information

MDPI

Article

The Geranium Platform: A KG-Based System for Academic Publications

Giovanni Garifo [1,*], Giuseppe Futia [2], Antonio Vetrò [1] and Juan Carlos De Martin [1]

1 Nexa Center for Internet and Society, Department of Control and Computer Engineering (DAUIN), Politecnico di Torino, 10129 Turin, Italy; antonio.vetro@polito.it (A.V.); demartin@polito.it (J.C.D.M.)
2 Graph Aware S.r.l., 73100 Lecce, Italy; giuseppe.futia@polito.it
* Correspondence: giovanni.garifo@polito.it

Abstract: Knowledge Graphs (KGs) have emerged as a core technology for incorporating human knowledge because of their capability to capture the relational dimension of information and of its semantic properties. The nature of KGs meets one of the vocational pursuits of academic institutions, which is sharing their intellectual output, especially publications. In this paper, we describe and make available the Polito Knowledge Graph (PKG) –which semantically connects information on more than 23,000 publications and 34,000 authors– and Geranium, a semantic platform that leverages the properties of the PKG to offer advanced services for search and exploration. In particular, we describe the Geranium recommendation system, which exploits Graph Neural Networks (GNNs) to suggest collaboration opportunities between researchers of different disciplines. This work integrates the state of the art because we use data from a real application in the scholarly domain, while the current literature still explores the combination of KGs and GNNs in a prototypal context using synthetic data. The results shows that the fusion of these technologies represents a promising approach for recommendation and metadata inference in the scholarly domain.

Keywords: Knowledge Graph; Graph Neural Network; scientific publications

check for updates

Citation: Garifo, G.; Futia, G.; Vetrò, A.; De Martin, J.C.; The Geranium Platform: A KG-Based System for Academic Publications. *Information* **2021**, *12*, 366. https://doi.org/10.3390/info12090366

Academic Editors: Pierpaolo Basile and Annalina Caputo

Received: 8 August 2021
Accepted: 4 September 2021
Published: 8 September 2021

1. Introduction

Academic institutions invest relevant resources in collecting, organizing and preserving their intellectual output in institutional repositories. Among the digital artifacts hosted therein, publications are the most relevant and the standard organisation of their metadata is fundamental not only for evaluation and dissemination, but also for improving the search, given the quantity of research papers published every year. In addition to traditional search, harnessing metadata for modeling information in a semantic way is a new highly demanded requirement which cannot be met by traditional search engines. This feature is increasingly relevant in a context of global challenges for modern science, as the covid-19 pandemic clearly showed [1–3].

Semantic technologies and Knowledge Graphs (KGs) can be helpful for these purposes, given the capability of KGs to capture the relational dimension of information and of its semantic properties. In fact, Scholarly Knowledge Graphs (SKGs) have been recently experimented by academic institutions. One of the most prominent examples is Wiser [4], a search engine system developed by the University of Pisa, which is built upon an SKG composed of approximately 65,000 publications. Another well-known success case is the Open Academic Graph (https://www.openacademic.ai/oag/ (accessed on 3 September 2021)), a publicly released SKG built by Microsoft and AMiner that supports the study of citation networks and publication contents.

Compared to traditional data sources (e.g., relational databases) for the storage and retrieval of publications, SKGs enable more advanced features [5], such as recommendations based on the semantic properties of texts and of their metadata. This novel type of recommendation engine makes use of Graph Representation Learning (GRL) techniques, among which Graph Neural Networks (GNNs) have received particular attention [6], as they are able to learn latent representations without any prior knowledge, other than the graph structure. However, their use for recommendation purposes is in a prototypal phase and still has to be properly investigated.

We contribute to this pioneering research field and we present Geranium, a semantic platform built on top of the Polito Knowledge Graph (PKG), a new SKG—built by us—that offers semantic search and recommendations of semantically affine publications and researchers at Politecnico di Torino. While the semantic search is provided with state-of-the-art Natural Language Processing (NLP) techniques in combination with the well-established DBpedia, the recommendations are enabled by a novel combination of KG embedding (KGE) techniques with GNN methods. In our contribution, we formally investigate and validate the performance of this combination in a real application scenario. In particular, our analysis focuses on the performance of the link prediction mechanism that is exploited to enable the recommendation features. The results show that the fusion of KGs and GNNs represents a promising approach to creating recommendations in the scholarly domain.

Through the Geranium platform, different stakeholders, such as researchers, policy-makers, and enterprises, have available an interactive tool for exploring the activities and research results of the Politecnico while fostering new opportunities for collaboration with researchers. The PKG can be explored and visualized through a public web application or by querying an open SPARQL endpoint. The Geranium source code and datasets are openly available on GitHub (https://github.com/geranium-project, accessed on 26 August 2021).

We detail the work as follows. Section 2 gives a brief introduction to KGs and GNNs. Section 3 presents the related work. Section 4 introduces the building blocks of the Geranium platform. Section 5 goes into the details of the provision of the PKG. In Section 6 are presented the advanced services and tools built upon the PKG. Section 7 explains the validation process of the link prediction algorithm employed for the recommendation task. In Section 8, we discuss our conclusions, and the path for future work is outlined.

2. Background and Notation

Knowledge Graphs (KGs) have become effective tools for capturing and organizing a large amount of structured and multi-relational data that can be explored by employing query mechanisms. A KG can be formalized as follows: $G = \{E, R, T\}$, where G is a labeled and directed multigraph and E, R, T are the set of entities, relations, and triples, respectively. In the semantic regime, a fact is formalized as a triple $(s, r, o) \in T$, where $s, o \in E$ are two entities—the subject and the object of the fact, respectively—while $r \in R$ is the relation that connects s and o. In the Semantic Web field, the KG facts are modeled using the Resource Description Framework (RDF) [7]. The RDF data model exploits web technologies, such as Uniform Resource Identifiers (URIs), to define entities and their relationships, and it provides a formal notion of meaning (semantics) that sets up the basis for founded deductions.

Knowledge Graph Embeddings (KGEs) are the result of specific representation learning models applied to KGs. The goal of KGE models is to embed the KG components—entities and relations—into a continuous and low-dimensional vector space. The latent factors projected into KGEs have an essential role in analyzing and mining additional *soft*-knowledge in KGs. In fact, for each entity pair $s, o \in E$ and any relation $r \in R$, it is possible to determine if a statement (s, r, o) is true according to the embeddings learned by the KGE techniques. The KGE approaches define a *scoring function* $f_{sco}(s, r, o)$ for each KG statement (s, r, o). In general, KGs comprise only true statements, while non-existing statements can be considered either missing or false. For these reasons, a closed-world assumption (the closed-world assumption (CWA) in the KG context means that lack of knowledge does imply falsity; in other words, missing facts in the KG are automatically considered false; the opposite perspective is the open-world assumption (OWA), according to which missing knowledge does not automatically imply falsity) is deemed to address this ambiguity in categorizing non-existing statements. As a consequence, the scoring function $f_{sco}(s,r,o)$ returns a higher score for existing statements and a lower score otherwise. DistMult [8] is one of the most common and simple factorization methods used as a scoring function. DistMult allows one to compute a statement score via the following transformation: $score_{(s,r,o)} = f_{sco}(s,r,o) = e_s^T R_r e_o$, in which e_s and e_o are the embeddings of the source and the destination node, and $R_r \in \mathbb{R}^{d\times d}$ is a relation-specific linear transformation.

KGE techniques encode the interactions between entities and relations through models that are not natively built for encoding the local graph structure. However, a novel family of neural architectures has been proposed to address this limitation. In this context, Graph Neural Networks (GNNs) are becoming the key framework for learning the latent representation of graph-structured data. GNNs are based on neural architectures that are designed by following a graph data topology, where the weighted connections of the neural network match the edges available in the graph structure. The most-adopted class of GNNs is known as Graph Convolutional Networks (GCNs).

The major novelty of GCNs is the use of self-learnt filters that aggregate the representation of adjacent nodes to obtain the embeddings and update them from one layer to the other: The embeddings obtained at the end of the network are the result of the node neighbors' features that are accumulated through the previous layers. GCNs have proved to work well with graph data in which the neighbors' characteristics are fundamental for a meaningful representation of nodes, such as in social graphs or molecular networks [9]. In a GCN, the representation of nodes at each layer is updated via the following transformation: $h_i^{l+1} = f_{agg}(h_i^l, \left\{h_j^l\right\}_{j\in N_i})$, in which:

- $h^l \in \mathbb{R}^{n\times d}$ is the hidden representation of the nodes in the l-th layer;
- $h^{l+1} \in \mathbb{R}^{n\times d}$ is the hidden representation of the nodes in the $l+1$-th layer;
- $\left\{h_j^l\right\}_{j\in N_i}$ includes all of the hidden representations of the neighbors N_i of node i in the l-th layer;
- f_{agg} is an aggregation function that establishes how to accumulate the representations of N_i into the node i.

