SCICERO: A Deep Learning and NLP Approach for Generating Scientific Knowledge Graphs in the Computer Science Domain

Danilo Dessí[a], Francesco Osborne[b,c], Diego Reforgiato Recupero[a,*], Davide Buscaldi[d], Enrico Motta[b]

[a]*Mathematics and Computer Science Department, University of Cagliari, Cagliari, Italy*
[b]*Knowledge Media Institute, The Open University, Milton Keynes, United Kingdom*
[c]*Department of Business and Law, University of Milano Bicocca, Milan, Italy*
[d]*Laboratoire d'Informatique de Paris Nord, Sorbonne Paris Nord University, Paris, France*

Abstract

Science communication has a number of bottlenecks that include the rising number of published research papers and its non-machine-accessible and document-based paradigm, which makes the exploration, reading, and reuse of research outcomes rather inefficient. Recently, Knowledge Graphs (KG), i.e., semantic interlinked networks of entities, have been proposed as a new core technology to describe and curate scholarly information with the goal to make it machine readable and understandable. However, the main drawback of the use of such a technology is that researchers are asked to manually annotate their research papers and add their contributions within the KGs. To address this problem, in this paper we propose SCICERO, a novel KG generation approach that takes in input text from research articles and generates a KG of research entities. SCICERO uses Natural Language Processing techniques to parse the content of scientific papers to discover entities and relationships, exploits state-of-the-art Deep Learning Transformer models to make sense and validate extracted information, and uses Semantic Web best

*corresponding author

Email addresses: danilo.dessi@unica.it (Danilo Dessí), francesco.osborne@open.ac.uk (Francesco Osborne), diego.reforgiato@unica.it (Diego Reforgiato Recupero), davide.buscaldi@lipn.univ-paris13.fr (Davide Buscaldi), enrico.motta@open.ac.uk (Enrico Motta)

practices to formally represent the extracted entities and relationships, making the written content of research papers machine-actionable. SCICERO has been tested on a dataset of 6.7M papers about Computer Science generating a KG of about 10M entities. It has been evaluated on a manually generated gold standard of 3,600 triples that cover three Computer Science subdomains (Information Retrieval, Natural Language Processing and Machine Learning) obtaining remarkable results.

Keywords: Knowledge Graph, Scholarly Domain, Scientific Facts, Artificial Intelligence

1. Introduction

Exploring, interlinking, and analysing scientific literature is crucial for producing new knowledge and tools, improving our understanding of the world, and addressing the fundamental challenges of our time. It is also a very challenging task, since the literature is composed by millions of heterogeneous articles in natural language. Researchers and other stakeholders typically explore this complex space by using academic search engine (e.g., Google Scholar, Scopus, PubMed, Semantic Scholar), which however allow only basic queries and simply returns list of relevant documents that need to be manually analysed. This limitations constitute a significant bottleneck in the knowledge flow of the scientific process [1, 2].

The main issue is that current systems lack a good representation of the underline scientific knowledge and thus cannot support more sophisticated queries about the entities described in the literature such as *methods*, *tasks*, and *materials* as well as their relations (e.g., a method can be used to solve a task, a material to evaluate a method). A partial exception may be the field of biology, which can rely on some comprehensive, even if still noisy, representations of the relevant entities (e.g., UMLS [3]).

For this reason, the research community has been proposing several solutions for producing structured, interlinked, and machine-readable description of the scientific knowledge within research publications [2, 4, 5]. Typically, the resulting representation uses semantic web technologies, such as ontologies and knowledge graphs. Ontologies in computer science are *"explicit specifications of a conceptualization"* [6] that are used to formalize the conceptual schema of a domain by defining the types of entities and their relations.

They are typically encoded using the Web Ontology Language (OWL)[1] and are considered the cornerstones of the Semantic Web [7]. KGs consist of large networks of entities and relationships that provide machine readable and understandable information about a specific domain following a formal semantics [8]. They are composed by triples in the form of <`subject, predicate, object`>, for instance: <`Bill Gates, founderOf, Microsoft`>. The schema of a KG is often defined in a domain ontology. Large-scale KGs are typically produced semi-automatically from structured and unstructured data. Some well-known examples are DBpedia [9], Google Knowledge Graph[2], BabelNet[3], and YAGO[4].

Generating large-scale and high quality knowledge graphs of scientific knowledge from the literature is still an unsolved challenge [10]. Current solutions either rely on systems for assisting human experts in formalizing their knowledge [2, 5] or on information extraction pipelines [11, 12]. The first class of solutions is unable to scale and can only be applied to small domains (e.g., computational linguistics [13], intrusion detection [14]). Information extraction techniques can scale, but typically struggle to produce a high-quality output that can be used in a practical setting. In particular, current approaches for extracting entities and relationships from scientific texts [11, 15, 16, 12, 17] typically focus on processing individual documents. Generating a large-scale, consistent, and semantically sound representation of the scientific literature from million of articles is a very different task. Therefore, simply applying current methods for entity and relationship extraction on a large set of papers will produce a very noisy and incoherent result [18]. We thus need to solve several challenges in this space, such as: 1) integrating the outputs of different documents and tools in a coherent representation, 2) assessing the validity of the resulting triples, and 3) defining a flexible ontological schema to formalize a variety of statements from the literature.

In 2021, we tackled these issues by introducing the information extraction approach described in Dessì et al. [18], which is able to combine information

[1]https://www.w3.org/OWL/

[2]http://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html

[3]https://babelnet.org/

[4]https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/

from different tools according to a domain ontology and produce large-scale KGs. This approach inspired several further works in the field [19, 20, 21, 22, 23, 24] and was used to produce the Artificial Intelligence Knowledge Graph (AI-KG) [25], a knowledge base describing 820K research entities in the field of AI. However, this first attempt also suffered from several limitations, such as: i) the entity extraction modules did not take advantage of the expert knowledge acquired from the analysis of the resulting knowledge graphs, ii) a limited ability to merge together multiple versions of the same entity (e.g., *data cleaning algorithm*, *data cleaning automation*, and *preset data cleaning strategy*), iii) a shallow and manual method for mapping verbal predicates to semantic relations, and iv) a limited methodology for assessing the validity of a triple, based on a very simple multilayer perceptron classifier.

In this paper, we introduce SCICERO, an improved information extraction architecture that addresses all these limitations. This novel solution is able to extract five types of entities (tasks, methods, materials, metrics, and other entities) and several relationships between them (we currently support 179 object properties, but more can be added). Specifically, we introduced the following advancements to the previous approach [18]:

- A novel method to extract relationships which exploits frequent patterns extracted from the scientific literature and manually revised by domain experts.
- A new module to merge various shapes of the same entity by using the Computer Science Ontology (CSO) [26], Wikidata[5], and DBpedia [27], and the *SentenceTransformers* framework [28].
- A novel methodology to semi-automatically map predicates to semantic relationships using VerbNet [29].
- A novel module to check the validity of the triples based on SciBERT [30], a state-of-the-art transformer model for the scientific domain.

The code of SCICERO is publicly available at `https://github.com/danilo-dessi/SKG-pipeline`, in order to ensure full reproducibility and allow the community to further build on our effort.

[5]`https://www.wikidata.org/wiki/Wikidata:Main_Page`

We evaluated SCICERO against several alternative solutions on a benchmark composed by 3,600 triples in the field of the Information Retrieval (IR), Natural Language Processing (NLP), and Machine Learning (ML). We show that SCICERO yields the best performance in terms of F1 and discuss the effect of its different components (Section 5.2). We also demonstrate that the number of source documents of a statement can be used to estimate its reliability (Section 5.3), allowing users to choose different compromises between coverage and accuracy.

In summary, the main research contributions of this paper are:

- We introduce SCICERO, a new approach that integrates Semantic Web best practices, NLP, and Machine Learning for building KGs of scientific concepts from research papers.
- We release a ground truth[6] of 3,600 triples covering the field of Information Retrieval, Natural Language Processing, and Machine Learning.
- We perform an evaluation of SCICERO in terms of precision, recall, and F-measure.
- We provide a use case of SCICERO on a big dataset of scientific literature for producing a Computer Science Knowledge Graph.
- We make available the full source code of SCICERO at `https://github.com/danilo-dessi/SKG-pipeline`.

The remainder of this paper is organized as follows. Section 2 discusses the related work. SCICERO is detailed in Section 3. A use case and a KG generated with the proposed architecture can be found in Section 4. Section 5 reports the evaluation. Finally, Section 7 ends the paper by discussing limitations and defining future research directions where we are headed.

2. Related Work

The knowledge extraction task of processing scientific literature to produce KGs of scientific concepts is in its early age and most of the efforts in that direction have been carried out in the last decade [31]. The scientific community, as well as industry, have an increasing interest to keep up

[6] `https://github.com/danilo-dessi/SKG-pipeline/tree/main/eval`

with the growing number of research findings as the need to efficiently query them in the most efficient way is more and more urgent. Therefore, it has become essential to build AI-based pipelines and methodologies to deal with the amount of research papers avoiding overwhelming researchers and practitioners in the exploration, study, and application of research findings [32].

The use of scientific KGs is one of the main innovations to support the dynamics mentioned above. They are mainly of two types: i) KGs that describe the content of scientific publications (e.g., AI-KG [25], ORKG [2], and Nanopublications [33]), and ii) KGs that describe metadata such as authors, venues, citations, and research topics (e.g., AIDA [34], OpenAlex[7], the Microsoft Academic Graph (MAG)[8], AMiner [35], Open Academic Graph[9], Scholarly-data.org [36], Scopus[10], Semantic Scholar[11], OpenCitations [37], Dimensions[12], Core [38]).

KGs can be generated either manually or automatically. For example, ORKG and Nanopublications are manually built: they require researchers and practitioners to create RDF triples to describe scientific literature. Conversely, AI-KG is automatically generated and curated. SCICERO contributes to the automatic generation of scientific KGs and focuses on the Computer Science domain, although it would be straightforward to extend it to other domains as well. Several KGs focus a specific domain and they can thus be referred to as domain KGs, defined in [39] as:

"an explicit conceptualisation to a high- level subject-matter domain and its specific subdomains represented in terms of semantically interrelated entities and relations".

