topics. In this paper, we present the latest version of the CSO Classifier (v3.0), an unsupervised approach for automatically classifying research papers according to the Computer Science Ontology (CSO), a comprehensive taxonomy of research areas in the field of Computer Science. The CSO Classifier takes as input the textual components of a research paper (usually title, abstract, and keywords) and returns a set of research topics drawn from the ontology. This new version includes a new component for discarding outlier topics and offers improved scalability. We evaluated the CSO Classifier on a gold standard of manually annotated articles, demonstrating a significant improvement over alternative methods. We also present an overview of applications adopting the CSO Classifier and describe how it can be adapted to other fields.

Keywords Scholarly data · Digital libraries · Ontology · Text mining · Topic detection · Science of science

1 Introduction

Characterising scholarly documents according to their relevant research topics enables a variety of functionalities, such as: (i) enhancing semantically the metadata of scientific publications, (ii) categorising proceedings in digital libraries, (iii) producing smart analytics, (iv) generating recommendations, and v) detecting research trends [53]. In general, state-of-the-art approaches either classify papers in a top-down fashion, taking advantage of pre-existent categories from domain vocabularies, such as MeSH[1], PhySH[2], and the STW Thesaurus for Economics[3], or instead proceed in a bottom-up fashion, by means of topic detection methods, such as probabilistic topic models [8,24]. The first solution has the advantage of relying on a set of formally defined research topics associated with human readable labels; however, it requires such a controlled vocabulary to be available. Conversely, while bottom-up approaches do not require a predefined vocabulary, they tend, however, to produce noisier and less interpretable results [42].

In 2019, we released the Computer Science Ontology (CSO) [55], a large-scale, granular, and automatically generated ontology of research areas in *Computer Science*, which includes more than 14K research topics and 159K semantic relationships. CSO has been adopted by Springer Nature editors to classify the computer science proceedings they publish, such as the well-known LNCS series [45]. We also released CSO as a publicly available resource, to foster its adoption and the development of novel CSO-powered applications. However, many users interested in adopting CSO for characterising their data have limited understanding of semantic technologies and how to use an ontology for annotating documents. Hence, the natural next step was to develop a classifier that supports the annotation of research papers according to CSO [54].

✉ Angelo Salatino
angelo.salatino@open.ac.uk

Francesco Osborne
francesco.osborne@open.ac.uk

Enrico Motta
enrico.motta@open.ac.uk

[1] Knowledge Media Institute, The Open University, Milton Keynes, UK

[1] Medical Subject Headings: https://www.nlm.nih.gov/mesh/.

[2] PhySH—Physics Subject Headings: https://physh.aps.org.

[3] STW Thesaurus for Economics: http://zbw.eu/stw.

Published online: 22 July 2021

In this paper, we present the latest version of the CSO Classifier (v3.0), a scalable solution for automatically classifying research papers according to the Computer Science Ontology. The CSO Classifier takes as input the textual components of a scientific paper (usually title, abstract, and keywords) and returns a selection of research topics drawn from CSO. It operates in three steps. First, it finds all topics in the ontology that are explicitly mentioned in the input text. Then, it identifies further semantically related topics by utilising part-of-speech tagging and world embeddings. Finally, it discards outliers and enriches this set of topics by taking advantage of the CSO taxonomy to include their super-areas. The Classifier has been evaluated on a gold standard of manually annotated research papers, demonstrating a significant improvement over a number of alternative approaches, including an earlier version (v2.0), which has been used by Springer Nature editors since 2018 to support the annotation of Computer Science proceedings [50].

This paper extends our earlier work, which was presented at TPDL 2019 [54]. In particular, the novel contributions presented here are as follows:

1. an improved version of the CSO Classifier (3.0), which takes advantage of a novel component for discarding outliers from the result set;
2. a new evaluation evidencing the improvement in performance brought about by the outlier detection component;
3. a new solution for improving the classifier's scalability;
4. a strategy for applying the CSO Classifier to other disciplines;
5. an overview of applications developed by adopters of the CSO Classifier;
6. a revisited and updated literature review.

The CSO Classifier is implemented in Python and can be installed from PyPI[4] using the command: `pip install cso-classifier`. It can also be downloaded from https://github.com/angelosalatino/cso-classifier. The data produced in the evaluation and the word embeddings model are available at http://w3id.org/cso/cso-classifier.

The rest of the manuscript is organised as follows. In Sect. 2, we review the literature regarding the classification of research articles and outline current limitations. In Sect. 3, we discuss the Computer Science Ontology. In Sect. 4, we describe the CSO Classifier and its modules. Next, in Sect. 5 we evaluate the CSO Classifier against alternative approaches, focusing on the performance of the new method for detecting outliers. In Sect. 6, we discuss our new solution for improving the scalability of the CSO Classifier, and in Sect. 7, we show how to apply the classifier to other fields of Science. In Sect. 8, we provide an overview of applications developed by early adopters of the CSO Classifier. Finally, in Sect. 9 we summarise the main contributions and outline future directions of research.

2 Literature review

The goal of *topic classification* is to identify the relevant subjects within a set of documents. Specifically, in the scholarly communication domain it aims at identifying research topics within scientific documents. These approaches can be typically characterised according to four main categories: (i) topic modelling, (ii) supervised machine learning approaches, (iii) approaches based on citation networks, and (iv) approaches based on natural language processing. We devote this section to present the main state-of-the-art approaches in these four categories and discuss their limitations.

2.1 Topic modelling

Topic modelling is a type of statistical approach for discovering topics that occur in a collection of documents. One of the most acclaimed approaches is the latent Dirichlet analysis developed by Blei et al. [7]. LDA is a three-level hierarchical Bayesian model to retrieve latent—or hidden—patterns in texts. The basic idea is that each document is modelled as a mixture of topics, where a topic is a multinomial distribution over words, characterised as a discrete probability distribution, defining the likelihood that each word will appear in a given topic. In other words, LDA aims to discover the latent structure which connects words to topics and topics to documents. This is achieved by computing the conditional distribution of the hidden variables (topics) given the observed variables (words) [7]. Over the years, LDA influenced many other approaches, such as Griffiths et al. [24] in which the authors designed a generative model for document collections. Their author–topic model simultaneously modelled the content of documents and the interests of authors. Bolelli et al. [8] further extended the author–topic model introducing the segmented author–topic model (S-ATM), a model that uses temporal ordering of documents in order to identify topic evolution and then exploits citations to evaluate the weights for the main terms in documents. Other approaches that fall within the topic modelling category are the latent semantic analysis (LSA) [17], the probabilistic latent semantic analysis (pLSA) [26], and the correlated topic model (CTM) [31], which is a consequent work from Blei et al. in mitigating the issues of the original LDA [7].

The advantage of these approaches is that they can achieve good results in the absence of a strong a priori categorisation and do not require training data. However, the resulting topics typically require manual verification by domain experts (e.g.

[4] PyPI: https://pypi.org/.

senior researchers) in order to assign to them sound labels and assess the best number of topics for a set of documents [6]. In addition, using a high number of topics usually introduces noise. As a result, the number of topics is normally kept low, with the consequence that the resulting classification is not very granular.

2.2 Supervised machine learning approaches

This second category of approaches for classifying research topics aims at developing a multi-class model in which each class refers to a research topic.

Mai et al. [36] developed an approach to subject classification using deep learning techniques, and they applied it on a set of papers annotated with the STW Thesaurus for Economics (∼5K classes) and MeSH (∼27K classes). Similarly, Chernyak [15] presented a supervised approach for annotating papers in Computer Science with topics from the ACM Computing Classification System[5].

Caragea et al. [12] developed an approach that combines research article's textual content and citation network to predict the topic of an article. Specifically, they trained two different classifiers on the two sets of data with the idea that their combined information has the potential to improve topics classification. Then, they trained their classifiers over a corpus of 3186 papers distributed over six classes, such as *agents*, *artificial intelligence*, *information retrieval*, *human computer interaction*, *machine learning*, and *databases*. However, the goal of this technique is to classify papers according to one single broad subject category. In contrast, we propose an approach for classifying research documents according to more than one topic.

Kandimalla et al. [29] propose a deep attentive neural network for classifying papers according to 104 Web of Science[6] subject categories. Their classifier was trained on 9M abstracts from Web of Science, and it can be directly applied to abstract without the need of additional information, e.g. citations. However, their approach finds difficulties in discriminating between overlapping categories. This is due to the nature of research papers, which encompass more than one subject.

HierClasSArt [1] is a recent approach for classifying articles according to a taxonomy of mathematical topics which uses combination of neural networks and knowledge graphs. It generates a knowledge graph from the abstracts and then classifies the papers using a latent representations that takes into account both the concepts in the knowledge graph and the metadata.

Garcia-Silva et al. [23] focus on the task of classifying scientific publications against a taxonomy of scientific disciplines, taking advantage of BERT [20] and its different flavours specialised in the scientific domain: BioBERT [33] and SciBERT [5]. Specifically, they train a multi-label classifier on 450K papers tagged with the 22 first-level categories of the ANZSRC[7] taxonomy.

One of the major difficulties arising when developing supervised approaches is related to the gold standard [12]. It requires an intensive manual labelling effort to generate a gold standard that includes all possible classes (research topics) and that is also balanced with regard to the number of papers labelled per topic. Indeed, very broad areas tend to have many published papers and hence are extensively represented, while very specific areas tend to have fewer papers.

Some recent deep learning models, such as few-shot learning and zero-shot learning, may be able to mitigate this issue, but they require more research and refinement [69].

2.3 Approaches based on citation networks

Another set of approaches for classifying documents use citation networks, and most of them are based on the principle of clustering scientific documents by means of a co-citation analysis. The use of citations for detecting topics has been explored in many different ways and some approaches combine citations with other entities, such as keywords and abstracts.

Upham et al. [64] used the Web of Science corpus to identify emerging topics within the years 1999–2004, represented as co-citation clusters. Small et al. [59] also performed co-citation analysis over Scopus[8] data, aiming at identifying top 25 emergent topics for each year from 2007 to 2010.

Boyack and Klavans [10] built a map of science using 20 million research articles over 16 years using co-citation techniques. Through this map, it is possible to observe the disciplinary structure of science in which papers of the same area tend to group together.

Van Eck et al. [65] developed CitNetExplorer[9] and VOSviewer[10], which can be used to cluster publications and to analyse the resulting clustering solutions. These two applications work at two different levels of granularity. Through visualisation techniques, CitNetExplorer focuses on the analysis of clusters at the level of individual publications, while VOSviewer focuses on the analysis of clusters at an aggregate level. CitNetExplorer and VOSviewer are heavily used

[5] ACM Computing Classification System— https://dl.acm.org/ccs.

[6] Web of Science—https://clarivate.com/webofsciencegroup/solutions/web-of-science/.

[7] Australian and New Zealand Standard Research Classification—https://www.abs.gov.au/ausstats/abs@.nsf/mf/1297.0.

[8] Scopus—https://www.scopus.com.

[9] CitNetExplorer—https://www.citnetexplorer.nl.