GCNs fall short when dealing with graph data where different relations are used to connect nodes, as in knowledge graphs. The Relational Graph Convolutional Network [10] (R-GCN) is a new kind of GCN that is focused on modeling multi-relational graphs composed of labeled and directed edges, and it is thus particularly capable of embedding both the nodes and relations of a KG. In an R-GCN, different sets of parameters are learned for different relations. At each step inside the network, the feature vector of a node is updated by convolving its first neighbors' features with a convolutional filter that is different based on the kind of relation that connects the nodes. The transformation used to update the representation of a node at the l-th layer is the following: $h_i^{(l+1)} = \sigma\left(W_0^{(l)} h_i^{(l)} + \sum_{r\in\mathcal{R}} \sum_{j\in\mathcal{N}_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)}\right)$, where:

- $h_i^{(l)}$ is the hidden representation of the nodes in the l-th layer;
- $W_0^{(l)}$ is the learned kernel for the self-loop relation;
- $\mathcal{N}_i^r$ is the set of indices of the neighbors of node i under the relation $r \in \mathcal{R}$, with $\mathcal{R}$ being the set of all of the relations present in the graph;
- $W_r^{(l)} \in \mathbb{R}^{d^{(l+1)}\times d^{(l)}}$ is the learned filter for the relation r;
- σ is a non-linear activation function and $c_{i,r}$ is a normalization constant, commonly initialized to $|\mathcal{N}_i^r|$.

GNN models and KGE techniques can be incorporated into an end-to-end architecture. Such kinds of architectures are referred to as Graph Auto-Encoders (GAE). The GNN models play the role of the encoder, which produces an enriched representation of all entities within the KG, accumulating the neighborhood features; the KGE techniques play the role of the decoder, exploiting this enriched representation to reconstruct the edges within the KG. We introduce the GAE architecture used in this paper in Section 5.2.

3. Related Work

The current work is related to the adoption of KGs in order to model scholarly data and build advanced services on top of this semantic information. In particular, our contribution embraces two well-known research areas in the Semantic Web field: (i) the automatic extraction of structured knowledge from textual sources to construct KGs; (ii) the adoption of KGs to mine latent information that is used for statistical analysis and recommendation tools.

With regards to the first area, many solutions [11] and frameworks [12] have been proposed to generate scientific KGs in the academic field. For a global and rapidly occurring phenomenon such as the COVID-19 pandemic, the adoption of crowd-sourced KGs can be a valid option [1]. However, in the case of scientific knowledge that is built on top of the metadata of existing publications, such as in our scenario, the generation of KGs in a controlled manner is effective for maintaining data consistency and applying automatic techniques in a supervised environment. One of the most challenging tasks in this context is extracting domain topics from the scientific literature to support the construction of RDF triples. Recent work [13] has shown how deep neural networks can be helpful for the extraction of triples from text. However, approaches based on pure ontology [14] and NLP [15,16] have been shown to be effective, particularly in the scholarly domain. We decided to employ a pure NLP solution called TellMeFirst (TMF) based on our previous work [17]. Compared to the work of Salatino et al. [14], which is focused on a specific domain, this solution allowed us to employ a general-purpose reference (DBpedia) that can be adapted to the variety of topics covered in the publications of Polito. Moreover, in comparison to the solutions proposed by Angeli et al. [15] and Labropoulou et al. [16], which are based on text data mining, the TMF topic extraction pipeline is based on similarity comparison.

With regards to the second area, which is related to recommendation tools, many efforts have been made to build a new generation of systems based on KGs [18,19]. In particular, solutions based on such technologies and aimed at researchers and academic institutions have been actively researched [4,20,21]. One of the most promising approaches is the use of KG embeddings to compute recommendations [22]. Our solution goes in this direction and employs a Graph Auto-Encoder (GAE) architecture based on a state-of-the-art GNN to learn KG embeddings that are used to empower the recommendation system. Different approaches have been explored by other authors, such as the one based on graph kernels by Kanakaris et al. [20] or the one based on semantic similarity measures by Vahdati et al. [21]. A GNN-based approach allows us to leverage the graph structure itself to obtain meaningful representations for these KG embeddings. Other approaches in the field, such as the one followed by Ameen et al. [23], are based on the use of ontologies to model knowledge about user–item interactions and compute recommendations. However, such a solution was not feasible for our scenario due to the lack of comprehensive information in the available datasets that could support a pure knowledge-driven recommender.

4. The Geranium Platform

The Geranium platform is based on specific components for ingesting, integrating, enriching, and publishing scholarly data, including information on publications, authors, and journals. The original data are mapped into RDF with respect to an ontology network in order to build the Polito Knowledge Graph (PKG). In the following subsections, we describe the main ingredients of the platform.

4.1. Data Providers

Two different data providers serve the content of the PKG. The main content is based on the information available from the Institutional Research Information System (IRIS) (https://www.cineca.it/sistemi-informativi-universita/ricerca/iris-la-gestione-della-ricerca (accessed on 3 September 2021, available in italian)) instance available at Polito, which collects all metadata and contents of the publications released by the university's researchers.

This main content is enriched by means of a second data provider, the DBpedia repository, which is available as the RDF. Considering its encyclopedic and generalist vocation, DBpedia is particularly suitable for integrating publications coming from different scientific disciplines.

Currently, IRIS is the research repository that is the most frequently used by the Italian universities and research centers. The system was developed by the CINECA Consortium, which is made up of 69 Italian universities and 23 research institutions. Thus, IRIS represents the de-facto standard for archiving, managing, and publishing the research output in the Italian research landscape. At the time of writing, the Polito instance contains 108,077 publications (of which approximately 38% are available in open access), 8328 authors, 190,903 keywords, and 7581 journals. We have chosen IRIS as the primary data provider because of its wide adoption, thus allowing the Geranium platform to ingest new data sources from different research institutions other than Polito. Moreover, IRIS uses the DublinCore ontology and terms to describe its content, thus giving us a standard knowledge description about a publication and its metadata.

Here, we list the most useful IRIS publication metadata, which we selected as input data for the generation of the Geranium ontology:

1. The *title*, *abstract*, *date*, and *identifier*: Information related to the publication, including an alphanumeric code that identifies a publication in the IRIS system and more traditional textual data related to the content of the publication and the publication date. Moreover, multiple *keywords* can be included by the authors in order to label the publication.
2. The author's *name*, *surname*, and *identifier*: If an author is part of the Polito staff, they are identified by an ID in the IRIS system, in addition to their name and surname. In the case of external authors, the identification code is not available.
3. The co-authors' *names*, *surnames*, and *identifiers*: IRIS allows one to distinguish the first author from the other contributors of the publication. As with the first author's metadata, only the Polito staff has a unique identifier in the management system.
4. The journal *title* and the *ISSN*: A specific conference venue or journal is connected to the publication. National and international conferences do not have a unique code associated with them, while in many cases, journals are identified by a unique ISSN code.

To overcome some of the limitations of IRIS, we used DBpedia, which represents structured information from Wikipedia. DBpedia is built and maintained by an open community; it provides a query service for its data and creates links to external datasets on the web. We took particular advantage of the latter characteristic: Using DBpedia, we were able to work with even more metadata than were made available by IRIS.

The DBpedia ontology is composed of 685 classes described by 2795 different properties, and currently, it contains about 4,233,000 instances. The most common classes by number of instances are:

1. *Person*, with 1,450,000 instances;
2. *Place*, with 735,000 instances;
3. *Work*, with 411,000 instances;
4. *Species*, with 251,000 instances;
5. *Organisation*, with 241,000 instances.

These instances are linked by more than 9,500,000,000 triples (data available from the official DBpedia website: http://wikidata.dbpedia.org/services-resources/ontology (accessed on 7 August 2021)). Thus, DBpedia is one of the biggest linked open datasets available to date.

4.2. Ontology Network

The local schemas from the input data sources can be mapped in the global schema of the ontology. For each type of entity, we defined the following classes:

1. *Publication*;
2. *Author*;
3. *Journal*;
4. *AuthorKeyword*;
5. *TMFResource*.

The *Publication*, *Author*, and *Journal* classes are directly mapped from the IRIS metadata to the PKG ontology. The *AuthorKeyword* class represents the keywords that are freely added by the authors to label the publication. These keywords are included in the KG; however, they are not part of a controlled vocabulary, and for this reason, they are not useful for connecting publications focused on the same topics. On the other hand, the *TMFResource* class represents the relevant semantic topics extracted from the publications' abstracts, which are directly mapped from the resources available in DBpedia.

Data and object properties defined in RDFS [7], FOAF [24], and DCMI [25] are used to semantically connect instances of the previous classes. Figure 1 depicts the schema of the PKG.