The first attempts to detect entities for KG generation exploited Part-Of-Speech (PoS) tags. For example, a graph-based approach called Babelfy[13] was built on top of Word Sense Disambiguation (WSD) and Entity Linking (EL) [40]. Later approaches started to combine a variety of resources and

[7]OpenAlex - `https://openalex.org/`

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

[9]Open Academic Graph - `https://www.openacademic.ai/oag/`

[10]Scopus - `https://www.scopus.com/`

[11]Semantic Scholar - `https://www.semanticscholar.org/`

[12]`https://www.dimensions.ai`

[13]`http://babelfy.org/`

include ensemble models that can capture contextual information to recognize pieces of text that represent entities. They could also infer potential relationships among them [12]. For example, FRED[14] [11], which builds on top of Boxer [41], identifies both entities and relationships from natural language text. More precisely, it extracts frames, events, concepts and entities, associates them to several ontologies, and structures the extracted pieces of text according to the Resource Description Framework (RDF). One challenge that it is difficult to address with FRED is the extraction of domain-specific entities and relationships that require tuned methods to capture and describe specific domain semantics. Moreover, FRED can only process a single text at a time and thus it cannot address the integration of information coming from various natural language sources into a KG. SCICERO differs from FRED because it specifically targets the scholarly computer science domain and aims at integrating information from a large number of research papers.

The parsing of scientific papers from unstructured text into structured form is a topic that has gained increasing attention from the scientific community and main publishers (e.g., Springer, Elsevier, etc.) [42, 43]. In 2017, within the *SemEval 2017 Task 10: ScienceIE - Extracting Keyphrases and Relations from Scientific Publications* [44] competition, participants were asked to provide tools and methods to find entities and relationships from a scientific annotated corpus of research publications. Since then, approaches for addressing this problem using syntactical patterns and machine learning-based models have been released and employed to transform plain text into structured graph representations [45, 46, 47]. An example of those is represented by the work described in [47] that was designed to capture the *hyponymy* relationship between noun phrases that were found in the text. Recurrent neural networks are used instead in [48] to jointly extract entities and relationships to create triples. In addition, authors in [48] also generated constraints (called *conditional tuples*) which can be applied to the extracted triples to check their validity according to their context of application.

Wadden et al. [49] introduced DyGIEpp, an approach for extracting triples from natural language scientific text. In their work, entities and relationships are extracted by using a transformer model able to capture the meaning of pieces of text that are associated with pre-defined entity types and find relationships among the entities. More recently, due to the pan-

[14]http://wit.istc.cnr.it/stlab-tools/fred/

demic situation, articles describing coronaviruses have been also targeted. For example, Wang et al. [50] adapted a recognition tool based on distant supervision for detecting 75 types of entities in the scientific literature about medicine. However, a drawback of the proposed approach is that the relationships among entities are not extracted from text.

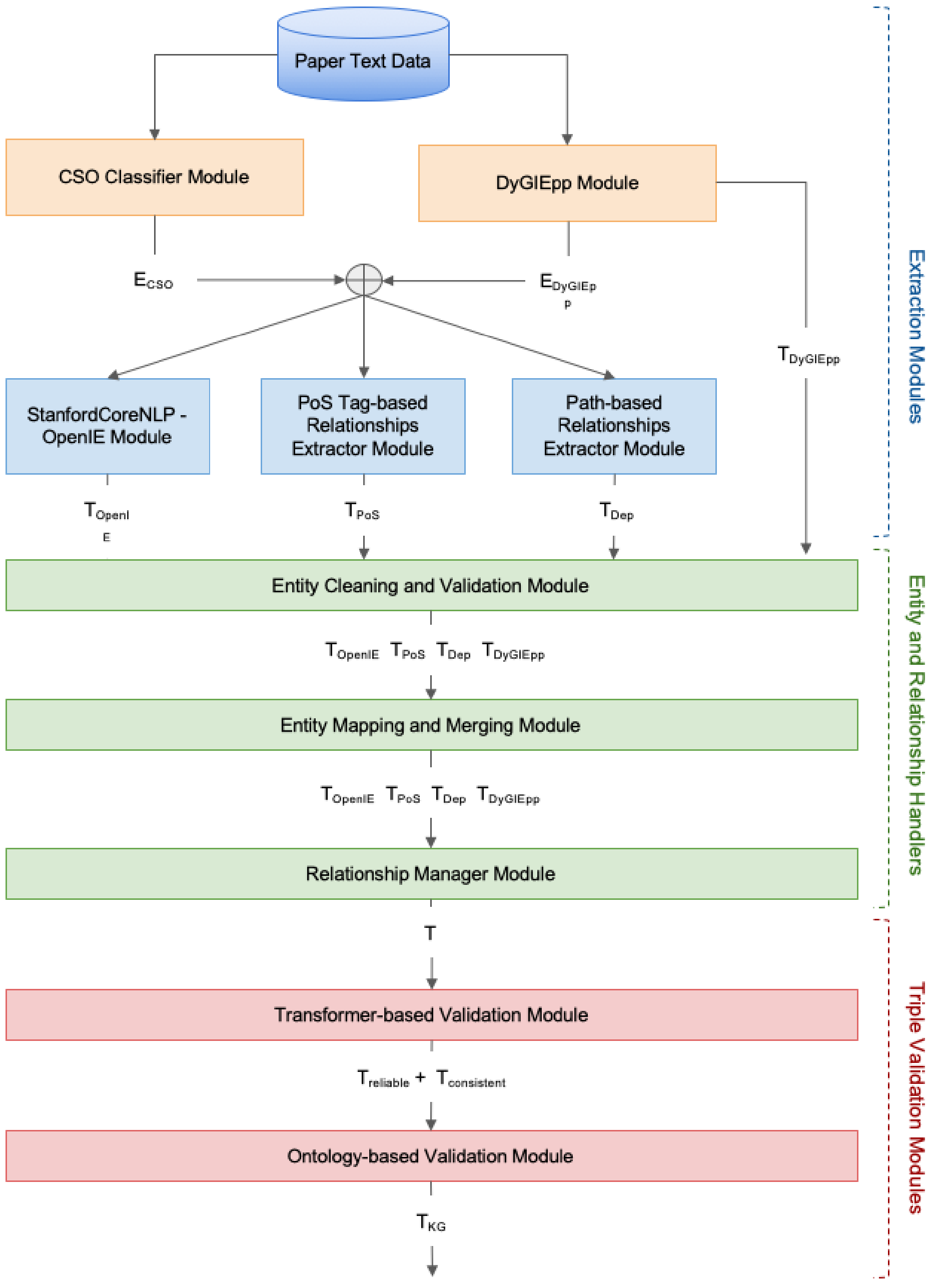
In addition, these tools present some limitations when applied on the vast amount of research papers available today. First of all, they do not fully integrate information coming from different pieces of text: the resulting graphs are simply created by generating nodes for the entities and edges for the relationships extracted by the single articles. This typically results in a very noisy output. Second, the limited number of supported relationships is not sufficient to express the meaning of difficult concepts expressed in natural language within a scientific paper. Additionally, existing methodologies usually produce KGs that do not have a good coverage, i.e., they do not contain the whole information about the entities of a specific domain, and present a variety of mistakes and incorrect statements. Indeed, KG construction methodologies usually present a trade-off between coverage and correctness [51]. This problem is particularly relevant for KGs which are generated by information extraction techniques. Therefore, addressing such shortcomings through the use of validation methods is crucial. The main developments about this topic can be found in recent works for KG completion and error detection [51]. To address this challenge, models built on top of KG embeddings [52, 53, 54], graph features [55], or transformers [56, 57], are proposed to verify the correctness of newly generated triples. They usually train a classifier on the existing KGs (i.e., they use a portion of the KG as a ground truth) since it is unfeasible to manually annotate thousands or millions of triples. The resulting classifier is then applied to validate new facts to be included in the KGs. Similarly to the existing literature, SCICERO exploits a transformer-based model which is fine-tuned on a set of reliable triples to identify triples that might describe erroneous facts about the Computer Science domain. Finally, existing approaches lack well-defined semantics and semantic web best practices to make sense of data and enable users to explore and reason on the generated graph. For example, the triples generated by DyGIEpp are not described by means of ontologies thus limiting the possibility to reason and make sense out of them. In comparison, SCICERO addresses the above-mentioned challenges by (i) integrating entities and relationships that come from different papers and extracted from different tools, (ii) validating the triples by means of ontology- and machine learning-based approaches, and (iii) following best practices in the design of ontologies to express the domain semantics.

3. The SCICERO Architecture

This section describes SCICERO, a novel approach for automatically generating KGs of scientific concepts. SCICERO takes in input 1) a collection of texts from scientific articles (typically titles and abstracts) and 2) a domain ontology and returns i) a set of *typed entities* extracted from the articles and ii) a set of *statements* describing the relationships between two entities as well as a number of relevant metadata. The entities are typed according to five possible classes (*Task*, *Method*, *Material*, *Metric*, and *OtherEntity*) and linked by 179 object properties as specified in the domain ontology available at `https://scholkg.kmi.open.ac.uk/cskg/ontology`. The ontology has been developed to formally describe the relationships among research entities within the computer science domain (see Section 3.3.2). Each statement is encoded as a *rdf:Statement* instance and contains the following information:

- the triple describing the relationship between two entities in the form of <`subject, predicate, object`> (using the object properties: *rdf:subject*, *rdf:predicate*, and *rdf:object*);
- the number of articles from which the claim was extracted (*cskg-ont:hasSupport*)
- the set of IDs of the articles from which the statement was extracted (*provo:wasDerivedFrom*);
- provenance and versioning information of the combination of techniques used to detect this specific statement (*provo:wasGeneratedBy*).

The architecture of SCICERO is depicted in Fig. 1. It includes three main components. First, the *extraction modules* (Section 3.1) apply two methods for identifying entities (CSO Classifier and DyGIEpp) and four techniques for extracting relationships between these entities (DyGIEpp, OpenIE, PoST, and DepT). This step produces four sets of triples: $T_{DYGIEpp}$, T_{OpenIE}, T_{PoS}, and T_{Dep}. In this phase, entities can be very noisy and relations are a combination of pre-determined relations from supervised methods (e.g., *part-of*,



Paper Text Data
CSO Classifier Module
DyGIEpp Module
E_CSO
E_DyGIEpp
T_DyGIEpp
StanfordCoreNLP - OpenIE Module
PoS Tag-based Relationships Extractor Module
Path-based Relationships Extractor Module
T_OpenIE
T_PoS
T_Dep
Entity Cleaning and Validation Module
T_OpenIE T_PoS T_Dep T_DyGIEpp
Entity Mapping and Merging Module
T_OpenIE T_PoS T_Dep T_DyGIEpp
Relationship Manager Module
T
Transformer-based Validation Module
T_reliable + T_consistent
Ontology-based Validation Module
T_KG
Extraction Modules
Entity and Relationship Handlers
Triple Validation Modules

Figure 1: The SCICERO's schema to generate Scientific KGs.

used-for, *feature-of*) and verbal predicates extracted by the NLP methods (e.g., *use*, *utilize*, *exploit*, *identify*, *select*).