[10] VOSviewer—[11].

by scientometricians to analyse developments in science [25,70].

The main drawback of citation-based approaches is that they are able to assign each document to only one topic, and a document is rarely monothematic.

2.4 Approaches based on natural language processing

This category of topic classifiers groups all those unsupervised approaches that take advantage of natural language processing techniques, such as text analysis [16,45] and word embeddings [72].

For instance, Decker [16] introduced an unsupervised approach that generates paper–topic relationships by exploiting keywords and abstracts, in order to analyse the trends of topics on different timescales.

Jo et al. [27] developed an approach that combines distributions of terms (i.e. n-grams) with the distribution of the citation graph related to publications containing that term. In particular, the authors assume that if a term is relevant for a topic, documents containing that term will have a stronger connection than randomly selected ones. Then, their algorithm identifies the set of terms having citation patterns exhibiting synergy.

Another set of methods rely only on keywords. For instance, Duvvuru et al. [21] built a network of co-occurring keywords and subsequently performed statistical analysis by calculating degree, strength, clustering coefficient, and end-point degree to identify clusters and associate them to research topics.

Some recent approaches use word embeddings, aiming to quantify semantic similarities between words based on their distributional properties in samples of text. For example, Zhang et al. [72] applied k-means on a set of words represented as embeddings. However, all these approaches to topic detection need to generate the topics from scratch rather than exploiting a domain vocabulary or ontology, resulting in noisier and less interpretable results [42]. The Microsoft Academic Graph's team developed an approach for tagging document according to the *fields of study*, a controlled vocabulary of research topics [58]. This approach associates embeddings to both topics and articles and computes the cosine similarity between them. It then classifies each article with all the topics that score a similarity higher than a threshold. However, this technique is not described in details and the evaluation data are not available, making it difficult for the scientific community to reuse or compare with this method.

In sum, we still lack practical unsupervised approaches for classifying papers according to a granular set of topics. The CSO Classifier aims to address this gap, by providing high-quality automatic classification of research papers in the domain of Computer Science.

3 The computer science ontology

The Computer Science Ontology (CSO) is a large-scale ontology of research areas in the field of Computer Science. It was automatically generated using the Klink-2 algorithm [42] on a dataset of 16 million publications, mainly in the field of Computer Science [44]. Compared to other solutions available in the state of the art (e.g. the ACM Computing Classification System), the Computer Science Ontology includes a much higher number of research topics, which can support a granular representation of the content of research papers, and it can be easily updated by running Klink-2 on recent corpora of publications.

The current version of CSO[12] includes 14K semantic topics and 159K relationships. The main root is Computer Science; however, the ontology includes also a few additional roots, such as Linguistics, Geometry, and Semantics. The CSO data model[13] is an extension of SKOS[14], and it includes four main semantic relations:

- *superTopicOf*, which indicates that a topic is a super-area of another one (e.g. Semantic Web is a super-area of Linked Data).
- *relatedEquivalent*, which indicates that two topics can be treated as equivalent for the purpose of exploring research data (e.g. Ontology Matching and Ontology Mapping).
- *contributesTo*, which indicates that the research output of one topic contributes to another.
- *owl:sameAs*, this relation indicates that a research concept in CSO is equivalent to a concept described in an external resource, such as DBpedia, Wikidata, and Freebase.

CSO is available through the CSO Portal[15], a web application that enables users to download, explore, and visualise sections of the ontology. Moreover, users can use the portal to provide granular feedback at different levels, such as rating topics and relationships, and suggesting missing relationships. The reader can refer to [56] for a more detailed description of CSO and how it has been developed.

CSO is used by several tools and proved to effectively support a wide range of tasks, such as exploring

[12] CSO is available for download at https://w3id.org/cso/downloads.

[13] CSO Data Model—https://cso.kmi.open.ac.uk/schema/cso.

[14] SKOS Simple Knowledge Organization System—http://www.w3.org/2004/02/skos.

[15] Computer Science Ontology Portal—https://cso.kmi.open.ac.uk.

and analysing scholarly data (e.g. Rexplore [44], ScholarLensViz [35], ConceptScope [71]), detecting research communities (e.g. TST [47], RCMB [46]), identifying domain experts (e.g. VeTo [66]), recommending articles [62] and video lessons [9], generating knowledge graphs [18] (e.g. Temporal KG [49], AIDA KG [2], AI KG [19]), knowledge graph embeddings (e.g. Trans4E [39]), and topic models (e.g. CoCoNoW [4]), and predicting academic impact (e.g. ArtSim [13]), research topics (e.g. Augur [52]), ontology concepts (e.g. SIM [11], POE [43]), and technologies (e.g. TTF [41], TechMiner [40]). CSO has also been adopted by Springer Nature, one of the top two international academic publishers, which uses it to support a number of innovative applications, including (i) Smart Topic Miner [51], a tool designed to assist the Springer Nature editorial team in classifying proceedings, (ii) Smart Book Recommender [62], an ontology-based recommender system for selecting books to market at academic venues, and (iii) the AIDA Dashboard [2], a web application for exploring and making sense of scientific conferences.

4 CSO classifier

The CSO Classifier is a tool that takes as input the textual components of a research paper (usually title, abstract, and keywords) and outputs the relevant topics drawn from CSO. It adopts an unsupervised approach, which has been shown to perform well against alternative methods—see Sect. 5. Here, we should emphasise that, although the classifier leverages a word embedding model, we consider this approach unsupervised because it does not require labelled examples, consistently with the characterisation of unsupervised methods in the work of Song and Roth [61] and other relevant literature [34].

The choice of an unsupervised approach is quite natural in our scenario. As already pointed out, a supervised machine learning algorithm would require an extensive set of annotated examples, covering the thousands of research topics provided by CSO. Clearly, such a dataset does not exist and it would be non-trivial to develop. A further advantage of an unsupervised approach is that there is no need for retraining the algorithm when new versions of the Computer Science Ontology are released.

The CSO Classifier consists of three main components: (i) the syntactic module, (ii) the semantic module, and iii) the post-processing module. Figure 1 shows its architecture.

The *syntactic module* parses the input documents and identifies CSO concepts that are explicitly referred in the document. The *semantic module* uses part-of-speech tagging to identify promising terms and then exploits word embeddings to infer semantically related topics. Finally, the *post-processing module* combines the results of these two modules, discards outliers, and enhances the topic set by including relevant super-areas. To assist the description of our approach, we will use the sample paper [30] shown in Table 1 as a running example.

4.1 Syntactic module

The syntactic module identifies topics that are explicitly referred in the textual input, mapping n-grams to CSO concepts. At the beginning, the algorithm removes English stop words and collects unigrams, bigrams, and trigrams. Then, for each n-gram, it computes its Levenshtein similarity with the labels of the topics in CSO. Research topics having similarity equal or higher than a threshold (i.e. the constant *msm*) with an n-gram, are selected for the final set of topics. We empirically set *msm* to 0.94, which allows us to recognise lexical variations of CSO topics, such as hyphens (e.g. "knowledge based systems" and "knowledge-based systems"), plurals (e.g. "database" and "databases"), and British versus American spelling (e.g. "data visualisation" and "data visualization").

In Table 2, we report the list of topics returned by the syntactic module for the running example. In particular, we can identify some key topics that are central to the analysed paper, such as: "neural networks", "image segmentation", "recurrent neural networks", and "image retrieval".

4.2 Semantic module

The semantic module is designed to identify topics that are semantically related to the paper, but may not be explicitly mentioned in the text. It uses word embeddings produced by a word2vec model to compute the semantic similarity between the terms in the document and the CSO concepts.

The semantic module follows four steps: (i) entity extraction, (ii) CSO concept identification, (iii) concept ranking, and (iv) concept selection.

In the following sections, we first describe how we trained the word embedding model and then illustrate the algorithm.

4.2.1 Word embedding generation

We generated the word embeddings by training a word2vec model [37,38] on a collection of text from Microsoft Academic Graph (MAG)[16]. MAG is a knowledge graph containing scientific publication records, citation relationships, authors, institutions, journals, conferences, and fields of study. It is the largest dataset of scholarly data publicly available [68], and, as of April 2021, it contains more than 258 million publications.

[16] Microsoft Academic Graph—https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/.

Fig. 1 Architecture of the CSO Classifier

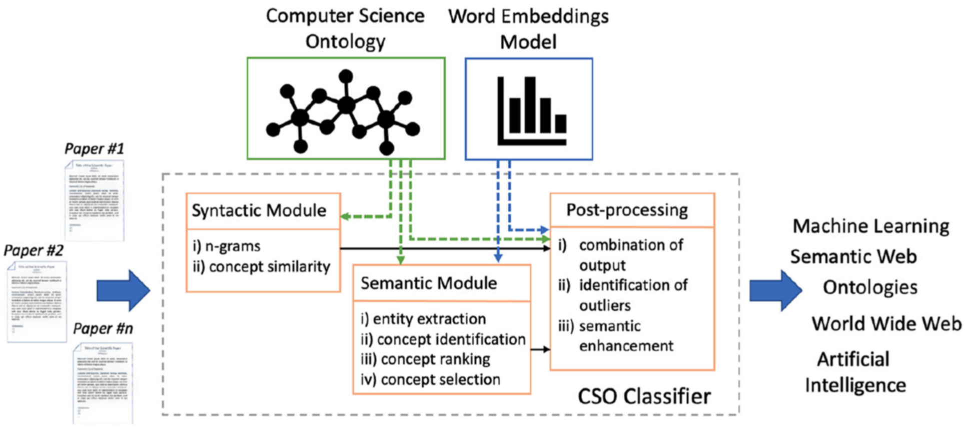


Table 1 Sample paper that will be analysed by the CSO Classifier [30]

Deep visual-semantic alignments for generating image descriptions

Authors: Andrej Karpathy and Li Fei-Fei

Abstract: We present a model that generates natural language descriptions of images and their regions. Our approach leverages datasets of images and their sentence descriptions to learn about the inter-modal correspondences between language and visual data. Our alignment model is based on a novel combination of convolutional neural networks over image regions, bidirectional recurrent neural networks over sentences, and a structured objective that aligns the two modalities through a multimodal embedding. We then describe a multimodal recurrent neural network architecture that uses the inferred alignments to learn to generate novel descriptions of image regions. We demonstrate that our alignment model produces state of the art results in retrieval experiments on Flickr8K, Flickr30K and MSCOCO datasets. We then show that the generated descriptions significantly outperform retrieval baselines on both full images and on a new dataset of region-level annotations

Keywords: image retrieval, image segmentation, learning (artificial intelligence), natural language processing, recurrent neural nets

We first downloaded titles and abstracts of 4,654,062 English papers in the field of Computer Science. Then, we pre-processed the data by replacing spaces with underscores in all n-grams matching the CSO topic labels (e.g. “digital libraries” became “digital_libraries”) and we performed a co-location analysis to identify frequent bigrams and trigrams (e.g. “highest_accuracies”, “highly_cited_journals”). These frequent n-grams were identified by analysing combinations of words that co-occur together, as suggested in [38][17]. Indeed, while it is possible to obtain the vector of an n-gram by summing the embedding vectors of all its tokens, the resulting representation usually is not as good as the one obtained by considering the n-gram as a single word during the training phase. As an example, Fig 2 shows a two-dimensional projection of the term *semantic_web* and of the vector obtained by summing the embeddings of *semantic* and *web* (*semantic*+*web*). We can appreciate that *semantic_web* is closer than *semantic*+*web* to well-known semantic web technologies such as *linked_data*, *ontologies*, and *RDF*.