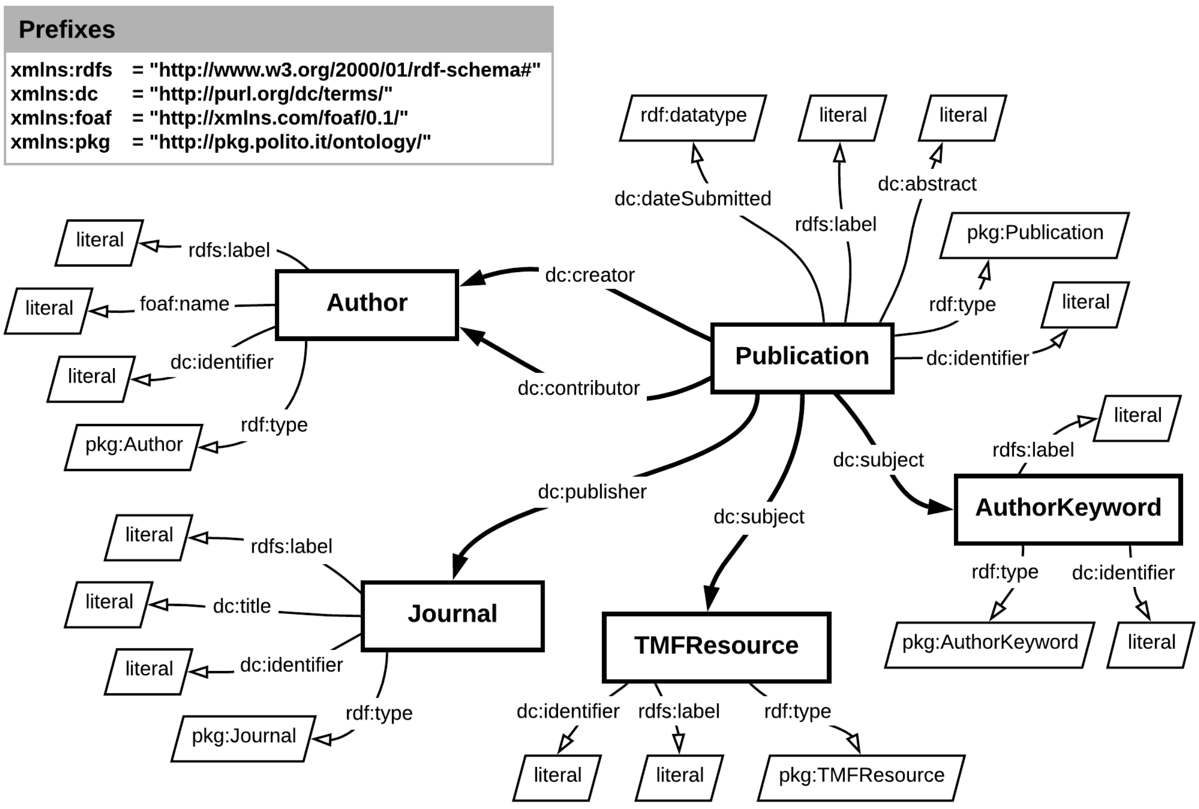


Figure 1. Class schema of the Geranium ontology.

4.3. Platform Architecture

The Geranium platform was developed according to the common standards for modern web applications based on microservices and containers. Figure 2 provides a high-level overview of the software architecture and its key components.

The first part of the architecture, which is depicted on the left side of the diagram, is responsible for the data provisioning and the KG ingestion. These operations are performed by following an offline pipeline. When processing the JSON records of the IRIS publications through the pipeline, the REST API of the classification service of TellMeFirst is used to enrich the publications metadata with DBpedia entities (steps 1 and 2 of the pipeline). The resulting RDF representations are then processed by the graph enhancer, which computes the graph embeddings and predicts new links that are added to the graph in the form of RDF statements (steps 3–6). These statements are used to empower the recommendation service. The enhanced KG obtained represents the Polito Knowledge Graph, which is then saved in a triplestore (step 7), thus allowing the system backend to gather the necessary data in order to fulfill the clients' requests.

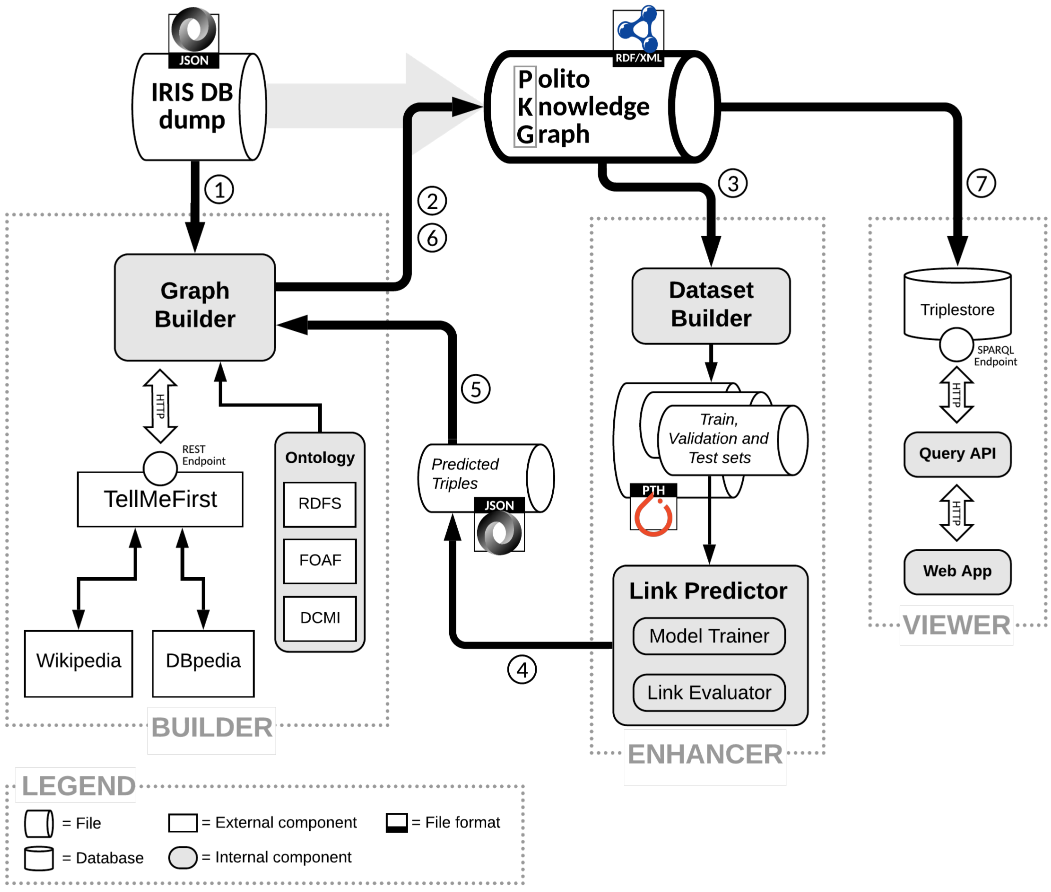


Figure 2. The pipeline of the Geranium architecture.

The second part of the architecture is based on a backend architecture that holds the RDF representation of the graph in a triplestore and exposes the Geranium services (see Section 6) through the REST and SPARQL APIs, as well as on a frontend architecture based on a Progressive Web App made available through a web server. The Geranium platform thus allows both developers, through its publicly available APIs, and users, through its webeb application, to explore the Polito Knowledge Graph.

4.4. Implementation Details and Code Availability

We implemented the architecture pipeline depicted in Figure 2 using the following languages, frameworks, and technologies.

The Builder (see Section 5.1) is implemented as a Python module using the *rdflib* (https://github.com/RDFLib/rdflib (accessed on 3 September 2021)) library to create and manage an in-memory RDF representation of the graph. The module takes as input the path to the JSON dump of IRIS, the ontology, and the JSON responses received by the TellMeFirst REST API in order to produce an RDF graph as output. The module supports some options that allow specific functionalities to be triggered, such as:

1. The update of an already existing RDF graph with new statements that are used, for example, to add the predicted triples.
2. The number of topics that must be extracted from each abstract by the TMF.
3. The retrieval of the topic images, which are scraped from Wikimedia Commons (https://commons.wikimedia.org/ (accessed on 3 September 2021)).

In order to increase the performance of the KG generation process, the Builder implementation is based on a *ThreadPoolExecutor* and *AsyncIO* coroutines.

The Enhancer (see Section 5.2) is composed if three submodules—the Dataset Builder, the Model Trainer, and the Link Evaluator. Like the Graph Builder, the Dataset Builder is also implemented as a Python module that processes the RDF graph triples with *rdflib*. The Model Trainer is implemented using Pytorch and the Deep Graph Library (DGL) [26], which implements the message-passing paradigm generalized for GNNs, allowing one to define the two basic operations for the construction of the per-edge messages and the per-node aggregation. The Encoder component of the Model Trainer is built upon an R-GCN [10] composed of two hidden layers, with the first-layer convolutional filters being $500 \times N$, where N is the number of nodes in the training graph, and the second-layer filters being 500×500. The embeddings obtained as the output of the forward pass are shaped as 1×500 row vectors. For the input layer, one-hot encoded feature vectors are used, acting as a mask for the selection of the corresponding column in the per-relation convolutional filters. The per-node normalization constants are initialized to the inverse of the nodes' indegree. The Link Evaluator is implemented as a Python module that loads the trained model (used as the Auto-Encoder) and computes the predictions, which are exported as JSON files and added to the graph by the Graph Builder.

Regarding the Viewer Module (see Section 5.3), the API microservice was developed using Flask, a Python web micro-framework based on the WSGI transmission protocol. The API is publicly available through a reverse proxy that acts as a TLS termination endpoint and WSGI adaptation layer.

Each of the endpoints are based on the same workflow:

1. checks the received query parameters;
2. uses these parameters to build a, SPARQL query on the fly;
3. sends a request to the correct SPARQL endpoint and waits for the response;
4. returns the response (JSON) to the client that sent the request.

In the current version, the requests to the SPARQL endpoints are made by blocking HTTP requests. We leave the implementation of asynchronous I/O for this task for future releases.

The triplestore of choice for Geranium is Blazegraph. The triplestore can be queried through Blazegraph's built-in SPARQL endpoint, which we made publicly available using the same Apache2 reverse proxy used for the Geranium REST API.