In the second phase, the *entity and relationship handlers* (Section 3.2) integrate these triples in a common representation. Entities are cleaned by removing punctuation characters, discarding stop-words, and filtering out generic terms. Entities that refer to the same concept (e.g., *machine learning approach* and *machine learning method*) are also merged together. Relationships are then mapped to a target ontological schema. For instance, the verbal predicates *analyze*, *inspect*, and *examine* are mapped to the object properties which use the verb *analyze* as representative (i.e., *cskg-ont:analyzesMaterial*, *cskg-ont:analyzesMethod*, *cskg-ont:analyzesOtherEntity*, *cskg-ont:analyzesTask*, and *cskg-ont:analyzesMetric*).

Finally, the *triple validation modules* (Section 3.3) select the set of triples that will compose the knowledge graph according to a two-steps validation process. First, the transformed-based validation module identifies high-quality triples that are consistent with the ones associated with a good number of articles. Second, the ontology-based validation module filters out triples that do not comply with the ontological schema. For example, the triple <`cskg:labeled_text`, `cskg-ont:usesTask`, `cskg:named_entity_recognition`>, where *cskg:labeled_text* is a *cskg-ont:Material* and *cskg:named_entity_recognition* is a *cskg-ont:Task*, does not comply with the ontology schema because the class *cskg-ont:Material* is not in the domain of the object property *cskg-ont:usesTask* (a material cannot use a task). In the following we will describe in detail every step of the process.

3.1. Extraction Modules

This section describes the tools used to extract the entities and the relationships from natural language text. Specifically, we extract entities using two modules: 1) *DyGIEpp Module* [58] (Section 3.1.1, also used for identifying relationships) and 2) the *CSO Classifier Module* [59] (Section 3.1.2). We detect relationships between these entities according to four approaches: 1) *DyGIEpp Module* [58] (Section 3.1.1), 2) the *Stanford Core NLP - OpenIE Module* [60] (Section 3.1.3), 3) *PoS Tag-based Relationships Extractor Module* [61] (Section 3.1.4), and 4) *Path-based Relationships Extractor Module* (Section 3.1.5), a novel approach that we introduce in this paper.

3.1.1. DyGIEpp Module

DyGIEpp[58], a framework designed by Wadden et al., analyzes scientific abstracts to generate a set of entities and relationships. Specifically, SCI-

CERO employs the *DyGIEpp* model *scierc*[15], a BERT-based model tuned for parsing computer science scientific text. *DyGIEpp* can detect six types of entities (*Method, Task, Material, Metric, Other-Scientific-Term* and *Generic*), and seven types of relations (i.e., *Used-for, Hyponym-Of, Compare, Part-of, Conjunction, Feature-of, Evaluate-for*). Since we do not differentiate between *Other-Scientific-Term* and *Generic*, these are merged in the *Other-Entity* category. DyGIEpp applies a feed-forward neural network on input text textual span representations to compute two scores. The first score gives the probability for a text to be a research entity of pre-defined types. The second score gives the probability that one of the pre-defined relationships exists between two extracted entities given the text surrounding the entities. Its output is a set of entities $E_{DyGIEpp}$ and a set of relationships $T_{DyGIEpp}$. A softmax function is applied to distinguish one of the possible entity types or relationship. As an example, for the sentence "*The required simulation time is usually long for the power grid design.*", the text span *simulation time* has been identified as a research entity of type *Metric* with a softmax value of 0.96, the text span *power grid design* has been identified as an entity of type *Method* with a softmax value of 1.0, and *EVALUATE-FOR* has been identified as a relationship with softmax value of 0.98 in the generated triple <`simulation time`, `EVALUATE-FOR`, `power grid design`>.

3.1.2. CSO Classifier Module

The *CSO Classifier*[16] [59] is a classifier built on top of the Computer Science Ontology (CSO), an ontology that describes research topics in the field of Computer Science. The CSO Classifier uses unsupervised syntactic- and semantic-based classification components to detect research topics in a text i.e., it verifies whether a piece of text represents a CSO topic. The syntactic component detects unigrams, bigrams, and trigrams using a rule-based approach, applies the Levenshtein similarity, and uses a pre-defined threshold to compare the n-grams to research topics in CSO. The semantic component builds on top of a domain-trained Word2Vec model, regular expressions on PoS tags, and a threshold to compare a text span with CSO research topics. The output of this sub-module is the set of entities E_{CSO}. The reader can find more details about the CSO classifier in [59].

[15]`https://github.com/dwadden/dygiepp#pretrained-models`

[16]`https://github.com/angelosalatino/cso-classifier`

3.1.3. Stanford Core NLP - OpenIE Module

This module is built on top of the *Stanford Core NLP* suite [60]. *OpenIE* [62] is an annotator that extracts domain-independent triples from a text. To start with, it builds clauses which are groups of terms that include at least a subject-noun and a verb by traversing the parsing tree of the input text. Clauses are shortened thus creating a set of short sentence fragments, that are subsequently transformed into triples. In the context of the proposed approach, the resulting triples are filtered by checking the overlapping of subjects or objects with the set $E_{CSO} \cup E_{DyGIEpp}$. Triples whose subject or object are not part of $E_{CSO} \cup E_{DyGIEpp}$ are discarded. Moreover, only triples that have a relationship whose tokens are verbs according to the associated PoS tags are used to build the KG. The output of this module is the set of triples T_{OpenIE}.

3.1.4. PoS Tag-based Relationships Extractor Module

PoS Tagger[17] [61] is an annotator that associates a PoS tag with each term in an input text. PoS tags are used to detect verbs between pairs of entities in a sentence. In more details, given a sentence s_i and the set of entities extracted from it $E_i \subset E_{CSO} \cup E_{DyGIEpp}$, all the verbs $V = \{v_0, \dots, v_z\}$ between each pair of research entities $(e_m, e_n) | e_m, e_n \in E_i$ are used to create triples $<e_m,\ v,\ e_n>$ where $v \in V$. To reduce noise that this approach might generate, verbs are sought only between couples of entities whose distance (i.e., the number of tokens between two entities) is equal or less than a predefined window size w. The output of this module is the set of triples T_{PoS}.

3.1.5. Path-based Relationships Extractor Module

This novel module builds on top of the *Stanford Core NLP Dependency Parser* [63] and aims to extract meaningful triples by exploiting a set of pre-defined paths on the dependency trees of the sentences. In order to identify high-quality and frequent paths, a set of dependency trees $DT = \{dt_0, \dots, dt_z\}$ built on a representative sample of scientific papers, and a set of research entity pairs $EE_i = \{(e_{im}, e_{in}), \dots, (e_{io}, e_{ip})\}$ for each dt_i have been employed. First, to define these paths, the set of shortest paths P on the dependency trees in DT containing at least one verb between the tokens of a pair of entities is generated (i.e., in a path the two endpoints must

[17] `https://nlp.stanford.edu/software/tagger.shtml`

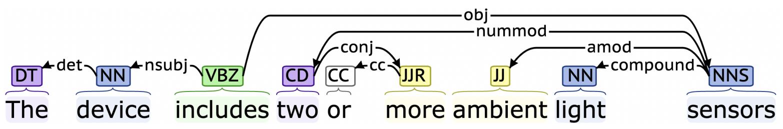


Figure 2: The dependency tree of the sentence "*The device includes two or more ambient light sensors*".

be identified as entities). Examples of paths on the dependency trees are ($nsubj \rightarrow obj$), ($nsubj \rightarrow obj \rightarrow conj$), ($acl \rightarrow obj$), and so on. The relationship between the two entities has to be a verb and is searched among the tokens of the path between the two entities. The 50 most frequent shortest paths composing the set $P' \subseteq P$ are considered candidates to be good paths (i.e., paths that capture meaningful text fragments that can be converted into triples and included in the KG). The rationale behind selecting frequent paths is that a path must identify a certain amount of knowledge; a path that only detects a few triples is not of interest for the KG.

For each $p \in P'$, 20 triples were subsequently extracted and manually annotated by 4 Computer Science researchers. For example, given the sentence "*The device includes two or more ambient light sensors*", the path ($nsubj \rightarrow obj$), the entities *device* and *light sensors*, and the sentence dependency tree (see Fig 2), where the words *device* and *includes* are linked by the relation *nsubj*, and the words *includes* and *sensors* are linked by *obj*, we can extract the triple <`device, includes, light sensors`>. Paths were evaluated considering the number of correct triples out of the twenty extracted. Paths which generated more than 60% of correct triples were chosen as good paths. This resulted in the set P'' composed by 12 paths[18].

Given a new input text t_{new} and its entities E_{tnew}, the module builds its dependency tree dt_{new} and uses the paths of the set P'' to extract relationships between pairs of entities $(e_m, e_n)|e_m, e_n \in E_{tnew}$. The output of this module is the set of triples T_{Dep}.

3.2. Entity and Relationship Handlers

This section describes how entities and relationships are cleaned and merged together.

[18]`https://github.com/danilo-dessi/SKG-pipeline/blob/main/resources/path.txt`

3.2.1. Entity Cleaning and Validation Module

The entity extraction process can generate very generic or noisy terms (*approach*, *model*, *model 1*, etc.). We thus apply a number of transformations. First, acronyms are solved by exploiting the fact that they are usually placed in brackets immediately after the extended form of the related entities (e.g., *Computer Science Ontology (CSO)*). Then, common English stop-words provided by the NLTK[19] library as well as punctuation signs (e.g., apostrophes and commas) are removed. All entities are lemmatized using the WordNet lemmatizer[20]. In order to discard generic entities, SCICERO filters out the ones with a Information Content (IC) score[21] lower than a empirically defined threshold th_{IC} (10 in the prototype, as reported in Section 4). It also use a blacklist of common entities[22]. To avoid discarding significant entities within the Computer Science domain, it uses a white-list which includes both CSO topics and the 'Fields of Study' subjects from MAG.

3.2.2. Entity Mapping and Merging Module

The same entity can appear in many different forms in a text, e.g., *machine learning approach*, *machine learning algorithm*, *machine learning method*, etc. All these forms need to be merged into a single entity. To achieve this goal, we apply two methods for entity merging that rely respectively on external knowledge bases and deep learning transformers.