Table 2 Topics returned from the syntactic module when processing the paper in Table 1

artificial intelligence, embeddings, image retrieval, image segmentation, languages, learning, natural language processing, natural languages, network architecture, neural networks, recurrent neural networks

Finally, we trained the word2vec model, after testing several combinations of parameters[18].

4.2.2 Entity extraction

Our main assumption is that research concepts are represented either by nouns or adjectives followed by nouns. Indeed, 12% of the topics in CSO consist of just nouns and the remaining 88% percent of topics follow the adjective–noun pattern (e.g. semantic web, neural networks). Analysing only these text chunks allows us to speed up computation and avoid combinations that usually result in false positives. To this end, the classifier tags each token in the text according

[17] For the collocation analysis, we used *min-count* = 5 and *threshold* = 10.

[18] The final parameters of the word2vec model are: *method* = skipgram, *embedding-size* = 128, *window-size* = 10, *min-count-cutoff* = 10, *max-iterations* = 5

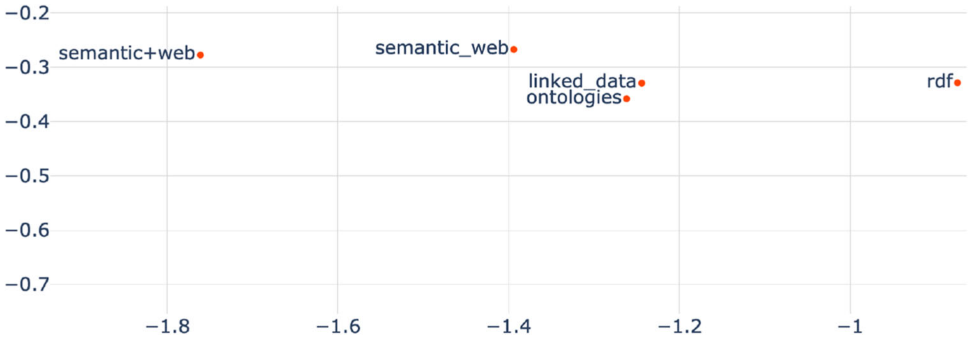


Fig. 2 Scatter plot of PCA projection of the words: *semantic_web*, *semantic+web*, *linked_data*, *ontology*, and *RDF*

to its part of speech (e.g. nouns, verbs, adjectives, adverbs) and then applies a grammar-based chunk parser to identify chunks of words, expressed by the following grammar:

$$< \mathrm{JJ}.* > * < \mathrm{NN}.* > + \tag{1}$$

where JJ represents adjectives and NN represents nouns.

4.2.3 CSO concept identification

In this phase, the classifier processes the extracted chunks of words and uses the word embedding model to identify semantically related topics. First, it decomposes these chunks in unigrams, bigrams, and trigrams. Next, for each gram, it retrieves the top ten similar words (with cosine similarity higher than 0.7) from the word2vec model. The CSO topics matching these words are added to the result set. Figure 3 illustrates this process more in detail.

When processing bigrams or trigrams, the classifier joins their tokens using an underscore, e.g. “web_application”, in order to refer to the corresponding word in the word2vec model. If a n-gram is not available within the vocabulary of the model, the classifier generates its representation averaging the embedding vectors of the given tokens.

A specific CSO concept can be identified multiple times, for two main reasons: (i) the same n-gram can appear multiple times within the title, abstract and keywords or (ii) multiple n-grams can be semantically related to the same CSO concept. Indeed, the concept “image_segmentation” can be inferred by several semantically related n-grams, such as: “segmentation”, “image_analysis”, “segmentation_method”, “contour_extraction”, “segmentation_techniques”, “object_segmentation”, and “image_segmentation_algorithm”.

4.2.4 Concept ranking

The previous step tends to produce a large number of topics (typically more than 70), some of which are only marginally related to the analysed research paper. For instance, when processing the paper in Table 1, some n-grams triggered topics like “automatic_segmentations” and “retrieval_algorithms”, that may be considered less relevant. For this reason, the semantic module weighs the identified CSO concepts according to their overall relevancy to the paper. The relevance score of a topic is computed as the product between the number of times it was identified (frequency) and the number of unique n-grams that led to it (diversity). For instance, if a concept has been identified seven times, from three different n-grams, its final score will equal 21. In addition, if a topic is directly mentioned in the paper, its score is set to the maximum score found. Finally, the classifier ranks the topics according to their relevance score.

Table 3 Topics returned from the semantic module when processing the paper in Table 1

color images, **digital image**, embeddings, **feedforward neural networks**, image retrieval, image segmentation, natural language processing, natural languages, network architecture, **recurrent networks**, recurrent neural networks, **reference image**, **retrieval performance**

In bold are the topics missing from the syntactic module in Table 2

4.2.5 Concept selection

Typically, the relevance scores of the candidate topics follow a long-tail distribution. For this reason, the classifier employs the elbow method [57] to ensure that only relevant topics are eventually selected. This technique was originally designed to find the appropriate number of clusters in a dataset. In particular, it observes the cost function for different numbers of clusters. The optimal number of clusters is then located at the elbow of the resulting curve. This point balances the number of clusters and the percentage improvement of the cost function.

In Table 3, we report the list of topics obtained using the semantic module on the running example. In bold are the topics that were detected by the semantic module, but not by the syntactic module.

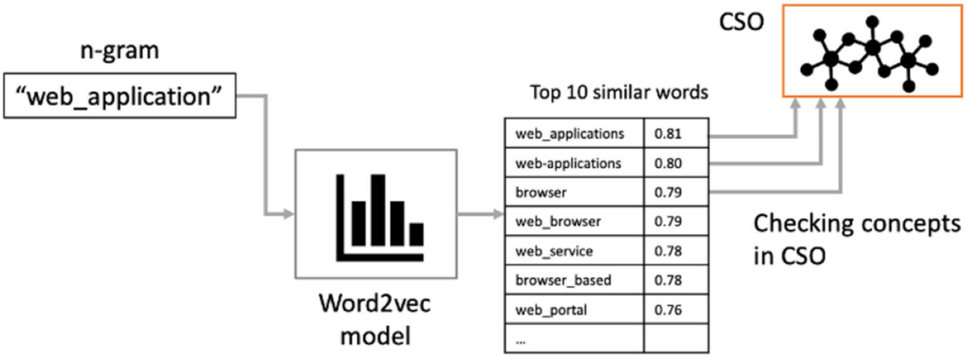


Fig. 3 Identification of CSO concepts semantically related to n-grams

4.3 Post-processing

The post-processing module of the CSO Classifier combines the output of both the syntactic and semantic modules, discards outliers, and enhances the topic set by including relevant super-areas.

It takes the union of the result sets of the two modules, since this solution maximises the f-measure according to our experiments (see Sect. 5). However, it discards the topics returned by the semantic module that appear among the first *n* most occurring words in the vocabulary of the embeddings (*n=3,000* in the current version). This is done to exclude very generic terms (e.g. "language", "learning", "component") that tend to have a good similarity value with a large number of n-grams, typically resulting in too many false positives.

Differently from the original design presented in [54], version 3.0 of the CSO Classifier takes advantage of a new component that identifies outliers. We introduced this solution to address a specific type of recurring errors pointed out by the users of the CSO Classifier during the last year: in some cases, the approach would return erroneous topics that were conceptually distant from the others. Even in the cases in which these outliers were actually mentioned in the document, they were very marginal to the core topics and often identified as erroneous by human users. Indeed, several false positives produced by previous versions would follow this pattern. For instance, let us consider a set of topics such as *bioinformatics*, *database*, *graph algorithms*, *keyword queries*, *query evaluation*, *query languages*, *query processing*, *query results*, *rdf*, *rdf graph*, *recommendation*, *routing scheme*, *semantic web*, *single path*, *sparql*, and *user query*. Here, it is clear that the focus of the paper leans towards the application of SPARQL queries. Hence, topics such as *bioinformatics*, *recommendation*, *routing scheme*, and *single path* are in principle outliers and therefore candidates for rejection.

To address this issue, we studied the effect of outliers on the quality of our results and developed different methods for outliers detection. The main idea behind all these approaches is to compute the pairwise similarity of the topics and then delete the uncoupled ones. In order to find a good solution in this space, we performed extensive experiments and compared five metrics for measuring pairwise similarity and three approaches for selecting uncoupled topics. Section 5.2 describes in detail these alternative solutions, and Sect. 5.4 reports the best results for each combination. The evaluation confirmed that some outlier detection approaches improve the performance of our classifier.

Section (4.3.1) describes the method for detecting outliers that produced the best performance in our experiments and is therefore included in the new version of the CSO Classifier. Section (4.3.2) describes the semantic enhancement component, which defines the last step of the post-processing module.

4.3.1 Outlier detection

In order to identify outliers, we compute the pairwise similarity of the topics and identify disconnected ones. We then apply a set of heuristics to detect outliers to discard. In the following subsections, we describe this approach more in detail.

Computation of Similarity Matrix We compute the pairwise similarity between pairs of topics by taking the maximum value between two similarity indices: *graph similarity* and *embeddings similarity*.

For the graph similarity, we take advantage of the graph representation of the Computer Science Ontology, where the topics and the relationships between them are, respectively, expressed by nodes and edges. Such representation allows us to identify the distance between pair of topics by applying the Dijkstra algorithm for finding the shortest path between them and thus computing the length of such path. The idea is that two topics are similar if they reside very close to each other within the CSO graph.

In practice, given a pair of topics we compute the length of their shortest path and we populate a distance matrix D_{graph}. The distance falls in the range $[1, n]$, where n is the diameter of the CSO graph (15 in the current version). The similarity matrix S_{graph} is the complement to 1 of the normalised distance matrix: $S_{graph} = 1 - norm(D_{graph})$.

For the embedding similarity, we use the word2vec model trained for the CSO Classifier and described in Sect. 4.2.1. The idea is that, since the word embedding model captures a semantic relation between words, it follows that similar topics have similar representations. In particular, for each pair of topics we compute the cosine similarity of their embeddings and we populate a similarity matrix $S_{embeddings}$. For the multi-word topics that are not available in the model vocabulary, we create their embedding representation by averaging the embeddings of all their tokens.

We create the final similarity matrix by comparing S_{graph} and $S_{embeddings}$, and taking the element-wise maxima. Such similarity matrix is a square ($n \times n$) matrix, where n is the number of topics. Since the similarity between a topic A and a topic B is equivalent to the similarity between topic B and topic A, the similarity matrix is symmetric. Moreover, similarity values are between 0 and 1.