The Geranium web frontend is implemented as a Progressive Web App (PWA) built upon the Angular and Ionic Framework that follows the Model–View–Controller pattern. We chose to implement the frontend as a PWA to create a responsive UI with a mobile-app-like user experience. In order to speed up the development process, we used the pre-built UI components made available by the Ionic Framework. We embedded in the application a reference to a self-hosted Matomo Analytics [27] instance, which we used to gather fully anonymous statistics about the PWA usage.

The source code of all of the components of the Geranium platform is publicly available on GitHub with the following link: https://github.com/geranium-project (accessed on 3 September 2021).

5. KG Provisioning

The provisioning step encompasses the ETL process in order to incorporate the source data into the KG. Then, the KG is published using a SPARQL endpoint and different types of REST API services. The ETL process is performed by two different components of the Geranium platform—the Builder and the Enhancement modules—while the publishing of the KG is entrusted to the Viewer Module.

5.1. Data Ingestion

The first phase of the ETL process, which is called Data Ingestion, is enabled by the Builder Module, and it deals with: (i) the extraction of metadata from the IRIS system, which distillate the relevant information; (ii) the mapping and the translation of the information derived from the metadata extraction into the RDF with respect to the ontology network (see Section 4.2); (iii) the integration into the KG of semantic topics provided by DBpedia.

The Builder Module analyzes each record in the JSON dump of the IRIS metadata, which currently include 23,268 records for the time period of 2008–2018. Taking into account the time required for the evaluation of a scientific paper, we decided to take only the publications that have been accepted and that are no longer subject to modification, thus limiting the upper publication date to 2018. Moreover, we decided to not take into account publications older than 10 years to obtain a fairly manageable size for the KG.

Then, the Builder maps the information contained in the record with the concepts and the properties defined by the ontology. For instance, considering a publication including more than one contributor, the ontology allows one to distinguish the main author and the other co-authors by employing different object properties.

At the conclusion of this extraction and mapping process, the resulting RDF is augmented by TellMeFirst (TMF) [17]. TMF is used to semantically classify the abstracts of the publications by extracting the main topics from texts as DBpedia entities. TMF is particularly suitable for our task for two fundamental reasons. On the one hand, TMF employs a general-purpose reference that can be adapted to the variety of topics covered in the publications of Polito. On the other hand, we are able to dynamically change the number of topics extracted from a document; this feature is particularly useful for understanding the impact of semantic topics for the link prediction task (see Section 7.3). The TMF model was trained on a text corpus derived from the Wikipedia page associated with each DBpedia entity. This corpus was translated into a vector representation during the training process by employing the TF-IDF [28] algorithm. To perform the classification task, the trained model transformed the abstract of a publication into a vector representation using the same algorithm and compared it with the Wikipedia corpus by using the cosine similarity. A publication abstract is classified by using the DBpedia/Wikipedia entities that achieve the best scores in terms of similarity. The URIs of such entities are added into the KG as new relations.

The output of the Builder is an initial version of the Polito Knowledge Graph (PKG), which includes a coherent description of uniquely identified and semantically connected authors, publications, journals, and semantic topics extracted from the abstracts of the paper.

The enrichment of the metadata provided by IRIS with the semantic topics computed by TMF is a crucial step in providing a system for searching for publications by using a topic-based engine, which constitutes an advanced service provided by the Geranium platform (see Section 6.1).

Table 1 summarizes the PKG features after the topic extraction performed by TMF. It reports the size of the graph, showing the number of entities categorized under each class and the number of edges for each relation type.

Table 1. Number of entities and edges in the Polito Knowledge Graph.

Polito Knowledge Graph				
Number of entities per class				
Publication	Author	Journal	TMFResource	AuthorKeyword
23,268	34,886	3211	16,988	41,807
Number of edges per relation				
Creator	Contributor	Publisher	Subject (TMF)	Subject (Keywords)
23,268	80,819	11,243	107,093	77,492

5.2. *Link Enrichment*

The second phase of the provisioning is called Link Enrichment, and it is performed by the Enhancement Module. This is the most sophisticated component of the platform architecture, as it feeds the recommender system of Geranium (see Section 6.2).

The goal of this component is to improve the first version of the Polito Knowledge Graph (PKG) generated during the Data Ingestion phase carried out by the Builder Module. This improvement consists of a collection of novel links in the PKG: these links are predicted by the Data Enhancement by exploiting a Graph Neural Network (GNN) model trained on the initial version of the graph. These computed links, which were not available in the initial version of the PKG, can be interpreted as new suggestions that, for instance, identify researchers who deal with similar topics, but have never been co-authors, or non-explicit semantic topics that are dealt with in a journal.

The functioning of the link prediction is based on a machine learning pipeline that includes three different phases: (i) the data preparation; (ii) the training step; (iii) the evaluation step. Each of these steps is tackled by the Enhancer module through different building blocks, which are, respectively: (i) the Dataset Builder, (ii) the Model Trainer, and (iii) the Link Evaluator.

5.2.1. The Dataset Builder

The Dataset Builder is in charge of translating the PKG into a usable dataset for the Model Trainer. Initially, the Dataset Builder selects the subset of entities (and the related connections), which introduce ambiguous information into the dataset. For instance, during this stage, the keywords defined by the authors are removed from the PKG because their ambiguity represents noise, which can lead to incorrect predictions. At the same time, all of the external contributors and co-authors are discarded because they do not have a unique identifier in the dataset.

After this filtering in the entity selection, the PKG statements require a numerical translation of the data. Indeed, the RDF representation of PKG statements is not suitable to be used for the link prediction task. Therefore, the Dataset Builder assigns numerical representations (one-hot encoded vectors) to the nodes and the edges of the PKG. Starting from this numerical representation, the training process will be able to assign embeddings to the entities of the PKG, which can be exploited for the link prediction. In order to generate such one-hot encoded vectors for the Model Trainer, the Dataset Builder produces the following PKG features:

- the number of nodes, equal to the number of RDF entities in the PKG;
- the number of different relations that link the PKG entities;
- the number of node labels, where each label refers to a class of the network ontology;
- a list of edges representing the RDF statements. Each edge is a tuple composed of three elements: the node index of the subject, the node and the object, and the index of the predicate;
- a list of node-specific normalization constants.

In order to generate these features from the RDF representation, the Dataset Builder leverages two look-up hash tables—one for the nodes and one for the edges—which are created by starting from the RDF statements and the PKG ontology. These tables are accessed by URI and allow retrieval of the corresponding entity or relation index. During the generation of the hash tables, the Dataset Builder keeps track of the total number of nodes and relations and builds a label list that stores in the i-th element the class (label) of the i-th node. Once the look-up tables and the other data structures are built, the Dataset Builder starts to process the RDF statements that compose the graph, thus building the data structures needed by the model trainer. An example of the look-up process for the creation of the data structure is shown in Figure 3.

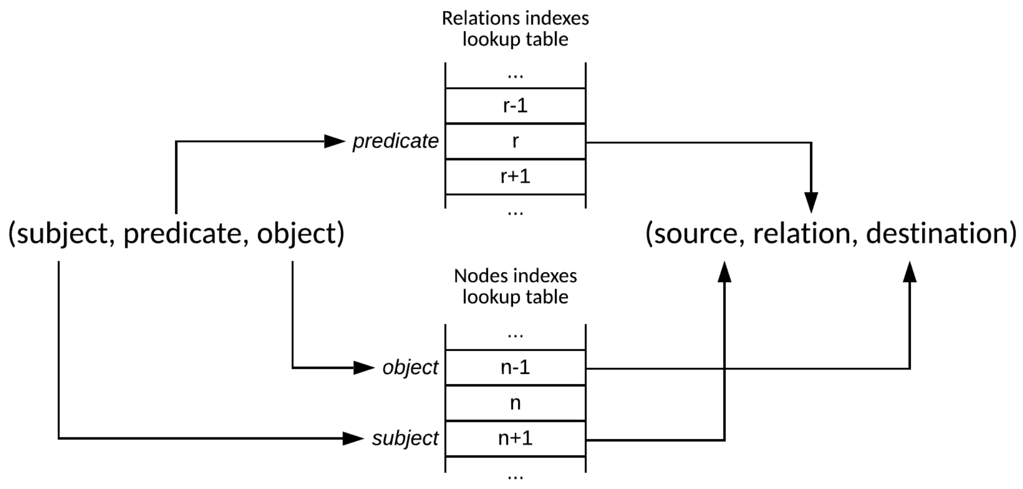

Relations indexes lookup table
...
r-1
r
r+1
...
predicate
(subject, predicate, object)
(source, relation, destination)
Nodes indexes lookup table
...
object
n-1
n
subject
n+1
...

Figure 3. Process of translation from the RDF statement to an edge tuple.

The edge list is the key data structure for the prediction task. To enable the training and the evaluation pipeline, this list is split into three disjoint sets—the training set, the evaluation set, and the test set—making sure that, in the training set, for each node, there is at least one edge for every relation type in the PKG. This aspect is mandatory in order to obtain meaningful embeddings because the Model Trainer learns the vector representations of the nodes by embedding the nodes' neighbors' features. As a consequence of this, it is crucial to have a neighborhood structure in the training graph that is similar to the one in the full graph.