Merging through Linking to External Resources. The set of entities E is mapped to CSO, Wikidata, and DBpedia. Given an entity $e \in E$, the linkage to CSO is done by performing string matching of a text representing the entity e and the research topics in CSO. The entity e is compared to the whole set of topic labels provided by CSO, say $\{l_{t_1}, \ldots, l_{t_n}\}$, where l_{t_i} is the label l of the i-th topic t. As a result, if there is a perfect match between e and a research topic label l_{t_i}, the entity e is used as representative entity of t_i, and of all alternative topics $\{t_n, \ldots, t_m\}$ of t_i (i.e., topics that have a different URI and are linked together by the property *cso:relatedEquivalent*[23]) whose labels $\{l_{t_n}, \ldots, l_{t_m}\}$ are present in the set of entities E. For example,

[19]`https://www.nltk.org/`

[20]`https://www.nltk.org/_modules/nltk/stem/wordnet.html`

[21]`https://www.nltk.org/howto/wordnet.html`

[22]`https://github.com/danilo-dessi/SKG-pipeline/blob/main/resources/blacklist.txt`

[23]`http://cso.kmi.open.ac.uk/schema/cso#`

the entity *natural language processing* is compared against all entities and its equivalent alternatives in CSO. The match occurs with the label *natural language processing* of the CSO topic *cso:natural_language_processing* which has alternative topics *cso:nlp* and *cso:natural_language_processing_systems*. In this case, the entity *natural language processing* is used as representative entity and is not changed within the triples. On the other side, the equivalent topics are subsequently checked within the set of entities E; if they are found, then the module replaces them with the entity *natural language processing* in the triple sets. For example, if *nlp* is found within E, then the entity *nlp* is replaced with the entity *natural language processing* in the triples of sets $T_{DyGIEpp}$, T_{OpenIE}, T_{PoS}, and T_{Dep}.

The linkage to Wikidata is done by using a SPARQL query which performs string matching with entities and their alternative labels (i.e., linked by the property *rdfs:altLabel* similarly to what was performed with the CSO). To limit potential false matches with entities of other domains, the SPARQL query is designed to find entities which are: i) instances of the entity *Computer Science*[24], ii) subclasses of *Computer Science*, iii) instances of a subclass of *Computer Science*, iv) part-of or facet-of *Computer Science*, and v) not ambiguous within Wikidata (i.e., they do not have *"Wikimedia disambiguation page"* as value of the property *schema:description*[25] which is present when a certain entity appears multiple times in Wikidata with different meanings).

Finally, entities are linked to DBpedia through *DBpedia Spotlight* [64][26], using a similarity score ≥ 0.8. Entities in the set E that are mapped to the same DBpedia entity are merged together. To avoid conflicts between the three linking strategies (e.g., the same entity is linked to different set of alternative labels by using the three external knowledge sources), this module links the entities by giving priority to CSO, then Wikidata, and finally DBpedia.

Merging through Transformers. Entities that are not linked to external resources are analyzed by this module to detect those that can be merged. To this purpose, entities in E are used to create an index based on the tokens they contain (i.e., each single token of the entities in E is used as key of the index, and the index values are the entities themselves). Two entities

[24]`https://www.wikidata.org/wiki/Q21198`
[25]`https://schema.org/description`
[26]`https://www.dbpedia-spotlight.org/`

$e_n, e_m \in E$ are compared if they share at least one token. The comparison is executed by employing the framework *SentenceTransformers* [28] and encoding the research entities with the *paraphrase-distilroberta-base-v2* transformer model. Research entities whose cosine similarity is equal to or greater than a threshold th_{merge} (set empirically to 0.9) are merged together. For example, suppose that the entities *machine learning approach*, *ml approach* and *ml method* are not linked by using external knowledge bases. First, the index {machine :[‘machine learning approach’], learning : [‘machine learning approach’], approach : [‘machine learning approach’, ‘ml approach’], ml : [‘ml approach’, ‘ml method’], method : [‘ml method’]} is created. Then, for each key of the index the encoding of the entities of its list of values are compared by the *SentenceTransformers* framework. Let us assume that the first key analyzed is ‘ml’. The entities *ml approach* and *ml method* are compared and merged (their similarity is $\geq$ than 0.9); e.g., the entity *ml method* is mapped to *ml approach*. Then, let us assume that the second key analyzed is *approach*. The entities *machine learning approach* and *ml approach* are compared and merged since also their similarity is $\geq$ than 0.9. The three initial entities are thus merged in the single entity *ml approach*.

3.2.3. Relationship Manager Module

The triples in $T_{DYGIEpp}$, T_{OpenIE}, T_{PoS}, and T_{Dep} typically present a large number of distinct relations that may be considered synonyms, produced by the four different methods of relationship extraction. For example, the relations *includes*, *involves*, *embeds*, and *contains* may be used to express the same information as their meaning is similar. In order to reduce redundant information, we map the extracted relations to a relatively small number of predefined relations. To this purpose, we generated a mapping that links 464 verbs to 39 representative verbs. This mapping has been built by using the VerbNet [29] verb taxonomy to extend the mapping of scientific verbs introduced in [25], which covered only 194 verbs. Therefore, thanks to VerbNet we were able to add 270 additional verbs.

VerbNet is a verb-based lexicon which structure syntactic and semantic information to categorize verbs in hierarchically organized classes. Classes are characterized by a set of verbs and by additional sub-classes, which we do not consider here since their meaning is often quite different from the initial class. VerbNet can be enriched with domain-specific jargon while holding as a core the most common use and semantics of verbs from more general contexts.

The verb mapping generated in [25] was built by applying a hierarchical clustering algorithm on the Word2Vec embeddings of the verbs. The closest verb to the centroid of each cluster was selected as representative verb for all the other verbs in the same cluster.

The representative verbs of the existing mapping are used to i) manually explore the VerbNet classes, and, ii) select the VerbNet class according to the meaning that we wanted to convey within the generated KG. For example, the representative verb *support* was firstly used to explore the extension of VerbNet classes yielding *support-15.3*, *contiguous_location-47.8* and *admire-31.2*; subsequently only the class *support-15.3* was held for the meaning that we wanted to give to the triples that include the *support* verb relation. For example, including the verbs *applaud* and *venerate* from the class *admire-31.2*, and hence including the class *admire-31.2* itself, would give a misleading meaning to the representative verb *support* according to what an expert would expect from a triple <`A, support, B`> in a scientific KG.

After selecting the VerbNet class for each representative verb, all the verbs of the chosen VerbNet class were mapped to the representative verb. For example, the verbs *hold* and *bear* from the class *support-15.3* were mapped to the representative verb *support*. Then, each representative verb which were present in the original mapping provided by [25] were included to complete the new manually generated map of verbs. For example, the verbs *assist* and *enable* and others from the mapping defined in [25] were also included and mapped to *support*. Finally, a manual review was carried out and a few more classes were created to avoid that some verbs were mapped to representative verbs whose meaning was largely different. The resulting mapping is available online[27].

SCICERO applies the mapping on the sets which include verb relations i.e., T_{OpenIE}, T_{PoS}, and T_{Dep}, switching the original verb with the representative verb. Some triples may end up having the same subject, relation, and object after this and are thus merged. For example, the triples <`knowledge discovery process, define, supervised machine learning`> and <`knowledge discovery, specify, supervised machine learning`> are mapped and merged to the same triple <`knowledge discovery process, defines, supervised machine learning`> because the verbs *define* and *specify* are

[27]`https://github.com/danilo-dessi/SKG-pipeline/blob/main/resources/CSKG_VerbNet_verb_map.csv`

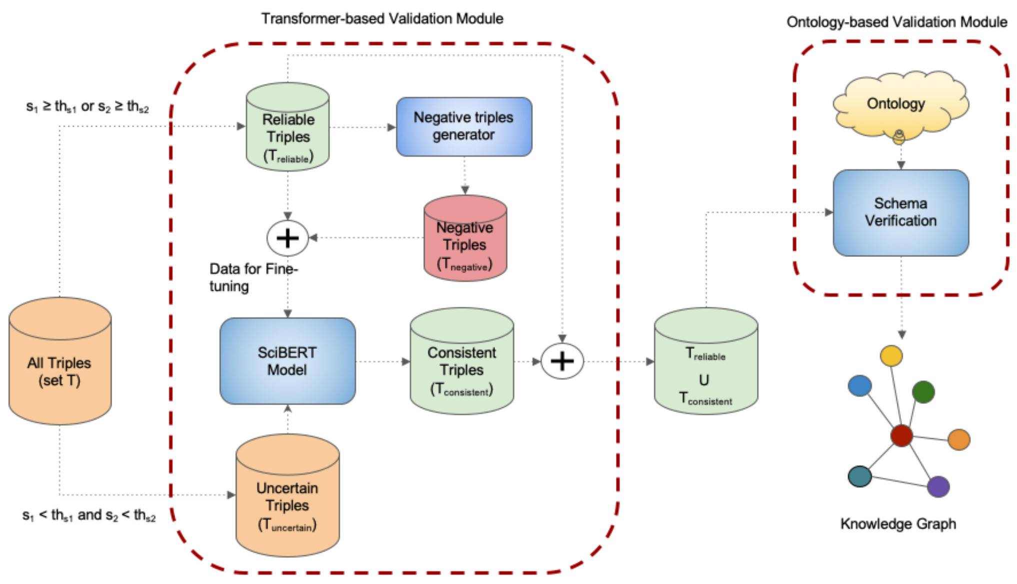

Transformer-based Validation Module
Ontology-based Validation Module
$s_1 \geq th_{s1}$ or $s_2 \geq th_{s2}$
Reliable Triples ($T_{reliable}$)
Negative triples generator
Negative Triples ($T_{negative}$)
Data for Fine-tuning
SciBERT Model
Consistent Triples ($T_{consistent}$)
All Triples (set T)
Uncertain Triples ($T_{uncertain}$)
$s_1 < th_{s1}$ and $s_2 < th_{s2}$
$T_{reliable}$ U $T_{consistent}$
Ontology
Schema Verification
Knowledge Graph

Figure 3: The schema of the validation modules used to discard erroneous triples.

mapped to the same representative verb *defines*. Finally, the pre-defined relations in the set $T_{DyGIEpp}$ are manually mapped to specific verb relations of the defined mapping which is available online[28]. After this process, a unique set T including all the extracted and processed triples is generated.

3.3. Triple Validation Modules

This section describes the two methodologies applied by SCICERO to discard triples that i) are erroneous artifacts of the information extraction process or the integration procedure, or ii) do not align with the knowledge contained in the generated KG or the domain ontology. Figure 3 shows the schema of the two modules (i.e., the *Transformer-based Validation Module* and the *Ontology-based Validation Module*) that perform this validation. The first method exploits transformers to identify low-quality triples. The second one checks that the type of entities and the relationships align with the ontological schema.