Identification of outliers In order to identify isolated topics we binarise the similarity matrix, based on whether the similarity between a pair of topics is higher or lower than a threshold k. To this end, we devised a dynamic threshold that considers as 1 the top k similarity values, where k is a multiple of the number of topics. Since the similarity matrix is symmetric, to discard redundant similarities, we identify the top k values by considering only the upper triangle and we exclude the 1s on the main diagonal. During the evaluation, and specifically in Sect. 5.5, we observe that the optimal value of k is 1, or in other words we select the same number of top similarities as the number of topics.

We then identify all topics that have no relationship with any other in the binarised matrix and exclude from this set: i) the syntactic topics that have more than two grams (multi-grams), ii) the super topics of the retained group, and iii) the topics that have high string similarity with the retained group. The remaining topics are considered outliers and discarded from the ones returned by the CSO Classifier.

We exclude from this process the topics returned by the syntactic module with more than two tokens because they are quite specific and the fact that they have been syntactically matched means they are central to the classified document. Similarly, we preserve super topics because they are entailed by at least another topic in the set. For a similar reason, we also retain topics with a high string similarity to at least another topic. We tested different string similarity measures such as cosine, Jaro-Winkler, Levenshtein, and the normalised Longest Common Subsequence. The method that produced best results is the normalised Longest Common Subsequence with a threshold of 0.5.

4.3.2 Semantic enhancement

The resulting set of topics is enriched by inferring all the direct super topics, according to the *superTopicOf* relationship in CSO [56]. For instance, when the classifier extracts the topic "machine learning", it will infer also "artificial intelligence". By default, the classifier enriches the set of topics by adding only the direct super topics. However, to provide a broader set of topics fitting the research paper, it is also possible to infer the list of all their super topics up to the root, i.e. Computer Science.

In Table 4, we report a brief summary on the list of topics obtained at the different stages of this post-processing module. The union set includes the results from both the syntactic and semantic modules. The outlier topics are the ones excluded by the newly developed component, and the enhanced topics are the super topics. Among the latter, we can see there are several other topics that are pertinent to the paper in Table 1, such as: "image analysis", "image enhancement", and "information retrieval systems".

5 Evaluation

To measure the performance of the approach introduced in this manuscript, we evaluated 10 versions of the CSO Classifier and 13 alternative approaches on the task of classifying papers drawn from a manually generated gold standard. In particular, the versions of the CSO Classifier tested in this evaluation differ according to the mechanism used for outlier detection, which is the main novelty introduced by CSO Classifier 3.0.

In Sect. (5.1), we describe the creation of the gold standard. Then, in Sect. 5.2 we discuss the metrics and approaches for outlier detection. Finally, Sect. 5.3 presents the experimental set-up, Sect. 5.4 reports the results, and Sect. 5.5 focuses on the different techniques for identifying the outlier topics.

5.1 Creation of the gold standard

Due to the absence of corpora annotated with fine-grained topics and being the Computer Science Ontology recently released, we lacked a gold standard for evaluating the CSO Classifier. To this end, we developed a gold standard with the support of 21 domain experts in classifying 70 papers according to the CSO ontology. The objectives of this gold standard are twofold. Firstly, it allows us to evaluate the classifier. Secondly, it will be a valuable resource to facilitate further evaluations by other members of the research community.

Table 4 Topics obtained from the enhancement process when processing the running paper

Union	artificial intelligence, color images, digital image, embeddings, feedforward neural networks, image retrieval, image segmentation, languages, learning, natural language processing, natural languages, network architecture, neural networks, recurrent networks, recurrent neural networks, reference image, retrieval performance
Outlier topics	embeddings, learning
Enhanced	backpropagation algorithm, color image processing, computational linguistics, computer programming languages, computer science, database systems, image analysis, image enhancement, image matching, image processing, image quality, information retrieval, information retrieval systems, linguistics, machine learning, natural language processing systems, network protocols, object oriented programming, pattern recognition, query languages, semantics

5.1.1 Data preparation

From Microsoft Academic Graph, we selected the 70 most cited papers published within the decade 2007–2017 from the fields of *Semantic Web* (23 papers), *Natural Language Processing* (23 papers), and *Data Mining* (24 papers)[19].

Next, we contacted 21 researchers in these fields, at various levels of seniority and without prior experience with CSO, and asked them to annotate ten papers each. We organised the data collection so that each paper is annotated by three experts, and we use the majority vote to address disagreements. We randomly assigned the papers to the experts, while minimising the number of shared papers between each pair of experts, in order to foster diversity.

5.1.2 Data collection

To support the domain experts during the annotation process, we developed a web application. Through this application, the authors were able to read title, abstract, keywords (when available), and the set of candidate topics of each paper. We asked the experts to thoroughly read all the information and assess a set of candidate topics by dragging and dropping them in two different baskets: *relevant* and *not relevant*. The application allowed the experts also to add further CSO topics that according to their judgement were missing from the candidate topics.

We created the set of candidate topics, which were displayed by the application, by aggregating the output of three classifiers: the syntactic module (Sect. 4.1), the semantic module (Sect. 4.2), and a third approach, which was introduced for mitigating the bias towards the first two methods.

The third approach first splits the input document into overlapping windows of size 10 (same as the training window of the word2vec model), each of them overlapping by five words. As a second step, it creates the embedding representation of each window by computing the average of the embedding vectors of all its tokens and uses the word2vec model to identify the top 20 similar words with similarity above 0.6. Then, the algorithm returns the CSO concept matching those words. Next, to each CSO concept it assigns a score based i) on the number of times it is found in the list of similar words and ii) on the embedding similarity, i.e. cosine similarity between the vector representation of the window and word embedding. Finally, it sorts them in descending order and prunes the result set using the elbow method [57]. For each paper, the combination of these approaches produced a very inclusive set of 41.8 ± 17.5 candidate topics.

5.1.3 Gold standard

The data collection process produced 210 annotations—70 papers times 3 annotations per paper or 21 experts times 10 annotations. On average, each paper was assigned with 18 ± 9 topics.

We computed the Fleiss' kappa to measure the agreement among the three annotators on each paper. We obtained an average of 0.451 ± 0.177, indicating a moderate inter-rater agreement, according to Landis and Koch [32].

We created the gold standard using the majority rule approach. Specifically, if a topic was considered relevant by at least two annotators, it was added to the gold standard. As a result, on average each paper is associated with 14.4 ± 7.0 topics. In order to consider the taxonomic relationships of CSO, the resulting set of topics were semantically enriched by including also their direct super-topics as in [45].

5.2 Methods for outlier detection

As discussed in Sect. 4.3, the CSO Classifier 2.0 [54] sometimes returns topics that have a very weak connection with all the other topics identified in the document. These outliers are often false positives and tend to have a negative impact on the performance of the classifier.

We thus decided to perform extensive experiments for answering two research questions:

[19] These three fields are well covered by CSO, which includes a total of 35 sub-topics for the Semantic Web, 173 for Natural Language Processing, and 396 for Data Mining.

Q1. Can an outlier detection algorithm improve the performance of an unsupervised approach for topic detection such as the CSO Classifier?

Q2. What kind of methodologies and similarity metrics would yield the best performance?

To reduce the search space, we considered a simple architecture that includes two steps: i) computing the pairwise similarity of topics and ii) detecting uncoupled topics. In order to evaluate different alternatives for these steps, we compared the performance of five similarity metrics and three approaches for detecting disconnected topics. These similarity measures were combined with additional heuristics, described in Sect. 4.3.1, resulting in eight different approaches. In Sect. 5.2.1 we describe the five similarity metrics, in Sect. 5.2.2 the three methods for finding uncoupled topics, and in Sect. 5.2.3 we describe more in detail the eight approaches for detecting outliers.

5.2.1 Similarity metrics

The **Graph** similarity considers only the S_{graph} described in Sect. 4.3.1, computing the distance between topics as the length of their shortest path in the CSO graph. The similarity matrix is obtained by complementing to one the normalised distance matrix.

The **Branch** similarity is also based on CSO. It takes advantage of the hierarchical relationships (see Sect. 3), and for each pair of topics, the branch similarity is computed as the Jaccard similarity between their sets of super topics. The idea is that two topics have high similarity if they share almost the same broader topics.

The **Graph-branch** similarity extends the graph similarity. It computes the length of the shortest path between topics. Additionally, for a given pair of topics, if one is the super topic of the other, then their distance is set to 0 (minimum distance). This is because intuitively a paper about "machine learning" is also a paper in "artificial intelligence" and "computer science". This similarity reflects the inclusion aspect of the CSO hierarchy.

The **Word2vec** similarity considers only the $S_{embeddings}$ matrix shown in Sect. 4.3.1, computing the cosine similarity of topic embeddings.

The **Graph-word2vec** similarity measure produces the similarity matrix as shown in Sect. 4.3.1.

5.2.2 Approaches for finding isolated nodes

In this section, we summarise the three approaches used for identifying isolated topics. First, we used the approach described in Sect. 4.3.1, identified as **MAT**. Then, we also employed the clique percolation method and the hierarchical clustering, which will be described in the following subsections.

Clustering with clique percolation method The clique percolation method (**CPM**) is an algorithm for finding clusters within networks, introduced by Palla et al. [48]. It takes as input a graph data structure, and it returns the list of topics organised within clusters. In this case, the similarity matrix, produced by the different metrics described above, can be seen as a weighted adjacency matrix representing a graph. However, since such weights fall within the range [0, 1], considering all these positive similarities as edges, will lead to a highly interconnected graph hindering the possibility of identifying outliers. To this end, we prune the graph retaining only the top k edges, based on the similarity, where k is a multiple of the number of vertices (topics). The optimal value of k will be object of the evaluation in Sect. 5.5. In brief, the constructed graph will have all topics as nodes and the edges between couple of topics if their similarity is higher than the defined threshold. This approach is equivalent to binarising the similarity matrix described in Sect. 4.3.1. The set of candidate topics are identified as the ones that form a cluster on their own.

As parameter of the algorithm, the dimension of cliques has been set to 3. The Python implementation of CPM is available on GitHub[20].

Hierarchical Clustering Another strategy we used for identifying uncoupled topics is the unweighted pair group method with arithmetic mean developed by Sokal et al. [60], and identified as **HIER**.

This algorithm builds a dendrogram (i.e. rooted tree) that reflects the existing structure within the similarity matrix. This approach starts with each observation in its own cluster, then in a bottom-up fashion it merges pairs of clusters, until it reaches the root where all observations belong to one single cluster. At each iteration, the two most similar clusters are combined into a higher-level cluster. The similarity values of the newly formed cluster are given by averaging the similarity values of the two clusters.

To obtain a good clustering, the algorithm then cuts the dendrogram at a certain level. Since cluster analysis is essentially an exploratory approach, the interpretation of the hierarchical structure depends on the context. To this end, we performed several cuts across the different levels, which will be object of the evaluation in Sect. 5.5.