Once this initial sampling has been performed, the remaining training edges are randomly taken. To cope with the requirement above, the edge list is split into approximately 90% of the tuples for training, 5% for validation, and 5% for testing. The dataset obtained is serialized and saved, so the Model Trainer does not require one to create the dataset from scratch at each run. Moreover, since some of the edges are randomly picked, this allows one to evaluate the different hyperparameter values on the same sets of triples. Table 2 summarizes the characteristics of the dataset obtained by the Dataset Builder. The size of the dataset—in particular, the number of nodes and edges—can be compared to the size of the initial version of the graph, whose statistics are available in Table 1. As can be seen, more than half of the nodes—the ones referring to the author keywords and the external authors—have been removed.

Table 2. Statistics of the dataset produced by the Dataset Builder.

Dataset			
Number of nodes	Number of classes/labels	Number of relations	Total number of edges
47,996	4	4	170,593
Numbers of training, evaluation, and testing samples			
Training edges	Evaluation edges		Testing edges
153,531	8528		8534

5.2.2. The Model Trainer

The main goal of the Model Trainer is to compute the node and edge embeddings of the PKG in a link prediction task. The Model Trainer is implemented by using a Graph Auto-Encoder, in which the encoder component is a GNN known as a Relational Graph Convolutional Network (R-GCN) [10] and the decoder is a factorization method known as DistMult [8].

In the first step, the R-GCN component takes as input two different edge lists. The first list includes positive examples that correspond to existing edges in the PKG, which are considered true facts. Negative examples are created under a closed-world assumption (the closed-world assumption (CWA) in the knowledge graph context means that lack of knowledge does imply falsity; in other words, missing facts in the KG are automatically considered false; the opposite perspective is the open-world assumption (OWA), according to which missing knowledge does not automatically imply falsity), according to which non-existing edges in the PKG represent false facts.

During the training process, the R-GCN exploits the information related to these edges and encodes the embeddings of feature-less nodes (defined using one-hot vectors), incorporating their local graph structure.

Then, the DistMult reconstructs the graph structure in the embedding space on the link prediction task and assigns a score to the reconstructed edges. The computed score represents the probabilities according to which the model is able to learn the graph structure in the embedding space. A sigmoid function is applied to the output of the DistMult decoder in order to formalize the link prediction task as a binary classification problem.

The loss function used for training the GAE is the binary cross-entropy loss with mean reduction, which is commonly used for classification tasks with just two classes:

$$\mathcal{L}(x, y, f(x)) = -\frac{1}{E}\sum_{i=1}^{E}(y_i \cdot log(f(x_i)) + (1 - y_i) \cdot log(1 - f(x_i))) \tag{1}$$

where E is the number of edges for the current batch, taking into account both positive and negative samples, and y_i represents their associated labels.

5.2.3. The Link Evaluator

The Link Evaluator is used for the prediction of facts that are not already present in the knowledge base and that are coherent with the defined network ontology, both in domain (for the subject) and range (for the object).

To create the set of new facts, the Link Evaluator firstly generates the set of all graph nodes, and then builds the set of possible facts by creating, for every node, all possible links to every other node in the graph. This is done by taking for each source node (subject) all possible permutations of relations (predicates) and destination nodes (objects). However, the majority of these automatically generated triples will be semantically incorrect. For this reason, the triples that are not correct with respect to the network ontology are discarded. Then, the remaining edges are scored by applying the DistMult factorization. The scored triples are grouped by subject and relation and sorted by their scores in descending order, discarding the ones which that scored lower than the threshold. The resulting new facts are then exported as a JSON file that can be used by the Graph Builder to create the corresponding RDF triples. These triples can then be added to the PKG, completing its knowledge base with new facts.

The predicted triples are used to obtain insights and recommendations, which are shown in the search results of the Geranium web application. For instance, among all of the predictions, there will be triples that link researchers to publications that have not been authored by them; these predicted authors can be interpreted as researchers that share the same research interests of the publication authors. By leveraging such predictions, the recommendation system can suggest unexplored research topics to the researchers, scientific journals that have published papers related to their fields of research, or the profiles of other researchers who share the same research interests but with whom they have never worked before.

5.3. KG Publication

The third phase of the provisioning, which is called KG publication, is made through an SPARQL endpoint, several REST APIs, and an interactive web interface empowered by the APIs.

5.3.1. REST API

The REST API, which is publicly available without any authentication or authorization required, only implements GET endpoints that allow one to request a batch (through offset-limit pagination) of results from the triplestore. The following is a list of the available endpoints with their respective query parameters:

- Get a set of publications: */publications?topic=${topic}&lines=${lines}&offset=${offset}*
- Get a publication from its URI: */publication?uri=${publicationUri}*
- Get a set of authors: */authors?topic=${topic}&lines=${lines}&offset=${offset}*
- Get an author from its URI: */author?uri=$authorUri&topic=${topic}&lines=${lines}&offset=${offset}*
- Get a set of topics: */topics?lines=${lines}&offset=${offset}*
- Get the abstract of a topic: */abstract?topic=${topic}*

5.3.2. Triplestore and SPARQL Endpoints

The triplestore is used to host an online version of the Polito Knowledge Graph and to expose its data through the SPARQL endpoint, which is publicly available like the REST API. The SPARQL endpoint is queried by the REST API to get the data requested by the frontend clients, which are returned as a list of JSON objects. The SPARQL queries are pre-built and implemented in a library made available to the REST API, which is only responsible for formatting the queries with the data sent in the API request.

5.3.3. Web Frontend

The Geranium platform is accessible by the users through a web application. On the homepage, a search bar allows the user to search for a given topic (with auto-completion) based on the topics present inside the graph. Once the user performs a search, the publications linked to the searched topic are presented to the user inside a UI card element. On the top of the page, some statistics about the searched topic are shown, such as the number of publications linked to the searched topic, together with a brief description taken from DBpedia. The user can then read the details about a publication, such as the authors, the recommended publications (obtained through the recommendation system, see Section 6.2), and the abstract, by clicking or tapping on one of the cards. Most of the elements in the cards' details are links that, if clicked, will trigger a new search.

6. Advanced Services and Tools

We implemented two advanced services on top of the PKG: a topic-based search engine and a link-prediction-based recommendation system.

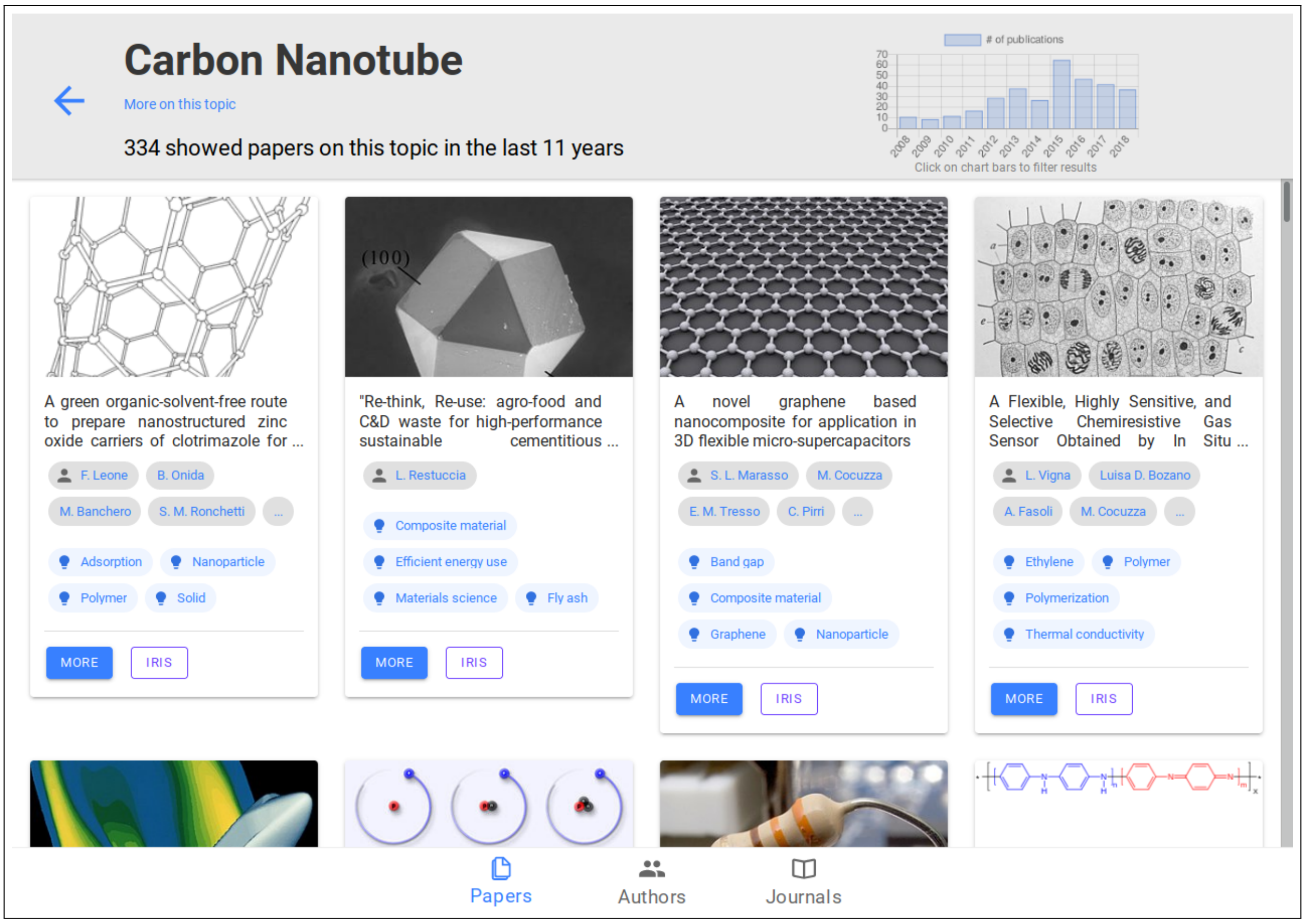
6.1. Topic-Based Search

One of the biggest limitations of the current state of IRIS is the lack of the possibility to search for all publications regarding a specific topic. In this context, the Geranium platform solves the problem by leveraging semantic tools in order to link each publication to its research topics, enabling the possibility of querying the PKG for the set of all of the publications given a specific input topic. This advanced functionality is made transparent to the user through the Geranium Web Application, which represents the entry point for the user and mimics the behavior of a modern search engine. A typical user, such as a researcher in the Politecnico di Torino, can search for a topic of interest and obtain as a result the list of all of the publications, authors, and journals that are linked to such a topic.