[28]`https://github.com/danilo-dessi/SKG-pipeline/blob/main/resources/SKG-dygiepp-Mapping.csv`

3.3.1. Transformer-based Validation Module

Intuitively, a triple that is extracted from multiple documents by multiple approaches has a higher probability to be correct than a triple that is extracted from one or a few documents using one or a few approaches. Section 5.3 discusses the quantitative data that confirm this intuition. Therefore, within SCICERO, we use two scores to define a triple as *reliable* or *uncertain*, thus selecting triples which can be directly included in the resulting KG and triples which need to be validated. The first score, s_1, is given by the number of papers from which a triple was extracted from. The second score, s_2, is given by the number of extractor tools that were able to detect the triple. For example, if a triple t is extracted from 10 papers and the tools that were able to detect it were *DyGIEpp Module* and the *Path-based Relationships Extractor Module*, the two scores will be $s_1 = 10$ and $s_2 = 2$. We refer to these scores as *support*.

Using the *support*, it is possible to define a subset of reliable triples $T_{reliable}$, and a subset of uncertain triples $T_{uncertain}$. The sets $T_{reliable}$ and $T_{uncertain}$ are disjoint and their union is equal to T. In this module, the set $T_{reliable}$ is used to fine-tune a transformer model on the sequence classification downstream task (i.e., the task of classifying sequences of text according to a given number of classes). More precisely, the transformer model implements a function $\theta : t \rightarrow \{0, 1\}$ that takes a triple t as an input and predicts 1 if t represents a correct fact that can be added in the KG, 0 if the triple t does not represent a correct fact and must be discarded. The rationale is that a triple consistent with the triples in $T_{reliable}$ is typically of good quality and could be included in the KG. In other words, the transformer model is tuned to verify whether a triple from the set $T_{uncertain}$ can be part of the KG whose triples are in the set $T_{reliable}$ [57]. This module uses the *scibert_scivocab_uncased* [30] transformer model which is pre-trained on the full texts of 1.14M papers from Semantic Scholar[29]. The transformer model is fine-tuned according to the set $T_{reliable}$ which includes all triples with support $s_1 \geq th_{s1}$ or $s_2 \geq th_{s2}$. The reader notices that SCICERO is aimed to be applied on large collections of research papers which will naturally include redundant statements appearing in multiple triples. Hence, it can always rely on a set of triples with high *support* for the training phase. For this reason our approach does not typically suffer from the cold start problem. For each triple $t \in T_{reliable}$, a triple $t' | t' \notin T$, generated by

[29]`https://www.semanticscholar.org/`

corrupting t, i.e., replacing either the subject or the object with a random chosen entity, is created. The triples $\{t'_0, \ldots, t'_n\}$ are used to build the set of negative triples $T_{negative}$. The set $T_{reliable} \cup T_{negative}$ is used to fine-tune the model. $T_{reliable}$ represents the triples whose label is 1, whereas $T_{negative}$ is the set of triples whose label is 0. To feed the model, each triple is transformed into text, e.g., the triple <`authentication protocol, uses, cryptography`> is transformed in the plain text *authentication protocol uses cryptography*, which is then encoded and used within the model. The trained model is then applied on all triples in $T_{uncertain}$ to verify whether they are consistent with the set $T_{reliable}$. All the triples with the label predicted by the transformer equal to 1 are used to build the set $T_{consistent}$.

3.3.2. Ontology-based Validation Module

This section describes the ontology used to represent scientific facts and how SCICERO exploits it to filter out semantically erroneous triples. The ontology uses the namespace `http://scholkg.kmi.open.ac.uk/cskg/ontology#` (prefix *cskg-ont*) and builds on top of SKOS[30] and PROVO[31]. It is designed to represent the domain semantics on top of the types of entities identified by the *DyGIEpp* tool (i.e., *Method, Task, Material, Metric, OtherEntity*) and the 39 representative verbs (described in Section 3.2.3). It describes five classes of research entities: *cskg-ont:Method*, *cskg-ont:Metric*, *cskg-ont:Material*, *cskg-ont:Task*, *cskg-ont:OtherEntity.* The types *Other-Scientific-Term* and *Generic* from *DyGIEpp*, which are associated to generic entities with no specific categorization, were both mapped to the class *cskg-ont:OtherEntity.* The ontology also describes 179 object properties (e.g., *cskg-ont:usesMethod*, *cskg-ont:includesMaterial, cskg-ont:Method, cskg-ont:analyzesMetric*) which were derived from the 39 representative verbs. For example, the representative verb *produces* was used to create the object properties *cskg-ont:producesMethod*, *cskg-ont:producesMetric*, *cskg-ont:produces:Task*, *cskg-ont:producesOtherEntity*, and *cskg-ont:producesMaterial*, the verb *queries* is used to create the object properties *queriesMaterial*, *cskg-ont:queriesOtherEntity*, and so on.

Each object property is associated with domain and range restrictions. For example, since it is correct to say that a *Method* or a *Task* uses a *Material*,

[30]SKOS - `https://www.w3.org/2004/02/skos/`
[31]PROV-O - `https://www.w3.org/TR/prov-o/`

the verb *uses* was utilized to create the object property *usesMaterial* which has both *Method* and *Task* in its domain, and *Material* as its range. At the same time, it is incorrect that a *Material* uses a *Method*, therefore, in the definition of the object property *usesMethod*, the class *Material* was not included in the domain of the property. The reader can find the ontology at `http://scholkg.kmi.open.ac.uk/cskg/ontology#`.

This module analyzes all triples from the set $T_{reliable} \cup T_{consistent}$ and discards them if the subject or the object do not comply with the range and domain of the relation as defined in the ontology. For example, the triple <`dbpedia, usesMethod, deep learning`>, where *dbpedia* is a *Material* and *deep learning* is a *Method*, is discarded, whereas the triple <`sentiment analysis, usesMaterial, lexical resource`>, with *sentiment analysis* being a *Task* and *lexical resource* being a *Material*, is included in the KG. Finally, the triples and their relevant metadata (e.g., support, linked papers) are refied and encoded in RDF.

4. Use Case

In this section, we describe the application of SCICERO on a big dataset of scientific literature for producing a Computer Science Knowledge Graph (CS-KG)[32], a knowledge graph describing 10M entities and 41M statements.

4.1. Dataset

We selected the abstracts and titles of 6.7M research articles in the field of Computer Science. This sample was retrieved from the MAG dataset, considering only papers associated with the Field of Study "Computer Science" in 2010-2021 and with at least one citation. We also required the abstracts to contain a number of tokens between 15 and 250.

4.2. Implementation Settings

This section describes the setting parameters as well as the statistics about the SCICERO KG generation process. The settings used to generate the scientific KG are reported in Table 1. Thresholds and scores were empirically set to find a balance between the quality of the overall triples and the coverage of the KG.

[32]CSKG - `https://scholkg.kmi.open.ac.uk/`

Table 1: Models and parameter values used to execute SCICERO to create the KG about Computer Science.

Description & Symbol	**Value**
PoS Tagger windows size w	15
Information Content th_{IC}	10
Embeddings similarity threshold th_{merge}	0.9
Support s_1	3
Support s_2	3

The *Extraction Modules* produced $53M$ triples from *DyGIEpp*, $34M$ from *OpenIE Module*, $58M$ from the *PoS Tag-based Relationships Extractor*, and $14M$ from the *Path-based Relationships Extractor*. Entities and Relationships were cleaned by the *Entities and Relationships handling* modules yielding a total of $68M$ candidate triples. Among them, $2.6M$ had a high support according to the thresholds of s_1 and s_2 reported in Table 1 and composed the set $T_{reliable}$. $500K$ of these highly supported triples were used in a train and test split setting to fine-tune the transformer model within the *Transformer-based Validation Module*. Specifically, 80% of the highly supported triples were employed for fine-tuning and the remaining 20% were used to evaluate the model. The evaluation of the transformer model in terms of precision, recall, and f-measure is reported in Table 2. The first row of Table 2 describes how the fine-tuned model is able to predict the Class 0 (i.e., to correctly discard inconsistent triples). The second row reports the performance in predicting the Class 1 (i.e., to correctly recognize highly supported triples and, therefore, consistent triples with the training set). Finally, the third row reports the macro average indicating an overall good performance of the model in recognizing consistent triples. The transformer model was applied on $66M$ lowly supported triples ($T_{uncertain}$) and recognized as consistent $46M$ triples ($T_{consistent}$). Finally, the *Ontology-based Validation Module* was applied on the set $T_{reliable} \cup T_{consistent}$ and filtered out about $7M$ triples that were inconsistent with the ontology. The remaining $41M$ triples were reified to generate the KG. In Table 3, we report some examples of i) triples that were considered valid by SCICERO and included in the KG, ii) triples that were discarded by the *Transformer-based Validation Module*, and iii) triples that were discarded by the *Ontology-based Validation Module*.

4.3. The Computer Science Knowledge Graph

Table 2: Precision (P) Recall (R) and F-measure score (F1) of the transformer model applied on the test set containing highly supported triples.

Class - Average	**P**	**R**	**F1**
Class 0 (triples removal)	0.85	0.81	0.83
Class 1 (consistent triples recognition)	0.82	0.86	0.84
Macro Average	0.83	0.83	0.83

```
cskg:statement_4623968 a cskg-ont:Statement, prov:Entity ;
            rdf:subject cskg:face_recognition ;
            rdf:predicate cskg-ont:providesTask ;
            rdf:object cskg:authentication ;
            cskg-ont:hasSupport 5 ;
            provo:wasDerivedFrom cskg:2403598213,
                                 cskg:2940931619,
                                 cskg:2966617151,
                                 cskg:3017824383,
                                 cskg:958306837 ;
            provo:wasGeneratedBy cskg:OpenIE,
                                 cskg:PoSTagger .
```

Figure 4: The statement describing the triple <cskg:face_recognition, cskg-ont:providesTask, cskg:authentication>.

The CS-KG generated by SCICERO includes 41M statements extracted from the 6.7M input abstracts. Statements are formalized according to the domain ontology described in Section 3.3.2.

Research entities are described within the `http://scholkg.kmi.open.ac.uk/cskg/resource/` namespace (prefix *cskg*). Each statement is associated with the research papers it was generated from (using the property *provo:wasDerivedFrom*), the number of source papers (*cskg-ont:hasSupport*), and the modules that were involved in its extraction (*provo:wasGeneratedBy*). An example of statement generated by SCICERO is depicted in Fig. 4. The triple <`face_recognition`, `providesTask`, `authentication`> is reified and linked to research papers where it was extracted from (e.g., *2940931619*, *2966617151*, and so on), the extraction modules that generated it (i.e., *OpenIE* and *PoSTagger*), and its support (i.e., 5).