From the returned clusters, we identify the candidate topics as the ones that create clusters on their own, or in other words the algorithm did not find a suitable merge for them.

[20] Clique Percolation Method developed in Python—https://github.com/angelosalatino/CliquePercolationMethod-Python.

5.2.3 Outlier detection approaches

From the five similarity metrics, we devised eight approaches for identifying outliers. These approaches combine also the heuristics described in Sect. 4.3.1. Specifically, we identify with **multigrams** the heuristic that keeps the syntactic topics that have more than two grams. Instead, we identify with **superTopics** the one that keeps the super topics of the retained group, and finally we identify with **stringSim** the heuristic that keeps the topics that have high string similarity with the retained group.

For these approaches, we also tested the most suitable algorithm for finding the isolated topics. In addition, for each of these algorithms we also tested different pruning thresholds. Specifically, for both removing isolated topics algorithm (described in Sect. 4.3.1) and clique percolation method we observed the optimal threshold for binarising the similarity matrix. This threshold identifies the top *k* similarity values where *k* is a multiple of the number of topics. The lower is *k*, the fewer *ones* will be available in the similarity matrix, resulting in a restrictive threshold. On the other hand, the higher is *k*, more *ones* will appear in the similarity matrix, resulting in a conservative approach. Indeed, when *k* is higher, there are fewer and fewer excluded topics and the results of the classifier converge to the ones presented in [54].

Instead, for the hierarchical clustering we identified the optimal threshold for cutting the dendrogram and therefore identifying the clusters of cohesive groups of topics. This threshold identifies the level where the cut is performed, starting from top. If *k* is low, it means that the cut is near to the root node where almost all topics will belong to a few large clusters. Instead, if *k* is higher, it means that the cut is performed closer to the leaf nodes and thus it will return several small clusters.

For each of the eight outlier detection approaches, we performed a grid search to determine the best configuration in term of the algorithm for finding uncoupled topics (CPM, MAT, or HIER) and the pruning threshold (1–5) by evaluating them as a component of the CSO Classifier against the gold standard described in Sect. 5.1. Table 5 summarises the settings which obtained the best f-measure for each approach. For instance, the approach GRA-S obtained the best results when using a threshold of 5 and the CPM algorithm. We report and discuss their performance in Sect. 5.4.

5.3 Experimental set-up

We compared 23 alternative methods and evaluated their results against the gold standard. Table 6 describes their main features and reports their performance.

TF-IDF returns for each paper a ranked list of words according to their term frequency-inverse document frequency (TF-IDF) score. The inverse document frequency of the terms was computed on the dataset of 4.6M papers in Computer Science, introduced in Sect. 4.2.1. **TF-IDF-M** maps these terms to CSO by returning all the CSO topics having Levenshtein similarity higher than 0.8 with them.

The next six classifiers use the latent Dirichlet allocation (LDA) [7] in order to produce a number of keywords extracted from the distribution of terms associated with the LDA topics. We trained three versions of LDA over the same corpus with different numbers of topics, i.e. 100, 500, and 1000, respectively, **LDA100**, **LDA500**, and **LDA1000**. These three classifiers select all LDA topics with a probability of at least *j* and return all their words with a probability of at least *k*. **LDA100-M**, **LDA500-M**, and **LDA1000-M** work in the same way, but the resulting keywords are then mapped to the CSO topics. In particular, they return all CSO topics that have Levenshtein similarity higher than 0.8 with the resulting set of terms. We performed a grid search for finding the best values of *j* and *k* on the gold standard and report here the best results of each classifier in terms of f-measure.

W2V-W is the classifier described in Sect. 5.1.2, designed to produce further candidate topics for the domain experts. It processes the input document with a sliding window and uses the word2vec model to identify concepts semantically similar to the embedding of the window.

STM is the classifier originally adopted by Smart Topic Miner [45], the application used by Springer Nature for classifying proceeding in the field of Computer Science. It works similarly to the syntactic module described in Sect. 4.1, but it detects only exact matches between the terms extracted from the text and the CSO topic labels. **SYN** is the first version of the CSO classifier, firstly introduced in [50], and it is equivalent to the syntactic module as described in Sect. 4.1. **SEM** consists of the semantic module described in Sect. 4.2. **INT** is a hybrid version that returns the intersection of the topics produced by the syntactic (**SYN**) and semantic (**SEM**) modules. Finally, **CC 2.0** is the implementation of the CSO Classifier v2.0 presented in [54]. As described in Sect. 4.3, this version produces the union of the topics returned by the two modules, but it does not include the outlier detection component, yet.

The remaining nine classifiers provide alternative approaches to extending CC 2.0, to produce a new version of the CSO Classifier.

CC+RAND is a very simple baseline that randomly removes 10% of the topics in the set. The other eight classifiers use the eight methods reported in Table 5 and are labelled as **CC+CODE**, where **CC** stands for CSO Classifier and *CODE* refers to the outlier detection approach described in Sect. 5.2.3, e.g. **GRA-S** for graph similarity.

We assessed the performance of these 23 approaches by means of precision, recall, and f-measure. When classifying a given paper *p*, the value of precision *pr(p)* and recall *re(p)*

Table 5 The eight approaches for detecting outlier topics according to their similarity measures, optimal thresholds (THR), optimal algorithms (Algor.), and heuristics

Outlier det. met.	Similarity measure	THR	Best Algor.	Additional heuristics
GRA-S	Graph	5	CPM	–
BRA-S	Branch	4	MAT	–
GBR-S	Graph-branch	5	CPM	–
W2V-S	Word2vec	3	MAT	–
GWV-S	Graph-word2vec	2	HIER	–
GWG-S	Graph-word2vec	1	MAT	multigrams
GWP-S	Graph-word2vec	1	MAT	multigrams + superTopics
GWS-S	Graph-word2vec	1	MAT	multigrams + superTopics + stringSimilarity

are computed as shown in Eq. 2:

$$\mathrm{pr}(p) = \frac{|cl(p) \cap gs(p)|}{|cl(p)|} \quad \mathrm{re}(p) = \frac{|cl(p) \cap gs(p)|}{|gs(p)|} \qquad (2)$$

where *cl(p)* identifies the topics returned by the classifier, and *gs(p)* the gold standard obtained for that paper, including the super-areas of the gold standard used to enrich the user annotations as mentioned in Sect. 5.1.3. In order to obtain a better comparison between the different classifiers, we enhanced the results of each method with their direct super-concepts. The overall precision and recall for a given classifier are computed as the average of the values of precision and recall obtained over the papers. The f-measure (F1) is the harmonic mean of precision and recall.

5.4 Results

Table 6 reports the values of precision, recall, and f-measure of the different classifiers. The upper panel shows the results of the 14 approaches discussed in [54], while the lower panel summarises the results of the nine new versions of the CSO Classifier.

The approaches based on LDA and TF-IDF performed poorly and did not exceed 30.1% of f-measure. Arguably, we could raise the precision by increasing the Levenshtein similarity threshold for matching terms with CSO topics; however, this drastically reduces the recall making these approaches mostly unfit for this task. An analysis on the LDA topics showed that these tend to be broad and noisy. Indeed, they cluster together distinct CSO topics (e.g. “databases” and “search engines”) in the same LDA topic. In a nutshell, LDA performs quite well in identifying broader topics characterising large collection of documents, but as discussed in Section 2.1 it is typically less suitable in inferring more specific research topics, which may be associated with a low number of publications (50–200). **W2V-W** performed also poorly in terms of both precision and recall, respectively, 41.2% and 16.7%.

STM and **SYN** yielded a very good precision of, respectively, 80.8% and 78.3%. Indeed, these methods are good at finding topics that get explicitly mentioned in the text, which tend to be very relevant. However, their low recall, respectively, of 58.2% and 63.8%, is because they fail to detect some more subtle topics that are just implied. The difference of performance between these two classifiers is due to the method used to map n-grams to CSO topics, because **STM** identifies only equal matches, while **SYN** finds also partial matches, thus increasing recall at the expense of precision.

Compared with **SYN**, the semantic module (**SEM**) lost some precision but gained in recall and f-measure. This suggests that it is able to identify further topics that are not explicitly available in the paper, but clearly this may also produce some more false positives. **INT** yielded a higher precision (79.3%) compared to the syntactic and the semantic modules (78.3% and 70.8%), but it did not perform well in terms of recall, which dropped from 63.8% and 72.2% to 59.1%.

Finally, the CSO Classifier v2.0 (**CC 2.0**), presented in [54], outperformed all previous methods in terms of both recall and f-measure, respectively, 75.3% and 74.1%.

The following nine classifiers represent the contribution introduced with this manuscript. The first solution (**CC+RAND**) randomly removes 10% of the identified topics for each paper. Compared with **CC 2.0** [54], it returns a slightly higher precision (73.1%), but the recall (67.6%) and f-measure (70.2%) are dramatically reduced.

The classifiers **CC+GRA-S**, **CC+BRA-S**, **CC+GBR-S**, and **CC+W2V-S** produce comparable results with f-measure around 74.4%. In particular, their precision values are higher than **CC 2.0**, ranging between [73.4%, 73.9%], as well as recall values that range between [74.9%, 75.3%].

CC+GWV-S combines **CC+GRA-S** and **CC+W2V-S**, and produces higher values of precision (75.6%), but its outlier detection method is very aggressive impacting the recall (73.9%). Both **CC+GWG-S** and **CC+GWP-S** extend **CC+GWV-S**, and in comparison with the latter, they improve

Table 6 Values of precision, recall, and f-measure for the different classifiers. In bold are the best results for the two sections

Classifier	Description	Prec. (%)	Rec. (%)	F1 (%)
TF-IDF	TF-IDF.	16.7	24.0	19.7
TF-IDF-M	TF-IDF mapped to CSO concepts	40.4	24.1	30.1
LDA100	LDA with 100 topics	5.9	11.9	7.9
LDA500	LDA with 500 topics	4.2	12.5	6.3
LDA1000	LDA with 1000 topics	3.8	5.0	4.3
LDA100-M	LDA with 100 topics mapped to CSO	9.4	19.3	12.6
LDA500-M	LDA with 500 topics mapped to CSO	9.6	21.2	13.2
LDA1000-M	LDA with 1000 topics mapped to CSO	12.0	11.5	11.7
W2V-W	W2V on windows of words (Sect. 5.1.2)	41.2	16.7	23.8
STM	Classifier used by STM, introduced in [45]	**80.8**	58.2	67.6
SYN	Syntactic module (Sect. 4.1) [50]	78.3	63.8	70.3
SEM	Semantic module (Sect. 4.2)	70.8	72.2	71.5
INT	Intersection of SYN and SEM	79.3	59.1	67.7
CC 2.0	The CSO Classifier v2.0 [54]	73.0	**75.3**	**74.1**
CC+RAND	Random selection of outliers	73.1	67.6	70.2
CC+GRA-S	Graph similarity	73.4	**75.3**	74.4
CC+BRA-S	Branch similarity	73.9	74.9	74.4
CC+GBR-S	Graph-Branch similarity	73.4	**75.3**	74.4
CC+W2V-S	Word2vec similarity	73.9	**75.3**	74.6
CC+GWV-S	Graph-word2vec similarity	75.6	73.9	74.7
CC+GWG-S	Graph-word2vec similarity; multigrams	**77.7**	73.1	75.3
CC+GWP-S	Graph-word2vec similarity; multigrams; superTopics	77.5	73.2	75.3
CC+GWS-S	Graph-word2vec similarity; multigrams; superTopics; syntactic similarity	76.7	74.0	**75.4**

the overall f-measure to 75.3%. These two versions of the CSO Classifier produce very high values of precision, respectively, 77.7% and 77.5%; however, their recall values go as low as 73.1%.