A screenshot of the results obtained by searching for the publications on *Carbon Nanotube* is available in Figure 4.

As can be seen, 334 publications are returned when searching for the "Carbon Nanotube" topic when using the Geranium semantic search engine, where only two (data taken from Polito IRIS instance: https://iris.polito.it/simple-search?query=&rpp=10&sort_by=score&order=desc&filter_field_1=title&filter_type_1=contains&filter_value_1=&filter_field_2=allkeywords&filter_type_2=equals&filter_value_2=Carbon+nanotube (accessed on 7

August 2021)) publications are returned when performing the same search on IRIS. Thus, the Geranium platform allows for the retrieval of all of the publications regarding a specific topic and, moreover, of all authors that have published the results of their studies in a given research field, allowing for a faster retrieval of expertise profiles. The latter is one of the mokst highly requested features by the Polito technological transfer office. The publication page presents an overview of the publication information, including the title, the authors, and the other research topics that have been linked to the publication itself. Moreover, in the upper-right area, there is a bar graph that can be exploited to filter the publications by year. We would like to implement advanced filtering tools in future releases.



Carbon Nanotube

More on this topic

334 showed papers on this topic in the last 11 years

A green organic-solvent-free route to prepare nanostructured zinc oxide carriers of clotrimazole for ...
F. Leone B. Onida M. Banchero S. M. Ronchetti ...
Adsorption Nanoparticle Polymer Solid

"Re-think, Re-use: agro-food and C&D waste for high-performance sustainable cementitious ...
L. Restuccia
Composite material Efficient energy use Materials science Fly ash

A novel graphene based nanocomposite for application in 3D flexible micro-supercapacitors
S. L. Marasso M. Cocuzza E. M. Tresso C. Pirri ...
Band gap Composite material Graphene Nanoparticle

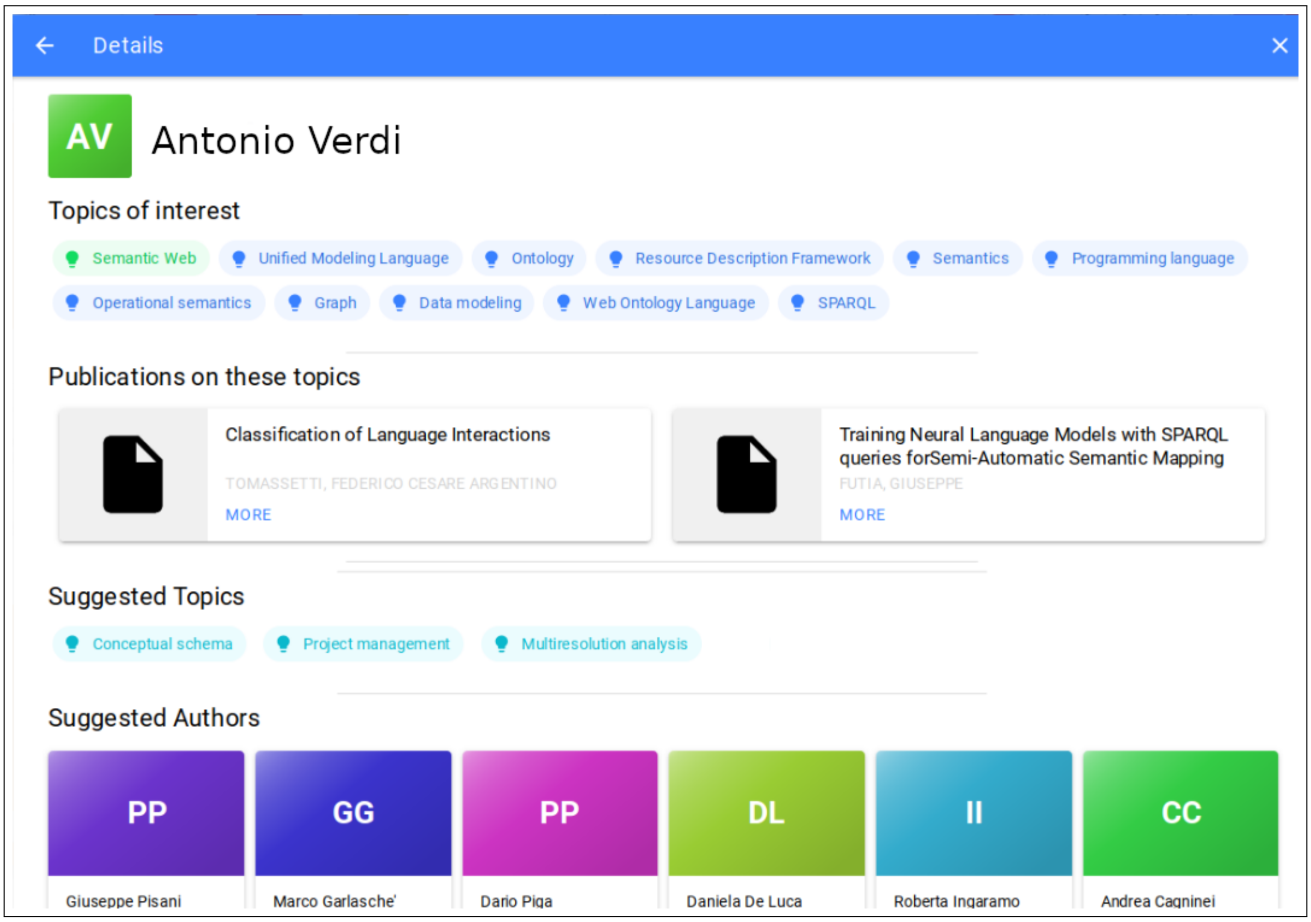
A Flexible, Highly Sensitive, and Selective Chemiresistive Gas Sensor Obtained by In Situ ...
L. Vigna Luisa D. Bozano A. Fasoli M. Cocuzza ...
Ethylene Polymer Polymerization Thermal conductivity

Figure 4. Screenshot of the Geranium Web App: results of the topic-based search.

6.2. Link-Prediction-Based Recommendations

Predicted facts can be used to obtain insights and recommendations that can be shown by the Viewer Module in the search results. By leveraging such predictions, the system can suggest different types of recommendations: (i) unexplored research topics for the researchers; (ii) scientific journals that have published papers related to their fields of research; (iii) the profiles of other researchers covering the same topics but come from different departments or disciplines.

The predictions of the GNN model are used to add valuable recommendations to the results visualized by means of the PWA. Considering, for instance, the researcher page, the system suggests new topics that are not directly covered by the author or other researchers with similar profiles. A screenshot of the topics and researchers suggested on the author page is available in Figure 5.


Details

AV Antonio Verdi

Topics of interest

Semantic Web · Unified Modeling Language · Ontology · Resource Description Framework · Semantics · Programming language · Operational semantics · Graph · Data modeling · Web Ontology Language · SPARQL

Publications on these topics

Classification of Language Interactions
TOMASSETTI, FEDERICO CESARE ARGENTINO
MORE

Training Neural Language Models with SPARQL queries forSemi-Automatic Semantic Mapping
FUTIA, GIUSEPPE
MORE

Suggested Topics

Conceptual schema · Project management · Multiresolution analysis

Suggested Authors

PP Giuseppe Pisani
GG Marco Garlasche'
PP Dario Piga
DL Daniela De Luca
II Roberta Ingaramo
CC Andrea Cagninei

Figure 5. Screenshot of the Geranium Web App: results of the link-prediction-based recommendation system.

7. Validation and Results

The main goal of our validation process is to evaluate the link prediction used for the recommendation service by employing an entity-ranking approach, which is a well-known technique in the context of knowledge base completion. This step allows the measurement of the model's capability of identifying the triples that correctly belong to the KG with respect to a set of synthetic negative examples.

Our validation approach compares the prediction scores achieved by the facts available in the PKG (true facts) and the synthetic facts that are not available in the PKG (false facts). These score values are computed by employing the DistMult factorization method. These negative examples are based on the generation of two sets of perturbed facts for each triple inside the PKG. The first set includes facts in which the source nodes have been perturbed, while the second set includes facts where the destination nodes have been perturbed. Source and destination nodes are perturbed using all possible entity nodes that are included in the initial version of the PKG (see Section 5.1).