Table 3: Examples of valid and discarded triples according to the validation performed by the *Transformer-based Validation Module* and the *Ontology-based Validation Module*

subject	**relation**	**object**	**status**
feed forward neural network system	uses	back propagation technique approach	valid
intelligent assistant system	uses	knowledge graph	valid
object recognition model set	includes	edge detection	valid
wind power forecasting methodology	uses	deep neural learning network	valid
time synchronization	uses	controls message	valid
robust networking	uses	continual connectivity	valid
blending anomaly	skos:broader	insider activity	valid
machine learning	skos:broader	climate adaptation strategy	discarded by transformer
machine learning	skos:broader	buzzword sounding technique	discarded by transformer
sentiment analysis	produces	time migration amount	discarded by transformer
cable tv network operator	produces	knowledge graph	discarded by transformer
numerical optimization	uses	wikidata	discarded by transformer
private cloud	skos:broader	post processing	discarded by ontology
natural language processing technique	skos:broader	cyber abuse	discarded by ontology
knowledge base	executes	virtual space	discarded by ontology
knowledge graph	produces	fault handling plan	discarded by ontology

5. Evaluation

This section reports the evaluation of SCICERO on a benchmark of 3,600 manually annotated triples and discusses in detail the contribution of the different components introduced in Section 3.

5.1. Evaluation Protocol

The evaluation of SCICERO has been performed by assessing its ability to extract and recognize correct triples. To this purpose, we produced *CS-KG3600*, a benchmark of 3,600 manually annotated triples. We first extracted a set of triples from the ones used to produce CS-KG (Section 4.3). Specifically, we selected 1,200 triples from three sub-fields of computer science, i.e., *Machine Learning*, *Natural Language Processing*, and *Information Retrieval*. We consider a triple about a certain domain if the subject or the object is a sub-topic of that domain in CSO[33].

For instance, the triple <`cskg:affinity_propagation`, `skos:broader`, `cskg:cluster_analysis` > was assigned to Machine Learning because `cskg:-cluster_analysis` is a sub-topic of Machine Learning in CSO. The full list of sub-topics is available online[34].

The resulting set is composed six sub-sets of 600 triples each:

- **Set 1 (Very High Support)**, which includes triples with $s_1 \geq 5$.
- **Set 2 (High Support)**, which includes triples with $3 \leq s_1 < 5$ or $s_2 \geq 3$.
- **Set 3 (Low Support)**, which includes triples with $s_1 < 3$ and $s_2 < 3$.
- **Set 4 (Discarded by the Transformer-based Validation)**, which includes triples that have been discarded by the *Transformer-based Validation Module* during the creation of CS-KG.

[33]The relevant topics on CSO are Machine Learning (`https://cso.kmi.open.ac.uk/topics/machine_learning`), Natural Language Processing (`https://cso.kmi.open.ac.uk/topics/natural_language_processing`), and Information Retrieval (`https://cso.kmi.open.ac.uk/topics/information_retrieval`).

[34]CSO Topics - `https://github.com/danilo-dessi/SKG-pipeline/blob/main/resources/cso_topcis.txt`

- **Set 5 (Discarded by the Ontology-based Validation)**, which includes triples have been discarded by the *Ontology-based Validation Module* during the creation of CS-KG.

- **Set 6 (Random)**, which includes triples that were randomly generated by replacing the head or tail of a triple in CS-KG.

The set *CS-KG3600* was manually annotated by 3 senior researchers (different than the authors of this paper) in the three chosen domains. For each triple, experts were asked to annotate 1 if the triple was correct according to their expertise and the scientific literature and 0 otherwise. Experts were provided with the type of the entities (e.g, *Method*, *Material*) and were free to use digital libraries, such as Google Scholar and Scopus. The Fleiss' kappa agreement [65] between the three annotators was 0.525, indicating a moderate agreement. We created the gold standard using the majority rule approach.

5.2. Results and Discussion

We compared 18 alternative approaches on *CSKG3600*:

- DyGIEpp [58], described in Section 3.1.1;

- OpenIE [60], described in Section 3.1.3;

- PoST [61], described in Section 3.1.4;

- DepT, introduced in this paper and illustrated in Section 3.1.5;

- eleven combinations of DyGIEpp, OpenIE, PoST, and DepT - integrated according to the *Entity and Relationship Handlers* (described in Section 3.2);

- a partial version of SCICERO using only the *Transformer-based Validation Module* (Section 3.3.1);

- a partial version of SCICERO using only the *Ontology-based Validation Module* (Section 3.3.2);

- the full version of SCICERO.

Section 5.2.1 focuses on the first 15 approaches, which are based on the four extraction tools and their integration, while Section 5.2.2 reports the performance of the methods that also use the validation modules.

5.2.1. Extractors Performance

Table 4 reports the performances of the 15 approaches in terms of precision, recall, and F-measure. The four basic methods for extracting triples (DyGIEpp, OpenIE, PoST, and DepT) obtained a precision score in the range 0.57 – 0.68. In terms of recall, the extractors obtained values lower or equal than 0.4, which indicates that a single tool or module is not sufficient to extract an amount of information that can be used to adequately describe the Computer Science domain. The F-measure of the extractors ranges from 0.24 to 0.49. However, when the extractors are combined, the recall increases without paying too much in terms of precision. The version which integrates all of them yields an F-measure score of 0.69. The F-measure increases with the number of tools that are combined: the combination of two tools leads to a maximum F-measure of 0.60, the combination of three of them reaches an F-measure of 0.65, and, as aforementioned, the combination of all the four tools achieves an F-measure of 0.69. This suggests that combining different tools for triple extraction, which rely on possibly complementary strategies, may be the best solution.

Table 4: Precision (P), Recall (R), and F-measure (F) of the evaluation of the proposed base modules on the *CS-KG3600* set.

Triples identified by	**P**	**R**	**F**
DyGIEpp	**0.68**	0.38	0.49
OpenIE	0.58	0.28	0.37
PoST	0.59	0.40	0.48
DepT	0.57	0.15	0.24
DyGIEpp + OpenIE	0.60	0.51	0.55
DyGIEpp + PosT	0.58	0.61	0.60
DyGIEpp + DepT	0.62	0.45	0.52
OpenIE + PoST	0.54	0.52	0.53
OpenIE + DepT	0.56	0.35	0.43
PosT + DepT	0.57	0.46	0.51
DyGIEpp + OpenIE + PoST	0.55	0.80	0.65
DyGIEpp + OpenIE + DepT	0.57	0.56	0.57
DyGIEpp + PoST + DepT	0.57	0.68	0.62
OpenIE + PoST + DepT	0.54	0.58	0.56
DyGIEpp + OpenIE + PoST + DepT	0.54	**0.95**	**0.69**

Table 5: Precision (P), Recall (R), and F-measure (F) evaluation of SCICERO on the *CS-KG3600* set.

Triples identified by	**P**	**R**	**F**
DyGIEpp + OpenIE + PoST + DepT	0.54	0.95	0.69
DyGIEpp + OpenIE + PoST + DepT + Transformer	0.65	**0.91**	0.76
DyGIEpp + OpenIE + PoST + DepT + Ontology	0.59	0.83	0.69
SCICERO (DyGIEpp + OpenIE + PoST + DepT + Transformer + Ontology)	**0.75**	0.79	**0.77**

5.2.2. Triple Validation Modules Impact

Although combining the extractor tools allows to achieve a good recall score, the resulting triples are fairly noisy, yielding a precision of 0.54 (see precision of *DyGIEpp + OpenIE + PoST + DepT* in Tables 4 or 5). In this section we study the effect of the two approaches for assessing triples: *Transformer-based Validation Module* and *Ontology-based Validation Module*. Table 5 reports the results of the experiments.

The application of the *Transformer-based Validation Module* allows to increase the precision by 0.11 while lowering the recall of by 0.04. The *Ontology-based Validation Module* increases the precision by 0.05, but loses 0.08 in terms of recall. The final version of SCICERO, which combines the two strategies, outperforms all the other methods in terms of precision (0.75) and F-measure (0.77).

Table 6 reports the performance of the full version of SCICERO on the three sub-fields: Natural Language Processing, Information Retrieval, and Machine Learning. The performance are similar across the different research areas, suggesting that SCICERO is quite consistent in the Computer Science domain.

In conclusion, the experiments yielded two main insights. First, tools only working at level of single documents may be not sufficient to produce a comprehensive representation of the domain. Therefore, we may need to develop solutions that are able to integrate and validate information from multiple tools and multiple documents. Second, the approaches for filtering triples based on their consistency with highly supported ones (the *Transformer-based Validation Module*) and compliance with the domain ontology (the *Ontology-based Validation Module*) can drastically improve the quality of the produced triples.

Table 6: Precision (P), Recall (R), and F-measure (F) evaluation of the full SCICERO pipeline (i.e., *DyGIEpp + OpenIE + PoST + DepT + Classifier + Ontology (full pipeline)*) on the Natural Language Processing, Information Retrieval, and Machine Learning subdomains in *CS-KG3600*.

Computer Science subdomain	**P**	**R**	**F**
Natural Language Processing	0.71	0.82	0.76
Information Retrieval	0.76	0.79	0.77
Machine Learning	0.78	0.77	0.78

5.3. Analysis of the CS-KG3600 Subsets

Figure 5 reports the percentage of correct triples for each of the six subsets in *CS-KG3600*, as defined in Section 5.2.1. The two sets with high support contain the majority of correct triples (86% and 75%), indicating that the support can be used as a reliable confidence score for the triples within the generated KG. Triples with a low support are less often correct (65%).

A manual analysis showed that highly supported triples typically describe more general facts about Computer Science and on which the community reached a consensus, e.g., <`cskg:computer_vision`, `cskg-ont:usesTask`, `cskg:image_retrieval`>, <`cskg:document_classification`, `cskg-ont:usesMethod`, `cskg:bayesian_classification`>). Conversely, lowly supported triples tend to describe more specific use cases or claims that often need to be interpreted within a specific context. For instance, it is difficult to say if the triple <`cskg:automatic_detection_of_gait_phase`, `cskg-ont:usesMethod`, `cskg:markov_model`> is correct or not without considering the specific research paper[35] where it was extracted from.

Interestingly, 34% of the triples discarded by the transformer module and 22% of those discarded by the ontology module are actually correct according to the annotators. Finally, 14% of the random generated triples have been labeled as correct by humans. This might be due to the fact that these triples contain at least one entity from the selected subdomains, and the other one from the KG, and this may produce a few plausible triples.