Finally, **CC+GWS-S** outperforms the other classifiers as it presents a f-measure of 75.4% providing a more balanced values of precision (76.7%) and recall (74.0%) compared to **CC+GWG-S** and **CC+GWP-S**. For this reason, we selected this configuration as the new version of the CSO Classifier (v3.0).

In order to assess the difference between CSO Classifier 2.0 [54] (**CC 2.0**) and CSO Classifier 3.0 (**CC+GWS-S**), we used the Wilcoxon nonparametric test. When considering the full results reported in Table 6, the two approaches are statistically different regarding precision ($p < 0.0001$), but not f-measure ($p = 0.1500$). However, this is due to the fact that the outlier detection component only activates in about half of the cases, i.e. when CSO Classifier 2.0 returns a topic that is semantically inconsistent with the others. When considering only the 34 articles (out of 70) in which the outlier detection component activated, the effect of the new approach becomes very evident: **CC+GWS-S** gains about 8% precision over **CC 2.0** (77.0% vs 69.3%) losing only 4% recall (76.7% vs 80.7%). When running the Wilcoxon test on this set, CSO Classifier 3.0 significantly outperforms CSO Classifier 2.0 in both precision ($p < 0.0001$) and f-measure ($p < 0.0001$).

5.5 Evaluating outliers detection techniques

In this section, we focus on the different techniques which were developed for identifying the topic outliers. We can see them as an operation of selection that, given the set of topics returned by the CSO Classifier, needs to identify and exclude the false positive, hopefully improving the precision of the classifier. We can then compute precision, recall, and f-measure of this selection operation to study their performance. Naturally, we cannot assume that all the false positives are outlier topics; therefore, the recall will be capped according to the percentage of outliers in the set of false positive.

In this context, the true positives (TP) are the topics that were selected to be excluded but are not present in the gold standard, the false positives (FP) are the topics that were selected but are in the gold standard, and the false negatives (FN) are the topics that were not selected but are not in the gold standard.

Table 7 reports the best values of precision, recall, and f-measure for all the similarity measures, alongside the values of f-measure of the CSO Classifier when such similarity measure in the outlier detection component is applied (last column).

The table shows that **GRA-S** and **GBR-S**, respectively, adopting the graph similarity and the hybrid graph and branch similarity, obtain a very high precision (88.9%). However, these approaches lack of ambition as they tend to identify just a few outliers.

The similarity measure that performed the best in terms of f-measure (38%) is **GWG-S**, which has 70.6% of precision and 26% of recall. However, for this task precision is more important than recall. **GWS-S** is the clear winner in terms of precision (75.0%), outperforming significantly ($p < 0.0001$ according to the Wilcoxon test) all the approaches which obtained a decent recall ($> 3.5\%$). For this reason, we adopted **GWS-S** for the CSO Classifier 3.0.

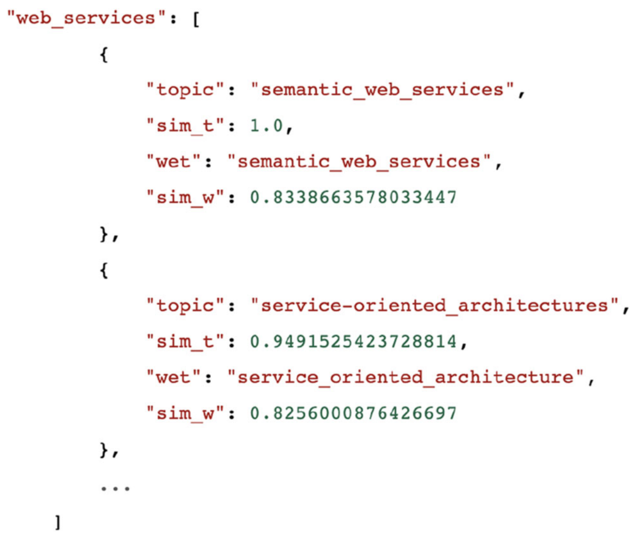
The fact that the best approaches for outlier detection obtained a recall around 20–25% suggests that about the same percentage of false positive produced by the CSO Classifier could be ascribed to outliers. However, we need further experiments to reach a definitive conclusion.

Answering the questions posed in Sect. 5.2, these results provide evidence that an outlier detection algorithm can indeed improve the performance of an unsupervised approach like the CSO Classifier.

Furthermore, the outlier detection approaches taking advantage of both the topological structure of CSO and word embedding model appears to perform best.

6 Improving classification scalability

A recurring problem of the early adopters of the CSO Classifier was its scalability. For instance, the version of the CSO Classifier presented in the previous section, **CC-GWS-S**, requires about 2.6 seconds, on average, to process one paper[21]. If we wish to integrate this classifier in the Smart Topic Miner [51], an application that classifies conference proceedings, which may contain hundreds of papers, this implies a time performance of about 4 and a half minutes for each batch of 100 papers. In addition, we estimated that classifying all Computer Science papers in Microsoft Academic Graph would require around 2 months, assuming such a computation is carried out by ten parallel processes. By inspecting the code and timing the several components of the classifier, we identified the major bottlenecks. In particular, we found two components that were requiring excessive amount of time. The first one concerns the code in the syntactic module, which compares the n-grams against the list of topics. The second bottleneck concerns the code in the semantic module, which retrieves the most similar words of a given n-gram from the word2vec model.



```
"web_services": [
        {
            "topic": "semantic_web_services",
            "sim_t": 1.0,
            "wet": "semantic_web_services",
            "sim_w": 0.8338663578033447
        },
        {
            "topic": "service-oriented_architectures",
            "sim_t": 0.9491525423728814,
            "wet": "service_oriented_architecture",
            "sim_w": 0.8256000876426697
        },
        ...
    ]
```

Fig. 4 An excerpt of the cached word2vec model

To patch the first problem, we assumed that the similarity between an n-gram and a research topic is higher if they share the initial four characters. To this end, we developed a topic-stems dictionary in which CSO topics are grouped together if they share the first four characters. In such case, the syntactic module instead of comparing the n-grams against all topics in the ontology will just identify the group of topics that share its stem (the first four characters) and compare them with a limited number of topics. For instance, a n-gram having “digi” as initial four characters will be compared only against 70 topics, such as *digital libraries*, *digital signal processing*, *digital-to-analog converters*, and *digital rights management*, instead of the whole set of 14K topics.

To mitigate the second bottleneck, we created a cached model. Specifically, this model connects words of the word2vec model vocabulary straight to the topics in the Computer Science Ontology. To create this cache, we iterated over the model vocabulary, and, for each word, we extracted its top ten similar words (considering only terms with a cosine similarity higher than 0.7). Then, for each similar word, we computed the Levenshtein similarity against all CSO topics. All topics matching with a similarity higher or equal to 0.94 are linked to the selected word of the vocabulary.

In Fig. 4, there is an excerpt of the designed cache, where, for example, the word “web_services” from the vocabulary model has 0.82 (sim_w) similarity with the word “service_oriented_architecture” (wet = word embedding token), that matches with the topic *service-oriented_architectures* with Levenshtein similarity of 0.949 (sim_t).

[21] This figure has been estimated on a MacBook Pro 14.2, 3.5 GHz Intel Core i7, 16 GB 2133 MHz LPDDR3.

Table 7 Values of precision, recall, and f-measure for the outlier detection techniques. In bold are the best results. In the last column, we report also the values of f-measure of the CSO Classifier, when such technique is applied

Outlier detection method	Outliers performance			Class.
	Prec. (%)	Rec. (%)	F1 (%)	F1 (%)
RAND	33.0	16.0	21.6	70.4
GRA-S	**88.9**	3.5	6.7	74.4
BRA-S	72.7	6.9	12.6	74.4
GBR-S	**88.9**	3.5	6.7	74.4
W2V-S	72.0	7.8	14.1	74.6
GWV-S	60.6	8.7	15.2	74.7
GWG-S	70.6	**26.0**	**38.0**	75.3
GWP-S	69.9	25.1	36.9	75.3
GWS-S (in CC 3.0)	75.0	20.8	32.5	**75.4**

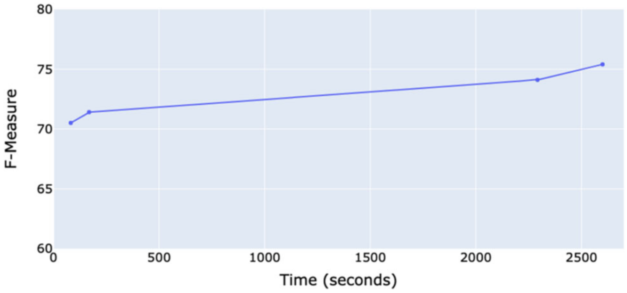


Fig. 5 The variation of f-measure based on computational time

In order to assess the effect of these improvements, we performed a scalability analysis of the different versions of the classifier. Specifically, we run CSO Classifier 2.0 (**CC 2.0**) and the CSO Classifier (**CC 3.0**) with and without cache on a sample of 1000 papers. In Table 8, we report the computational times expressed in terms of i) the total number of seconds for classifying the set of papers, ii) papers per minute, iii) papers per second, and iv) gained speed. The gained speed is relative to **CC 3.0**, being the slowest classifier. To facilitate the reader in comparing the trade-off between speed and loss of accuracy, we also report the corresponding values of precision, recall, and f-measure when evaluated against the gold standard.

We can observe that when switching from **CC 3.0** to **CC 3.0 Cached**, we lose 4% f-measure, but the classifier is more than 15x faster. When switching from **CC 3.0** to **CC 2.0 Cached**, we lose less than 5% f-measure, but we gain a speed up to 31x. In Fig. 5, we can better appreciate the trade-off between computational time and f-measure.

The CSO classifier 3.0 offers two flags that allow the users to choose the best trade-off between precision and speed. The flag *delete_outliers* = *True* enables the outliers module (CC 3.0), whereas *fast_classification* = *True* enables the cache.

7 Employing the CSO Classifier in other domains of Science

The architecture of the CSO Classifier makes it possible to apply it to other scientific domains, as long as a comprehensive ontology or taxonomy of research areas is available for the domain in question. In this section, we illustrate the methodology required to use the CSO Classifier in other scientific domains.