More formally, starting from the *i*-th evaluation fact or edge (s_i, p_i, o_i), the new sets of synthetic edges include edges such as (a, p_i, o_i) and (s_i, p_i, b), with $a, b \in N$. An ideal model should be able to rank all of the triples in the evaluation set in the first position. However, this scenario never happens in real cases; for these reasons, we will exploit metrics that are well known in the literature in order to evaluate the capability of the model of setting a true fact in the highest position.

In our analysis, we used the following metrics:

- The *Reciprocal Rank* (RR): This is computed for a single evaluated edge (s_i, p_i, o_i) as $\frac{1}{rank_{(s_i,r_i,o_i)}}$, where the *rank* is obtained as explained above.

- The *Mean Reciprocal Rank* (MRR): This is the average of the RRs of all of the evaluation edges. The value obtained is representative of the accuracy of the entire model.
- The *Hits-at-N* (HITS@N): This is the number of triples for which the computed rank is between 0 and N.

To clarify these metrics, we consider the following example. A triple ranked in the tenth position of the sorted list of scores obtains an RR value of 0.1. If the facts in the evaluation set are 100, with half of them ranked in the first position and the other half in the tenth, the MRR obtained is $\frac{1}{100}\sum_{i=1}^{100}\frac{1}{rank_{(s_i,r_i,o_i)}} = 0.55$, while the HITS@1 is equal to 50, and the HITS@10 is equal to 100.

7.1. Validation Setup and Limitations

The training and the evaluation were performed on a workstation running Ubuntu 20.04 LTS and characterized by the following hardware:

- Intel CPU, 4C/8T architecture, 32GB of RAM.
- Nvidia GPU, 2304 cores, 8GB of VRAM.

A graph sampling step was adopted during the training process: 20,000 edges were randomly selected at each epoch and, for each sampled edge, we created ten synthetic negative examples. As a result, we used 220,000 edges per epoch for the training step, which was performed on the GPU. We could not extend the sample size due to the limitations of the available VRAM. In the future, we plan to adopt more powerful hardware in order to increase the number of edges per epoch and investigate the impacts of larger samples on the link prediction task. Compared to the execution on the CPU, the training time on the GPU was reduced by two orders of magnitude: from approximately 30 s per epoch for the forward and the backward pass to less than 0.3 s. The biggest bottleneck during training was still the graph sampling phase, which was implemented as single-threaded code using NumPy and required approximately five seconds to perform with the hardware at our disposal, which had limited single-threaded performance. The implementation of more efficient sampling methods is also left for future releases.

The model evaluation was performed on the CPU due to memory constraints. As a consequence of the dimensionality of the evaluation graph, which included the perturbed edges, the matrices used to perform the trained model evaluation could not be stored in the 8 GB of VRAM available in the GPU. Moreover, the available memory on the CPU was not sufficient to perform the evaluation task in parallel for all of the edges, thus requiring the evaluation to be performed in batches. The system memory available in the workstation allowed us to perform the evaluation in batches of 80 triples. Considering the above limitations, the model evaluation proved to be the slowest task of the whole pipeline, taking approximately twenty minutes for each run.

7.2. Hyperparameter Selection

Table 3 shows the results in terms of MRR results when testing different hyperparameter values. In particular, our experiment tested the model's behavior with different combinations of the learning rate and the regularization parameters; the highest result was achieved by setting 0.001 and 0.5 as the learning rate and regularization value, respectively. This regularization value was much higher than the benchmark reported by the R-GCN authors [10], which corresponded to 0.01. The motivation was that a very low number of relationships characterized the PKG compared to the datasets available in the literature. This peculiarity implied a low number of convolutional filters, which determined a limited variety when encoding the local graph structure of the KG nodes. In this context, a higher regularization value helps the model to better differentiate the embedding values, even in the case of a low number of relations, and to avoid the overfitting of common features. In Table 3, the values in the round brackets are the results of the validation test when using only the decoder component (DistMult) and removing the encoder. These results demonstrate that the GNNs play a fundamental role in improving the accuracy of traditional link predictions.

Table 3. Evaluation of different combinations of the learning rate and regularization parameters, using the MRR value as a benchmark. The first table shows the MRR of the best model during the training phase. The second table shows the MRR obtained by the best model on the test set.

(a) MRR of the Best Model During Training

		Learning Rate				
	MRR	**0.05**	**0.01**	**0.005**	**0.001**	**0.0005**
Regularization	**1.0**		0.0623	0.0504	0.0823	0.0793
	0.5		0.0677	0.0718	0.0924	0.0865
	0.1	0.0004	0.0703	0.852	0.0764	0.0706
	0.05		0.0707	0.0763	0.0673	
	0.01		0.062	0.0725		

(b) MRR Obtained When Evaluating the Best Model on the Test Set

		Learning Rate				
	MRR	**0.05**	**0.01**	**0.005**	**0.001**	**0.0005**
Regularization	**1.0**		0.0611	0.0497	0.081	0.079
	0.5		0.0662	0.0723	0.0882(0.008)	0.0818(0.006)
	0.1	0.0002	0.0693	0.0827(0.001)	0.0726	0.0666
	0.05		0.0687	0.0726	0.0654	
	0.01		0.0589	0.07		

The best MRR value obtained in our experiment (0.0924) showed that the correct fact, on average, was ranked at the eleventh position. The MRR benchmark value achieved by the authors of R-GCN was 0.158 (sixth position in the ranking) on the FB15K dataset (14,541 entities and 237 relation types), which is commonly adopted to evaluate link prediction tasks. Compared to the benchmark value, our best result is noteworthy because the model is able to rank (on average) the true triple in the eleventh position when ranking 47,995 corrupted triples, while in the case of the FB15k-237 dataset, only 14,951 corrupted triplets are generated for each true fact. Figure 6 shows how the testing triples are distributed over the possible ranking values. In this case, the model is able to correctly assign a high rank to most of the true facts.

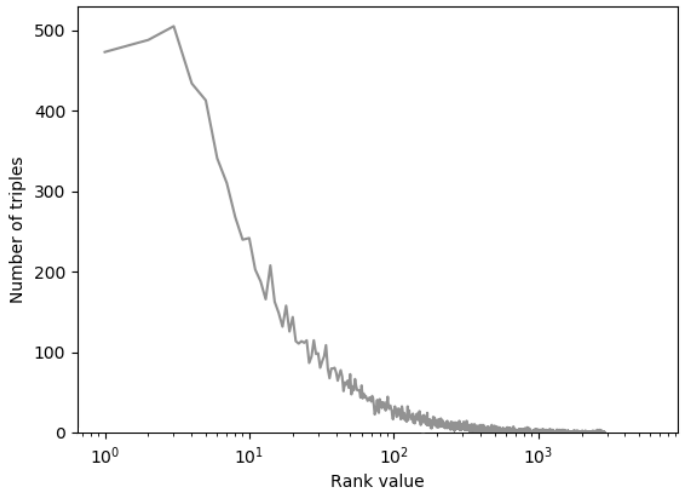


Figure 6. Number of triples for each ranking position obtained by evaluating the best model on the test set. The ranking positions range from 1 to 47,995.

7.3. Topics' Impact on the Prediction Task

Most of the entities included in the PKG are represented by research topics, which correspond to the *TMFResource* instances connected to the publications by means of the *dc:subject* property. Table 1 shows that almost a third of the PKG edges connected a publication to a topic extracted by TMF. Considering the potential impact of the number of these entities on the link prediction mechanism, a second experiment was conducted to evaluate the impact of a diverse number of research topics within the PKG. Three different versions of the PKG were created—versions with three, seven, and fourteen topics. The experiment employed the learning rate and regularization hyperparameters that obtained the best results in the previous experiment.

Table 4 shows that the best results were obtained with the PKG that included only three topics. According to this result, decreasing the number of topics seemed to improve the evaluation result. However, this did not directly lead to a better link prediction model. In fact, the better ranking was a consequence of the less complex graph structure. This version of the PKG included almost half of the topic entities, and as a consequence, it was an easy task for the model to rank the correct facts in a higher position. Moreover, due to the low number of topics and the features of TMF, there was a high number of publications that shared the same topics. Therefore, it was not difficult for the model to predict facts that included such topics during the evaluation stage on the test set. On the other hand, increasing the number of topics within the PKG led to a degradation of the MRR values. This behavior was a consequence of the more complex graph structure. Moreover, increasing the number of facts (and edges) that included the predicate *dc:subject* had an impact on the neighborhood representation encoded by the GNN component. This bias in the neighborhood structure resulted in closer values in the embedding representations of the nodes and could lead to more inaccurate predictions during the evaluation phase of the link prediction.

Table 4. Impact of the number of topics present in the RDF graph on the accuracy of the trained model.

Number of topics extracted per abstract	**3**	**7**	**14**
Number of *TMFResource* entities in the graph	9.591	16.988	26.541
Number of *dc:subject* edges	45.423	107.093	212.226
MRR on test data	0.103	0.0882	0.0671
HITS@15	31.5%	27.9%	20.3%

8. Conclusions and Future Work

This article presented Geranium, a semantic platform for constructing and publishing the Polito Knowledge Graph (PKG), a novel KG in the scholarly domain. In this work, we extracted the metadata from the IRIS database, the system adopted by the Politecnico di Torino to release its publications, and we mapped the extracted information to a reference ontology. The current implementation of Geranium provides a web user interface and a collection of APIs that allow users with different technical backgrounds to explore all of the platform's features. We made the source code and datasets publicly available (https://github.com/geranium-project, accessed on 26 August 2021).