5.4. Limitations and Applicability to other Domains

This section discusses the main limitations of SCICERO in its current implementation and outlines what needs to be further developed for applying

[35]`https://doi.org/10.1109/TMECH.2018.2836934`

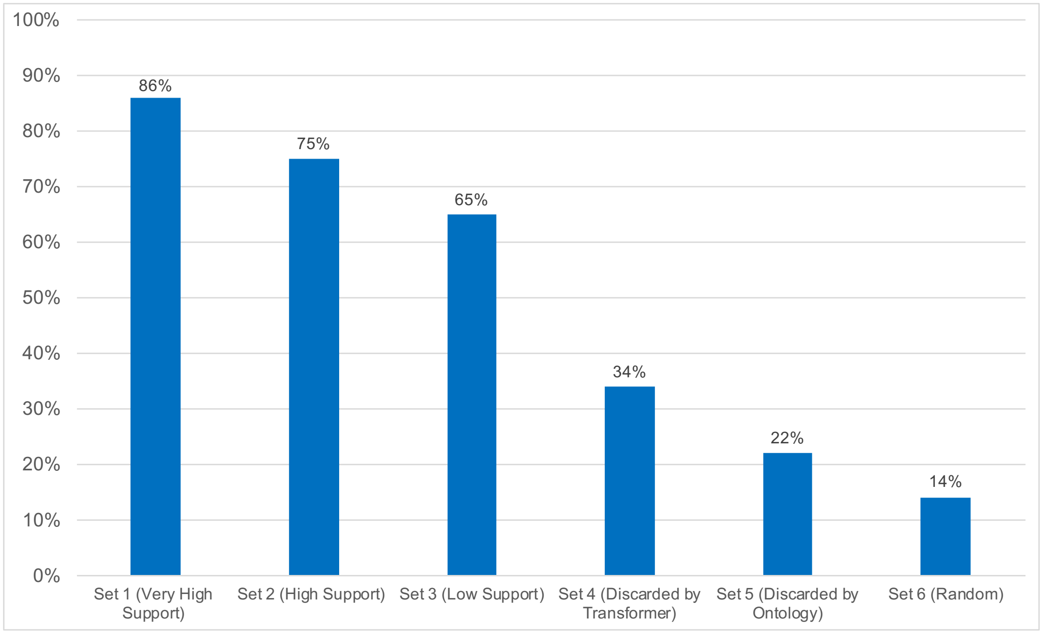


Figure 5: Percentage of correct triples within subsets of CS-KG3600.

it to other domains.

A first limitation is indeed related to the fact that several of the SCICERO components are tailored to computer science, therefore it cannot be used as it is on other scientific fields. Specifically, the extractor modules which use *DyGIEpp* and its *scierc* model as well as the *CSO Classifier* are exclusive to this research area. However, both of them can be modified to work on different fields. Specifically, the *DyGIEpp* framework provides several pre-trained models that can be exploited to extract pre-defined types of entities and relationships from other fields. For instance, it can be used in the biomedical domain by adopting the *GENIA*[36] model, in the protein/drug domain by using the ChemProt[37] model, and in the news domain by adopting the model trained on the *ACE 05*[38] corpus.

As far as the *CSO Classifier* is concerned, it can be replaced by tools

[36]GENIA - `https://s3-us-west-2.amazonaws.com/ai2-s2-research/dygiepp/master/genia.tar.gz`

[37]ChemProt - `https://ai2-s2-research.s3-us-west-2.amazonaws.com/dygiepp/master/chemprot.tar.gz`

[38]ACE 05- `https://catalog.ldc.upenn.edu/LDC2006T06`

that can recognize entities belonging to a specific taxonomy. For example, *SciSpacy*[39] can be used to recognize entities belonging to the biomedical domain from plain text and to map them to several ontologies and taxonomies such as *Medical Subject Headings (MeSH)*[40], *Unified Medical Language System (UMLS)*[41], *Gene Ontology (GO)*[42], *RxNorm*[43], and *Human Phenotype Ontology (HPO)*[44]. The existing modules to find and link entities to external resources can also be developed for other domains considering the abundance of available resources (e.g., in Mathematics using the *Mathematics Subject Classification (MSC)*[45] and in Physics using the *Physics Subject Headings (PhySH)*[46]). In order to run SCICERO on a different field it is also necessary to modify the domain ontology and the mapping between verbal predicates and relations. Indeed, the types of entities and the terminology used may depend heavily on the domain.

A second limitation of SCICERO is represented by its computational costs. For example, it required more than 200GB of RAM for handling all the entities and relationships extracted from 6.7M research articles. Such a limitation can be addressed by using efficient big data frameworks. We are now working in this direction by porting to Apache Spark[47] some of the most computational expensive modules. A first prototype is already available in the repository[48].

6. Future Research Directions

We expect that the KGs generated by SCICERO will be employed for supporting a variety of significant tasks. For example, KGs of research concepts similar to the output of our use case (i.e., CS-KG) have been used to provide terminology about specific research entities for key-phrase extrac-

[39]SciSpacy - `https://allenai.github.io/scispacy/`
[40]MeSH - `MedicalSubjectHeadings.`
[41]UMLS - `https://www.nlm.nih.gov/research/umls/index.html`
[42]GO - `http://geneontology.org/`
[43]RxNorm - `https://www.nlm.nih.gov/research/umls/rxnorm/index.html`
[44]HPO - `https://hpo.jax.org/app/`
[45]MSC - `https://mathscinet.ams.org/mathscinet/msc/msc2020.html`
[46]PhySH - `https://physh.aps.org/`
[47]`https://spark.apache.org/`
[48]`https://github.com/danilo-dessi/SKG-pipeline/tree/main/spark_entity_cleaning_and_mapping`

tion [66], to generate scholarly knowledge graph embeddings for classifying research publications [21], to develop systems for recommending research articles [24], and to study novel link prediction methodologies targeting the scholarly domain [67].

The resulting KGs may also be used for several other ambitious tasks such as research trend forecasting [68] (i.e., identifying and predicting emerging technologies and research trends) and hypothesis generation (i.e., suggesting to scientists new hypotheses to test). Indeed, these tasks require an accurate, large-scale, and machine-readable representation of the scientific domain. For example, for the research trend forecasting task, it is crucial to rely on a knowledge base that describes how research entities evolved and interacted in the past. CS-KG provides such knowledge since triples are associated to research papers published in a specific year. As far as the hypothesis generation task is concerned, the knowledge graph can be used to generate new triples (that can be seen as hypotheses to test) and verify their plausibility. For instance, similarly to what has been done in the *Transformer-based Validation Module*, the knowledge graph can be employed to train machine learning models to verify whether new triples aligns with or complement the current knowledge.

As next step, we will focus on improving the pipeline to overcome the limitations discussed in Section 5.4 and applying the produced KG to the described downstream tasks. More specifically, future work will involve i) a new design of the source code to make it fully parallel and scalable by using big data frameworks, ii) the exploration of a more fine-grained categorization of our entities (e.g., among instances of *cskg-ont:Material* we can identify images, text data, audio data, videos, and so on), iii) the use of the generated KG for a variety of tasks such as forecasting research dynamics, entity linking, and hypothesis generation.

7. Conclusion

In this paper we presented SCICERO, a novel approach for automatically generating scientific KGs from the titles and abstracts of research papers. SCICERO relies on a modular architecture that combines five extraction modules (the *DyGIEpp Module*, the *CSO Classifier Module*, the *Stanford Core NLP - OpenIE Module*, the *PoS Tag-based Relationships Extractor Module*, and the *Path-based Relationships Extractor Module*), three modules for integrating entities and relationships, and two validation modules.

Within SCICERO, entities are extracted from the input text and linked by exploiting both pre-defined and verb-based relations. They are then merged by mapping them to external resources such as CSO, Wikidata, and DBpedia. Moreover, deep learning transformers are employed for two different purposes: i) merging entities that are not linked to external resources by using the framework SentenceTransformers and encoding the research entities with the *paraphrase-distilroberta-base-v2* transformer model; ii) predict whether a certain triple is correct or not using the *scibert_scivocab_uncased* transformer model pretrained on 1.14M papers from Semantic Scholar. SCICERO also validates the resulting triples against an ontology that describes 179 possible relations between scientific entities.

SCICERO has been applied to a collection of about 6.7M research paper titles and abstracts from Computer Science ranging from 2010 to 2021. The evaluation that has been carried out concerns the ability of SCICERO to extract and recognize correct triples. It has been performed on three different domains: Machine Learning, Natural Language Processing, and Information Retrieval. The experiments shows that SCICERO yields a precision of 0.75, a recall of 0.79, and a F-measure of 0.77 on a gold standard of $3,600$ triples annotated by three humans experts. Last but not least, we publicly release the source code of SCICERO to make our results reproducible and to support the scientific community, as it can be easily adapted to other domains.

Acknowledgments

This work was partially supported by the ASTRID project (Fondazione di Sardegna, L.R. 7 agosto 2007, n. 7, CUP: F75F21001220007)

References

[1] J. Brainard, Scientists are drowning in covid-19 papers. can new tools keep them afloat, Science 13 (10) (2020) 1126.

[2] M. Y. Jaradeh, A. Oelen, K. E. Farfar, et al., Open research knowledge graph: Next generation infrastructure for semantic scholarly knowledge, in: Proceedings of the 10th International Conference on Knowledge Capture, 2019, pp. 243–246.

[3] L. Amos, D. Anderson, S. Brody, A. Ripple, B. L. Humphreys, Umls users and uses: a current overview, Journal of the American Medical Informatics Association 27 (10) (2020) 1606–1611.

[4] J. P. Tennant, H. Crane, T. Crick, J. Davila, et al., Ten hot topics around scholarly publishing, Publications 7 (2) (2019) 34.

[5] M. Wijkstra, T. Lek, T. Kuhn, K. Welbers, M. Steijaert, Living literature reviews, arXiv preprint arXiv:2111.00824 (2021).

[6] N. Guarino, D. Oberle, S. Staab, What is an ontology?, in: Handbook on ontologies, Springer, 2009, pp. 1–17.

[7] L. Vogt, Fair data representation in times of escience: a comparison of instance-based and class-based semantic representations of empirical data using phenotype descriptions as example, Journal of Biomedical Semantics 12 (1) (2021) 1–25.

[8] L. Ehrlinger, W. Wöß, Towards a definition of knowledge graphs., SEMANTiCS (Posters, Demos, SuCCESS) 48 (2016).

[9] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, et al., Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia, Semantic Web 6 (2) (2015) 167–195.

[10] H. Kitano, Nobel turing challenge: creating the engine for scientific discovery, npj Systems Biology and Applications 7 (1) (2021) 1–12.

[11] A. Gangemi, V. Presutti, D. Reforgiato Recupero, et al., Semantic web machine reading with fred, Semantic Web 8 (6) (2017) 873–893.

[12] J. L. Martinez-Rodriguez, I. Lopez-Arevalo, A. B. Rios-Alvarado, Openie-based approach for knowledge graph construction from text, Expert Systems with Applications 113 (2018) 339–355.

[13] J. D'Souza, S. Auer, Pattern-based acquisition of scientific entities from scholarly article titles, in: International Conference on Asian Digital Libraries, Springer, 2021, pp. 401–410.