7.1 Ontology or taxonomy of research topics

The Computer Science Ontology is a crucial component for the CSO Classifier, as it provides the list of possible topics to associate to documents. However, as CSO only covers *Computer Science*, when applying the CSO Classifier to another scientific field, the first step requires replacing CSO with an alternative taxonomy or ontology of research topics. In particular, a good replacement ought to provide the following relationships:

– **superTopicOf**, which provides hierarchical information and shows how research topics are distributed, from the most generic to the most specific ones;
– **relatedEquivalent**, which identifies synonyms of a research concept;
– **label**, which provides the possible lexical materialisations of a research topic in a document.

An optional relation is **primaryLabel**, which defines which of the labels of a concept to use by default. This is advisable since it makes the resulting annotations easier to read and analyse.

7.2 Word embedding model

The advantage of using a word embedding model within the CSO Classifier is that it enables to capture the semantics

Table 8 Computational time of the different versions of the classifier both with and without cache

	Time for 1000 papers (s)	Papers per minute	Papers per second	Speed	Prec. (%)	Rec. (%)	F1 (%)
CC 2.0	2,293.34	26.16	0.44	1.13×	73.0	75.3	74.1
CC 2.0 Cached	82.39	728.24	12.14	31.53×	66.3	75.3	70.5
CC 3.0	2,597.62	23.10	0.38	1.00×	76.7	74.0	75.4
CC 3.0 Cached	169.59	353.79	5.90	15.32×	70.0	73.0	71.4

of words. However, such semantics highly depends on the domain of application, i.e. *Computer Science*, and how they are used within the language of that field.

In order to apply the classifier to another domain, it is crucial to train the word2vec model with another corpus of research papers that fits the new domain, so to be able to capture the semantic of the words in that particular domain. Moreover, it is also important to evaluate the number n of most occurring words in the vocabulary, as described in Sect. 4.3, so that the semantic module can avoid inferring generic terms.

As shown in Sect. 4.2.1, the word2vec model was trained using titles and abstracts from Microsoft Academic Graph in the field of *Computer Science*. Since MAG covers also other areas of Science, it can be a suitable resource for training the new model. However, depending on the scientific field in question, there can be other available sources that can be used to train the word2vec model. In particular, we are currently working on transferring the CSO Classifier to the field of *Medicine*, and we are training the model using PubMed[22].

8 Applications that use the CSO classifier

Since the introduction of the CSO Classifier in 2019, several research papers and surveys [63] have cited it, emphasising its value. In addition, a number of researchers have embedded the classifier in concrete applications.

Dörpinghaus et al. [22] developed a scientific knowledge graph that integrates bibliographic data and metadata from Pubmed and DBLP. Since Pubmed data was already annotated, they extracted topics from DBLP using the CSO Classifier. Then, the authors use this knowledge graph to generate graph embeddings applied on tasks such as topic detection, document clustering, and knowledge discovery.

Vergoulis et al. [67] used the CSO Classifier for classifying 1.5M papers and use such topical representation for identifying experts that share similar publishing habits. This exercise can unfold in various real-life applications such as reviewer recommendation, collaborator seeking, and new hire recommendation. The same research team used the CSO Classifier on a corpus of over 3M papers. This was then used to develop ArtSim [14], an approach that estimates the popularity of papers in their cold-start period. This work is based on the intuition that similar papers are likely to follow a similar trajectory in terms of popularity. The authors calculate paper similarities using metapath analyses on scholarly knowledge graphs, which provide better results that use instead citation-based measures.

Jose et al. [28] developed an ontology-based framework that integrates CSO and the CSO Classifier for retrieving specific journal articles from academic repositories. This framework also aims at dynamically expanding the existing Computer Science Ontology with new specialisations, by analysing recently published research papers. In this way, the academic repositories reflect such recently introduced specialties; hence, they support the retrieval of more accurate results.

In addition, the CSO Classifier is currently integrated within the Smart Topic Miner [54], an application that assists the Springer Nature editorial team in annotating the volumes of all books covering conference proceedings in Computer Science. STM uses the CSO Classifier to annotate each paper with the topics from CSO. Then it groups and ranks the topics according to the number of papers addressing them. A demo of STM is available at http://stm-demo.kmi.open.ac.uk/.

The CSO Classifier has also been used to generate the Academia/Industry DynAmics (AIDA) Knowledge Graph [2], which characterises 14M papers and 8M patents according to the research topics drawn from the Computer Science Ontology. We used this dataset to develop the AIDA Dashboard, a tool for exploring and making sense of scientific conferences [3].

9 Conclusions

In this paper, we introduced an improved version of the CSO Classifier, v3.0, which takes advantage of a new approach for detecting outliers, thus producing a more accurate classification of research documents. This solution was evaluated on a gold standard of 70 manually annotated documents and shown to outperform alternative approaches in terms of recall and f-measure. The code of the CSO Classifier and all the

[22] Pubmed—https://pubmed.ncbi.nlm.nih.gov.

relevant material is freely available to the wider research community.

This work opens up several interesting research directions. We plan to test BERT [20], SciBERT [5], and similar modern embeddings, to try and enhance the semantic and post-processing modules. We also intend to explore the application of our approach to other research fields. In particular, as already mentioned, we are working on adapting the CSO Classifier to the Biomedical domain, with the aim of classifying PubMed documents according to concepts drawn from the Medical Subject Headings.

References

1. Alam, M., Biswas, R., Chen, Y., Dessì, D., Gesese, G.A., Hoppe, F., Sack, H.: Hierclassart: knowledge-aware hierarchical classification of scholarly articles. In: Companion Proceedings of the Web Conference (2021)
2. Angioni, S., Salatino, A., Osborne, F., Recupero, D.R., Motta, E.: The aida dashboard: Analysing conferences with semantic technologies. In: 19th International Semantic Web Conference (ISWC 2020) (2020). http://oro.open.ac.uk/72293/
3. Angioni, S., Salatino, A.A., Osborne, F., Recupero, D.R., Motta, E.: Integrating knowledge graphs for analysing academia and industry dynamics. 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. (eds.) ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium, pp. 219–225. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-55814-7_18
4. Beck, M., Rizvi, S.T.R., Dengel, A., Ahmed, S.: From automatic keyword detection to ontology-based topic modeling. In: International Workshop on Document Analysis Systems, pp. 451–465. Springer (2020). https://doi.org/10.1007/978-3-030-57058-3_32
5. Beltagy, I., Lo, K., Cohan, A.: SciBERT: A pretrained language model for scientific text. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3615–3620. Association for Computational Linguistics, Hong Kong, China (2019). https://doi.org/10.18653/v1/D19-1371
6. Blei, D.M., Lafferty, J.D., et al.: A correlated topic model of science. Ann. Appl. Stat. **1**(1), 17–35 (2007)
7. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
8. Bolelli, L., Ertekin, Ş., Giles, C.L.: Topic and trend detection in text collections using latent dirichlet allocation. In: Boughanem, M., Berrut, C., Mothe, J., Soule-Dupuy, C. (eds.) Advances in Information Retrieval, pp. 776–780. Springer, Berlin (2009)
9. Borges, M.V.M., dos Reis, J.C.: Semantic-enhanced recommendation of video lectures. In: 2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT), vol. 2161, pp. 42–46. IEEE (2019). https://doi.org/10.1109/ICALT.2019.00013
10. Boyack, K.W., Klavans, R.: Creation of a highly detailed, dynamic, global model and map of science. J. Assoc. Inf. Sci. Technol. **65**(4), 670–685 (2014). https://doi.org/10.1002/asi.22990
11. Cano-Basave, A.E., Osborne, F., Salatino, A.A.: Ontology forecasting in scientific literature: Semantic concepts prediction based on innovation-adoption priors. In: Blomqvist, E., Ciancarini, P., Poggi, F., Vitali, F. (eds.) Knowledge Engineering and Knowledge Management, pp. 51–67. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-49004-5_4
12. Caragea, C., Bulgarov, F., Mihalcea, R.: Co-training for topic classification of scholarly data. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 2357–2366 (2015)
13. Chatzopoulos, S., Vergoulis, T., Kanellos, I., Dalamagas, T., Tryfonopoulos, C.: Artsim: improved estimation of current impact for recent articles. In: ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium, pp. 323–334. Springer (2020). https://doi.org/10.1007/978-3-030-55814-7_27
14. 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. (eds.) ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium, pp. 323–334. Springer, Cham (2020)
15. Chernyak, E.: An approach to the problem of annotation of research publications. In: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM '15, p. 429–434. Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2684822.2697032
16. Decker, S.L.: Detection of bursty and emerging trends towards identification of researchers at the early stage of trends (2007)
17. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. J. Am. Soc. Inf. Sci. **41**(6), 391–407 (1990). https://doi.org/10.1002/(SICI)1097-4571(199009)41:6%3c391::AID-ASI1%3e3.0.CO;2-9
18. Dessì, D., Osborne, F., Recupero, D.R., Buscaldi, D., Motta, E.: Generating knowledge graphs by employing natural language processing and machine learning techniques within the scholarly domain. Future Generat. Comput. Syst. **116**, 253–264 (2021). https://doi.org/10.1016/j.future.2020.10.026
19. Dessì, D., Osborne, F., Recupero, D.R., Buscaldi, D., Motta, E., Sack, H.: Ai-kg: an automatically generated knowledge graph of artificial intelligence. In: International Semantic Web Conference, pp. 127–143. Springer (2020). https://doi.org/10.1007/978-3-030-62466-8_9
20. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (2019). https://doi.org/10.18653/v1/N19-1423
21. Duvvuru, A., Radhakrishnan, S., More, D., Kamarthi, S., Sultornsanee, S.: Analyzing structural & temporal characteristics of keyword system in academic research articles. Procedia Computer

Science **20**, 439–445 (2013). https://doi.org/10.1016/j.procs.2013.09.300.https://www.sciencedirect.com/science/article/pii/S1877050913011009. Complex Adaptive Systems