Our work showed that adopting semantic technologies is essential in order to create advanced services, including topic-based search and recommendation functionalities. In regards to the first feature, we automatically categorized the Polito publications using TellMeFirst (TMF) and DBpedia entities, thus addressing the ambiguity issues related to the traditional keyword-based search engines. For the recommendation task, we provided evidence of the promising role of the combination of KGs and Graph Neural Networks (GNNs) in the scholarly domain. The current literature still explores this combination in a prototypal context using synthetic data. However, our investigation showed that such an approach works well in a real application scenario in order to uncover interesting collaboration opportunities between researchers. Our work validated, in particular, the link prediction mechanism adopted for the recommendation task. We performed different experiments to detect the hyperparameters that were suitable for our scenario and to understand the role of semantic topics in improving the prediction accuracy.

For future work, we plan to extend different aspects of the Geranium platform in terms of data sources, functionalities, and novel approaches and algorithms. We will update the current version of the PKG by integrating new information from external repositories, including data from other academic institutions and domain data from the Linked Open Data (LOD) community. To improve the topic-based search system, we intend to overcome the lack of hierarchical dependence between the topics used to enrich the initial version of the PKG by also exploiting the semantic relations between the entities available in DBpedia. In regards to the recommendation system, the current implementation is not able to predict new facts regarding entities that were not seen during the training phase. For this reason, we plan to overcome these limitations by extending the Geranium architecture with other approaches available in the literature. Starting from the results achieved with the current version of the recommender, we will extend the evaluation part by submitting a questionnaire to potential users of Geranium, including researchers, policymakers, and citizens. We want to investigate the impact of Geranium as a tool for supporting universities by sharing their intellectual outputs, especially publications, among all members of society.

Author Contributions: Conceptualization, G.G., G.F. and A.V.; Data curation, G.G. and G.F.; Investigation, G.G. and G.F.; Methodology, G.G.; Project administration, G.F.; Software, G.G.; Supervision, G.G. and G.F.; Validation, G.G., G.F. and A.V.; Visualization, G.G.; Writing—original draft, G.G. and G.F.; Writing—review and editing, G.G., G.F., A.V. and J.C.D.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are openly available at: https://github.com/geranium-project (accessed on 3 September 2021).

Acknowledgments: The authors would like to thank Cosma Alex Vergari and Mehdi Khrichfa, interns at the Nexa Center for Internet and Society, for their contributions in the development of the Geranium platform.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Turki, H.; Hadj Taieb, M.; Shafee, T.; Lubiana, T.; Jemielniak, D.; Ben Aouicha, M. Representing COVID-19 Information in Collaborative Knowledge Graphs: the Case of Wikidata. *Semantic Web J.* **2021**. [CrossRef]
2. Kejriwal, M. Knowledge Graphs and COVID-19: Opportunities, Challenges, and Implementation. *Harv. Data Sci. Rev.* **2020**. [CrossRef]
3. Chatterjee, A.; Nardi, C.; Oberije, C.; Lambin, P. Knowledge Graphs for COVID-19: An Exploratory Review of the Current Landscape. *J. Pers. Med.* **2021**, *11*, 300. [CrossRef] [PubMed]
4. Cifariello, P.; Ferragina, P.; Ponza, M. Wiser: A semantic approach for expert finding in academia based on entity linking. *Inf. Syst.* **2019**, *82*, 1–16. [CrossRef]
5. Chen, X.; Jia, S.; Xiang, Y. A review: Knowledge reasoning over knowledge graph. *Expert Syst. Appl.* **2020**, *141*, 112948. [CrossRef]
6. Dwivedi, V.P.; Joshi, C.K.; Laurent, T.; Bengio, Y.; Bresson, X. Benchmarking graph neural networks. *arXiv* **2020**, arXiv:2003.00982.
7. Lassila, O. Web metadata: A matter of semantics. *IEEE Internet Comput.* **1998**, *2*, 30–37. [CrossRef]
8. Yang, B.; Yih, W.t.; He, X.; Gao, J.; Deng, L. Embedding entities and relations for learning and inference in knowledge bases. *arXiv* **2014**, arXiv:1412.6575.
9. Zhang, S.; Tong, H.; Xu, J.; Maciejewski, R. Graph convolutional networks: a comprehensive review. *Comput. Soc. Netw.* **2019**, *6*, 1–23. [CrossRef]
10. Schlichtkrull, M.; Kipf, T.N.; Bloem, P.; Van Den Berg, R.; Titov, I.; Welling, M. Modeling relational data with graph convolutional networks. In Proceedings of the European Semantic Web Conference, Heraklion, Greece, 3–7 June 2018; Springer: Cham, Switzerland, 2018; pp. 593–607.
11. 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 Gener. Comput. Syst.* **2021**, *116*, 253–264. [CrossRef]
12. Ronzano, F.; Saggion, H. Dr. inventor framework: Extracting structured information from scientific publications. In Proceedings of the International Conference on Discovery Science, Banff, AB, Canada, 4–6 October 2015; Springer: Cham, Switzerland, 2015; pp. 209–220.
13. Luan, Y.; He, L.; Ostendorf, M.; Hajishirzi, H. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. *arXiv* **2018**, arXiv:1808.09602.
14. Salatino, A.A.; Osborne, F.; Thanapalasingam, T.; Motta, E. The CSO classifier: Ontology-driven detection of research topics in scholarly articles. In Proceedings of the International Conference on Theory and Practice of Digital Libraries, Oslo, Norway, 9–12 September 2019; Springer: Cham, Switzerland, 2019; pp. 296–311.
15. Angeli, G.; Premkumar, M.J.J.; Manning, C.D. 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)*; Association for Computational Linguistics: Beijing, China, 2015; pp. 344–354.
16. Labropoulou, P.; Galanis, D.; Lempesis, A.; Greenwood, M.; Knoth, P.; Eckart de Castilho, R.; Sachtouris, S.; Georgantopoulos, B.; Anastasiou, L.; Martziou, S.; et al. OpenMinTeD: A platform facilitating text mining of scholarly content. In Proceedings of the 7th International Workshop on Mining Scientific Publications (WOSP 2018) at LREC 2018, Miyazaki, Japan, 7–12 May 2018.
17. Rocha, O.R.; Vagliano, I.; Figueroa, C.; Cairo, F.; Futia, G.; Licciardi, C.A.; Marengo, M.; Morando, F. Semantic annotation and classification in practice. *IT Prof.* **2015**, *17*, 33–39. [CrossRef]
18. Chicaiza, J.; Valdiviezo-Diaz, P. A Comprehensive Survey of Knowledge Graph-Based Recommender Systems: Technologies, Development, and Contributions. *Information* **2021**, *12*, 232. [CrossRef]
19. Nayyeri, M.; Vahdati, S.; Zhou, X.; Yazdi, H.S.; Lehmann, J. Embedding-based recommendations on scholarly knowledge graphs. In Proceedings of the European Semantic Web Conference, Heraklion, Greece, 31 May–4 June 2020; Springer: Cham, Switzerland, 2020; pp. 255–270.
20. Kanakaris, N.; Giarelis, N.; Siachos, I.; Karacapilidis, N. Shall I Work with Them? A Knowledge Graph-Based Approach for Predicting Future Research Collaborations. *Entropy* **2021**, *23*, 664. [CrossRef]
21. Vahdati, S.; Palma, G.; Nath, R.J.; Lange, C.; Auer, S.; Vidal, M.E. Unveiling scholarly communities over knowledge graphs. In Proceedings of the International Conference on Theory and Practice of Digital Libraries, Porto, Portugal, 10–13 Septembe 2018; Springer: Cham, Switzerland, 2018; pp. 103–115.
22. Liu, C.; Li, L.; Yao, X.; Tang, L. A survey of recommendation algorithms based on knowledge graph embedding. In Proceedings of the 2019 IEEE International Conference on Computer Science and Educational Informatization (CSEI), Kunming, China, 16–19 August 2019; pp. 168–171.
23. Ameen, A. Knowledge based recommendation system in semantic web-a survey. *Int. J. Comput. Appl* **2019**, *182*, 20–25. [CrossRef]
24. Graves, M.; Constabaris, A.; Brickley, D. Foaf: Connecting people on the semantic web. *Cat. Classif. Q.* **2007**, *43*, 191–202. [CrossRef]
25. Weibel, S.L.; Koch, T. The Dublin core metadata initiative. *D-Lib Mag.* **2000**, *6*, 1082–9873. [CrossRef]
26. Wang, M.; Yu, L.; Zheng, D.; Gan, Q.; Gai, Y.; Ye, Z.; Li, M.; Zhou, J.; Huang, Q.; Ma, C.; et al. Deep Graph Library: Towards Efficient and Scalable Deep Learning on Graphs. *arXiv* **2019**, arXiv:1909.01315v1.

27. Quintel, D.; Wilson, R. Analytics and Privacy. *Inf. Technol. Libr.* **2020**, *39*, 3. [CrossRef]
28. Zhang, Y.; Ling, G.; Yong-cheng, W. An improved TF-IDF approach for text classification. *J. Zhejiang Univ.-Sci. A* **2005**, *6*, 49–55.