[14] Y. Zhang, M. Wang, M. Saberi, E. Chang, From big scholarly data to solution-oriented knowledge repository, Frontiers in big Data (2019) 38.

[15] S. Auer, V. Kovtun, M. Prinz, et al., Towards a knowledge graph for science, in: 8th International Conference on Web Intelligence, Mining and Semantics, 2018.

[16] S. Mesbah, C. Lofi, M. V. Torre, A. Bozzon, G.-J. Houben, Tse-ner: An iterative approach for long-tail entity extraction in scientific publications, in: ISWC, Springer, 2018, pp. 127–143.

[17] Y. Luan, L. He, M. Ostendorf, H. Hajishirzi, Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction, in: Proceedings of the EMNLP 2018 Conference, 2018, pp. 3219–3232.

[18] D. Dessì, F. Osborne, D. R. Recupero, D. Buscaldi, E. Motta, Generating knowledge graphs by employing natural language processing and machine learning techniques within the scholarly domain, Future Generation Computer Systems 116 (2021) 253–264.

[19] K. Blagec, A. Barbosa-Silva, S. Ott, M. Samwald, A curated, ontology-based, large-scale knowledge graph of artificial intelligence tasks and benchmarks, arXiv preprint arXiv:2110.01434 (2021).

[20] P. Pramanik, R. K. Jana, Identifying research trends of machine learning in business: a topic modeling approach, Measuring Business Excellence (2022).

[21] F. Hoppe, D. Dessì, H. Sack, Deep learning meets knowledge graphs for scholarly data classification, in: Companion proceedings of the web conference 2021, 2021, pp. 417–421.

[22] X. Li, M. Daoutis, Unsupervised key-phrase extraction and clustering for classification scheme in scientific publications, arXiv preprint arXiv:2101.09990 (2021).

[23] F. Hoppe, D. Dessì, H. Sack, Understanding class representations: An intrinsic evaluation of zero-shot text classification, in: Workshop on Deep Learning for Knowledge Graphs (DL4KG@ ISWC2021), Vol. 3034 of CEUR Workshop Proceedings, CEUR-WS.org, 2021.

[24] A. Brack, A. Hoppe, R. Ewerth, Citation recommendation for research papers via knowledge graphs, in: International Conference on Theory and Practice of Digital Libraries, Springer, 2021, pp. 165–174.

[25] D. Dessì, F. Osborne, D. Reforgiato Recupero, D. Buscaldi, E. Motta, H. Sack, Ai-kg: an automatically generated knowledge graph of artificial

intelligence, in: International Semantic Web Conference, Springer, 2020, pp. 127–143.

[26] A. Salatino, F. Osborne, T. Thanapalasingam, E. Motta, The cso classifier: Ontology-driven detection of research topics in scholarly articles (2019).

[27] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, Dbpedia: A nucleus for a web of open data, in: The semantic web, Springer, 2007, pp. 722–735.

[28] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2019.

[29] K. K. Schuler, VerbNet: A broad-coverage, comprehensive verb lexicon, University of Pennsylvania, 2005.

[30] I. Beltagy, K. Lo, A. Cohan, Scibert: A pretrained language model for scientific text, in: EMNLP, Association for Computational Linguistics, 2019.

[31] S. Wang, The survey of joint entity and relation extraction, in: W. Cao, A. Ozcan, H. Xie, B. Guan (Eds.), Computing and Data Science, Springer Nature Singapore, Singapore, 2021, pp. 363–381.

[32] F. Ronzano, H. Saggion, Knowledge extraction and modeling from scientific publications, in: International workshop on semantic, analytics, visualization, Springer, 2016, pp. 11–25.

[33] P. Groth, A. Gibson, J. Velterop, The anatomy of a nanopublication, Information Services & Use 30 (1-2) (2010) 51–56.

[34] S. Angioni, A. Salatino, F. Osborne, D. R. Recupero, E. Motta, AIDA: A knowledge graph about research dynamics in academia and industry, Quantitative Science Studies 2 (4) (2021) 1356–1398.

[35] Y. Zhang, F. Zhang, P. Yao, J. Tang, Name disambiguation in aminer: Clustering, maintenance, and human in the loop., in: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, 2018, pp. 1002–1011.

[36] A. G. Nuzzolese, A. L. Gentile, V. Presutti, A. Gangemi, Conference linked data: the scholarlydata project, in: ISWC, Springer, 2016, pp. 150–158.

[37] S. Peroni, D. Shotton, F. Vitali, One year of the opencitations corpus, in: ISWC, Springer, 2017, pp. 184–192.

[38] P. Knoth, Z. Zdrahal, Core: three access levels to underpin open access, D-Lib Magazine 18 (11/12) (2012).

[39] B. Abu-Salih, Domain-specific knowledge graphs: A survey, Journal of Network and Computer Applications 185 (2021) 103076. doi:https://doi.org/10.1016/j.jnca.2021.103076.

[40] A. Moro, A. Raganato, R. Navigli, Entity linking meets word sense disambiguation: a unified approach, Transactions of the Association for Computational Linguistics 2 (2014) 231–244.

[41] J. R. Curran, S. Clark, J. Bos, Linguistically motivated large-scale nlp with c&c and boxer, in: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, 2007, pp. 33–36.

[42] F. Ronzano, H. Saggion, Dr. inventor framework: Extracting structured information from scientific publications, in: Discovery Science, 2015, pp. 209–220.

[43] D. O'Donoghue, Y. Abgaz, D. Hurley, F. Ronzano, Stimulating and simulating creativity with dr inventor, in: Proceedings of the Sixth International Conference on Computational Creativity June 2015, Brigham Young University, Utah, 2015, pp. 220–227, this is the postprint version of the published chapter.

[44] I. Augenstein, M. Das, S. Riedel, L. Vikraman, A. McCallum, SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications, in: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 546–555.

[45] A. Li, X. Wang, W. Wang, A. Zhang, B. Li, A survey of relation extraction of knowledge graphs, in: J. Song, X. Zhu (Eds.), Web and Big Data, Springer International Publishing, 2019, pp. 52–66.

[46] P. Labropoulou, D. Galanis, A. Lempesis, et al., Openminted: A platform facilitating text mining of scholarly content, in: 11th International Conference on Language Resources and Evaluation (LREC 2018), Paris, France, 2018.

[47] R. A. Al-Zaidy, C. L. Giles, Extracting semantic relations for scholarly knowledge base construction, in: IEEE 12th ICSC, 2018, pp. 56–63.

[48] T. Jiang, T. Zhao, B. Qin, T. Liu, N. Chawla, M. Jiang, The role of "condition": A novel scientific knowledge graph representation and construction model, Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2019).

[49] D. Wadden, U. Wennberg, Y. Luan, H. Hajishirzi, Entity, relation, and event extraction with contextualized span representations, ArXiv abs/1909.03546 (2019).

[50] Q. Wang, M. Li, X. Wang, N. N. Parulian, G. Han, J. Ma, J. Tu, et al., Covid-19 literature knowledge graph construction and drug repurposing report generation, ArXiv abs/2007.00576 (2021).

[51] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, Semantic web 8 (3) (2017) 489–508.

[52] A. Bordes, N. Usunier, A. García-Durán, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: NIPS, 2013, pp. 2787–2795.

[53] Z. Sun, Z.-H. Deng, J.-Y. Nie, J. Tang, Factorizing yago: scalable machine learning for linked data, in: ICLR, 2019, pp. 271–280.

[54] T. Dettmers, P. Minervini, P. Stenetorp, S. Riedel, Convolutional 2d knowledge graph embeddings, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 32, 2018.

[55] A. Borrego, D. Ayala, I. Hernández, C. R. Rivero, D. Ruiz, Cafe: Knowledge graph completion using neighborhood-aware features, Engineering

Applications of Artificial Intelligence 103 (2021) 104302. doi:https://doi.org/10.1016/j.engappai.2021.104302.

[56] L. Yao, C. Mao, Y. Luo, Kg-bert: Bert for knowledge graph completion, arXiv preprint arXiv:1909.03193 (2019).

[57] M. Y. Jaradeh, K. Singh, M. Stocker, S. Auer, Triple classification for scholarly knowledge graph completion, in: Proceedings of the 11th on Knowledge Capture Conference, 2021, pp. 225–232.

[58] D. Wadden, U. Wennberg, Y. Luan, H. Hajishirzi, Entity, relation, and event extraction with contextualized span representations, in: Proceedings of the 2019 Joint Conference EMNLP-IJCNLP, 2019, pp. 5788–5793.

[59] A. Salatino, F. Osborne, E. Motta, Cso classifier 3.0: a scalable unsupervised method for classifying documents in terms of research topics, International Journal on Digital Libraries 23 (1) (2022) 91–110.

[60] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, et al., The stanford corenlp natural language processing toolkit, in: Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations, 2014, pp. 55–60.

[61] K. Toutanova, D. Klein, C. D. Manning, Y. Singer, Feature-rich part-of-speech tagging with a cyclic dependency network, in: Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, 2003, pp. 252–259.

[62] G. Angeli, M. J. J. Premkumar, C. D. Manning, Leveraging linguistic structure for open domain information extraction, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2015, pp. 344–354.

[63] D. Chen, C. D. Manning, A fast and accurate dependency parser using neural networks, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 740–750.

[64] P. N. Mendes, M. Jakob, A. García-Silva, C. Bizer, Dbpedia spotlight: shedding light on the web of documents, in: Proceedings of the 7th international conference on semantic systems, 2011, pp. 1–8.

[65] J. L. Fleiss, J. C. Nee, J. R. Landis, Large sample variance of kappa in the case of different sets of raters., Psychological bulletin 86 (5) (1979) 974.

[66] X. Li, M. Daoutis, Unsupervised key-phrase extraction and clustering for classification scheme in scientific publications, in: A. P. B. Veyseh, F. Dernoncourt, T. H. Nguyen, W. Chang, L. A. Celi (Eds.), Proceedings of the Workshop on Scientific Document Understanding co-located with 35th AAAI Conference on Artificial Inteligence, SDU@AAAI 2021, Virtual Event, February 9, 2021, Vol. 2831 of CEUR Workshop Proceedings, CEUR-WS.org, 2021.

[67] M. Nayyeri, G. M. Cil, S. Vahdati, F. Osborne, A. Kravchenko, S. Angioni, A. Salatino, D. R. Recupero, E. Motta, J. Lehmann, Link prediction of weighted triples for knowledge graph completion within the scholarly domain, IEEE Access 9 (2021) 116002–116014.

[68] A. A. Salatino, F. Osborne, E. Motta, Augur: forecasting the emergence of new research topics, in: Proceedings of the 18th ACM/IEEE on joint conference on digital libraries, 2018, pp. 303–312.