22. Dörpinghaus, J., Jacobs, M.: Knowledge detection and discovery using semantic graph embeddings on large knowledge graphs generated on text mining results. In: 2020 15th Conference on Computer Science and Information Systems (FedCSIS), pp. 169–178 (2020). https://doi.org/10.15439/2020F36
23. Garcia-Silva, A., Gomez-Perez, J.M.: Classifying scientific publications with bert – is self-attention a feature selection method? (2021)
24. Griffiths, T.L., Steyvers, M.: Finding scientific topics. Proceedings of the National Academy of Sciences **101**(suppl 1), 5228–5235 (2004). https://doi.org/10.1073/pnas.0307752101
25. Hamidah, I., Sriyono, S., Hudha, M.N.: A bibliometric analysis of Covid-19 research using vosviewer. Indonesian J. Sci. Technol. **5**(2), 209–216 (2020)
26. Hofmann, T.: Probabilistic latent semantic indexing. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '99, p. 50–57. Association for Computing Machinery, New York, NY, USA (1999). https://doi.org/10.1145/312624.312649
27. Jo, Y., Lagoze, C., Giles, C.L.: Detecting research topics via the correlation between graphs and texts. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '07, p. 370–379. Association for Computing Machinery, New York, NY, USA (2007). https://doi.org/10.1145/1281192.1281234
28. Jose, V., Jagathy Raj, V.P., George, S.K.: Ontology-based information extraction framework for academic knowledge repository. In: X.S. Yang, S. Sherratt, N. Dey, A. Joshi (eds.) Proceedings of Fifth International Congress on Information and Communication Technology, pp. 73–80. Springer Singapore, Singapore (2021)
29. Kandimalla, B., Rohatgi, S., Wu, J., Giles, C.L.: Large scale subject category classification of scholarly papers with deep attentive neural networks. Front. Res. Metrics Anal. **5**, 31 (2021). https://doi.org/10.3389/frma.2020.600382
30. Karpathy, A., Fei-Fei, L.: Deep visual-semantic alignments for generating image descriptions. IEEE Trans. Pattern Anal. Mach. Intell. **39**(4), 664–676 (2017). https://doi.org/10.1109/TPAMI.2016.2598339
31. Lafferty, J., Blei, D.: Correlated topic models. In: Y. Weiss, B. Schölkopf, J. Platt (eds.) Advances in Neural Information Processing Systems, vol. 18, pp. 147–154. MIT Press (2006). https://proceedings.neurips.cc/paper/2005/file/9e82757e9a1c12cb710ad680db11f6f1-Paper.pdf
32. Landis, J.R., Koch, G.G.: The measurement of observer agreement for categorical data. Biometrics **33**(1), 159–174 (1977). http://www.jstor.org/stable/2529310
33. Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C.H., Kang, J.: BioBERT: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics (2019). https://doi.org/10.1093/bioinformatics/btz682
34. Lilleberg, J., Zhu, Y., Zhang, Y.: Support vector machines and word2vec for text classification with semantic features. In: 2015 IEEE 14th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC), pp. 136–140 (2015). https://doi.org/10.1109/ICCI-CC.2015.7259377
35. Löffler, F., Wesp, V., Babalou, S., Kahn, P., Lachmann, R., Sateli, B., Witte, R., König-Ries, B.: Scholarlensviz: A visualization framework for transparency in semantic user profiles. In: Taylor, K., Gonçalves, R., Lecue, F., Yan, J. (eds.) Proceedings of the ISWC 2020 Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 19th International Semantic Web Conference (ISWC 2020), Globally online, November 1-6, 2020 (UTC). (2020)
36. Mai, F., Galke, L., Scherp, A.: Using deep learning for title-based semantic subject indexing to reach competitive performance to full-text. In: Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries, JCDL '18, p. 169–178. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3197026.3197039
37. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space (2013)
38. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality (2013)
39. Nayyeri, M., Cil, G.M., Vahdati, S., Osborne, F., Rahman, M., Angioni, S., Salatino, A., Recupero, D.R., Vassilyeva, N., Motta, E., Lehmann, J.: Trans4e: link prediction on scholarly knowledge graphs. Neurocomputing (2021). https://doi.org/10.1016/j.neucom.2021.02.100
40. Osborne, F., De Ribaupierre, H., Motta, E.: Techminer: extracting technologies from academic publications. In: European Knowledge Acquisition Workshop, pp. 463–479. Springer (2016). https://doi.org/10.1007/978-3-319-49004-5_30
41. Osborne, F., Mannocci, A., Motta, E.: Forecasting the spreading of technologies in research communities. In: Proceedings of the Knowledge Capture Conference, K-CAP 2017. Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3148011.3148030
42. Osborne, F., Motta, E.: Klink-2: Integrating multiple web sources to generate semantic topic networks. In: Arenas, M., Corcho, O., Simperl, E., Strohmaier, M., d'Aquin, M., Srinivas, K., Groth, P., Dumontier, M., Heflin, J., Thirunarayan, K., Thirunarayan, K., Staab, S. (eds.) The Semantic Web - ISWC 2015, pp. 408–424. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-25007-6_24
43. Osborne, F., Motta, E.: Pragmatic ontology evolution: reconciling user requirements and application performance. In: International Semantic Web Conference, pp. 495–512. Springer (2018). https://doi.org/10.1007/978-3-030-00671-6_29
44. Osborne, F., Motta, E., Mulholland, P.: Exploring scholarly data with rexplore. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) The Semantic Web – ISWC 2013, pp. 460–477. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41335-3_29
45. Osborne, F., Salatino, A., Birukou, A., Motta, E.: Automatic classification of springer nature proceedings with smart topic miner. In: Groth, P., Simperl, E., Gray, A., Sabou, M., Krötzsch, M., Lecue, F., Flöck, F., Gil, Y. (eds.) The Semantic Web—ISWC 2016, pp. 383–399. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-46547-0_33
46. Osborne, F., Scavo, G., Motta, E.: A hybrid semantic approach to building dynamic maps of research communities. In: International Conference on Knowledge Engineering and Knowledge Management, pp. 356–372. Springer (2014). https://doi.org/10.1007/978-3-319-13704-9_28
47. 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. (eds.) The Semantic Web: Trends and Challenges, pp. 114–129. Springer International Publishing, Cham (2014). https://doi.org/10.1007/978-3-319-07443-6_9
48. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature **435**(7043), 814–818 (2005). https://doi.org/10.1038/nature03607
49. Rossanez, A., dos Reis, J.C., da Silva Torres, R.: Representing scientific literature evolution via temporal knowledge graphs (2020)

50. Salatino, A., Thanapalasingam, T., Mannocci, A., Osborne, F., Motta, E.: Classifying research papers with the computer science ontology. In: M. van Erp (ed.) ISWC 2018 Posters & Demonstrations and Industry Tracks (2018). http://oro.open.ac.uk/55908/
51. Salatino, A.A., Osborne, F., Birukou, A., Motta, E.: Improving editorial workflow and metadata quality at springer nature. In: Ghidini, C., Hartig, O., Maleshkova, M., Svátek, V., Cruz, I., Hogan, A., Song, J., Lefrançois, M., Gandon, F. (eds.) The Semantic Web—ISWC 2019, pp. 507–525. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-30796-7_31
52. Salatino, A.A., Osborne, F., Motta, E.: Augur: Forecasting the emergence of new research topics. In: Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries, JCDL '18, p. 303–312. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3197026.3197052
53. Salatino, A.A., Osborne, F., Motta, E.: Ontology extraction and usage in the scholarly knowledge domain (2020). arXiv:2003.12611
54. Salatino, A.A., Osborne, F., Thanapalasingam, T., Motta, E.: The cso classifier: Ontology-driven detection of research topics in scholarly articles. In: Doucet, A., Isaac, A., Golub, K., Aalberg, T., Jatowt, A. (eds.) Digital Libraries for Open Knowledge, pp. 296–311. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-30760-8_26
55. Salatino, A.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. **2**(3), 379–416 (2020). https://doi.org/10.1162/dint_a_00055
56. Salatino, A.A., Thanapalasingam, T., Mannocci, A., Osborne, F., Motta, E.: The computer science ontology: A large-scale taxonomy of research areas. In: Vrandečić, D., Bontcheva, K., Suárez-Figueroa, M.C., Presutti, V., Celino, I., Sabou, M., Kaffee, L.A., Simperl, E. (eds.) The Semantic Web – ISWC 2018, pp. 187–205. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-030-00668-6_12
57. Satopaa, V., Albrecht, J., Irwin, D., Raghavan, B.: Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In: 2011 31st International Conference on Distributed Computing Systems Workshops, pp. 166–171 (2011). https://doi.org/10.1109/ICDCSW.2011.20
58. Shen, Z., Ma, H., Wang, K.: A web-scale system for scientific knowledge exploration. arXiv preprint arXiv:1805.12216 (2018)
59. Small, H., Boyack, K.W., Klavans, R.: Identifying emerging topics in science and technology. Res. Policy **43**(8), 1450–1467 (2014). https://doi.org/10.1016/j.respol.2014.02.005
60. Sokal, R., Michener, C.D.: A statistical method for evaluating systematic relationships. Univ. Kansas Sci. Bull. **38**, 1409–1438 (1958)
61. Song, Y., Roth, D.: Unsupervised sparse vector densification for short text similarity. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1275–1280. Association for Computational Linguistics, Denver, Colorado (2015). https://doi.org/10.3115/v1/N15-1138
62. Thanapalasingam, T., Osborne, F., Birukou, A., Motta, E.: Ontology-based recommendation of editorial products. In: Vrandečić, D., Bontcheva, K., Suárez-Figueroa, M.C., Presutti, V., Celino, I., Sabou, M., Kaffee, L.A., Simperl, E. (eds.) The Semantic Web – ISWC 2018, pp. 341–358. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-030-00668-6_21
63. Tsvetanka, G.T., Kaloyan, Z., Donika, V.: Application of semantic technologies in bibliographic databases: a literature review and classification. Electron. Library **38**(1), 113–137 (2019). https://doi.org/10.1108/EL-03-2019-0081
64. Upham, S.P., Small, H.: Emerging research fronts in science and technology: patterns of new knowledge development. Scientometrics **83**(1), 15–38 (2010). https://doi.org/10.1007/s11192-009-0051-9
65. van Eck, N.J., Waltman, L.: Citation-based clustering of publications using CitNetExplorer and VOSviewer. Scientometrics **111**(2), 1053–1070 (2017). https://doi.org/10.1007/s11192-017-2300-7. https://pubmed.ncbi.nlm.nih.gov/28490825 https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5400793/
66. Vergoulis, T., Chatzopoulos, S., Dalamagas, T., Tryfonopoulos, C.: Veto: Expert set expansion in academia. In: Hall, M., Merčun, T., Risse, T., Duchateau, F. (eds.) Digital Libraries for Open Knowledge, pp. 48–61. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-54956-5_4
67. Vergoulis, T., Chatzopoulos, S., Dalamagas, T., Tryfonopoulos, C.: Veto: expert set expansion in academia. In: Hall, M., Merčun, T., Risse, T., Duchateau, F. (eds.) Digital Libraries for Open Knowledge, pp. 48–61. Springer, Cham (2020)
68. Visser, M., van Eck, N.J., Waltman, L.: Large-scale comparison of bibliographic data sources: Scopus, web of science, dimensions, crossref, and microsoft academic (2021)
69. Xian, Y., Lampert, C.H., Schiele, B., Akata, Z.: Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly (2020)
70. Xie, L., Chen, Z., Wang, H., Zheng, C., Jiang, J.: Bibliometric and visualized analysis of scientific publications on atlantoaxial spine surgery based on web of science and vosviewer. World Neurosurgery **137**, 435–442 (2020)
71. Zhang, X., Chandrasegaran, S., Ma, K.L.: Conceptscope: Organizing and visualizing knowledge in documents based on domain ontology. In: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, pp. 1–13 (2021)
72. Zhang, Y., Lu, J., Liu, F., Liu, Q., Porter, A., Chen, H., Zhang, G.: Does deep learning help topic extraction? A kernel k-means clustering method with word embedding. J. Inf. **12**(4), 1099–1117 (2018). https://doi.org/10.1016/j.joi.2018.09.004

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.