

Article

# Semantic Publication of Agricultural Scientific Literature Using Property Graphs

Francisco Abad-Navarro <sup>1,2</sup>, José Antonio Bernabé-Díaz <sup>1,2</sup>, Alexander García-Castro <sup>2</sup> and Jesualdo Tomás Fernández-Breis <sup>1,\*</sup>

<sup>1</sup> Departamento de Informática y Sistemas, Universidad de Murcia, IMIB-Arrixaca, CP 30100 Murcia, Spain; francisco.abad@um.es (F.A.-N.); joseantonio.bernabe1@um.es (J.A.B.-D.)

<sup>2</sup> BASF SE, G-FDR/BI-G200, Carl-Bosch-Strasse 38, 67056 Ludwigshafen am Rhein, Germany; alexgarcia@gmail.com

\* Correspondence: jfernand@um.es; Tel.: +34-868884613

Received: 31 December 2019; Accepted: 20 January 2020; Published: 26 January 2020



**Abstract:** During the last decades, there have been significant changes in science that have provoked a big increase in the number of articles published every year. This increment implies a new difficulty for scientists, who have to do an extra effort for selecting literature relevant for their activity. In this work, we present a pipeline for the generation of scientific literature knowledge graphs in the agriculture domain. The pipeline combines Semantic Web and natural language processing technologies, which make data understandable by computer agents, empowering the development of final user applications for literature searches. This workflow consists of (1) RDF generation, including metadata and contents; (2) semantic annotation of the content; and (3) property graph population by adding domain knowledge from ontologies, in addition to the previously generated RDF data describing the articles. This pipeline was applied to a set of 127 agriculture articles, generating a knowledge graph implemented in Neo4j, publicly available on Docker. The potential of our model is illustrated through a series of queries and use cases, which not only include queries about authors or references but also deal with article similarity or clustering based on semantic annotation, which is facilitated by the inclusion of domain ontologies in the graph.

**Keywords:** knowledge graph; property graph; semantic web; digital publishing; literature search

## 1. Introduction

The importance of science in society has significantly increased in the last century. Science has become a big, global, complex, and collaborative effort due to the increase of the resources assigned to science world-wide [1,2]. This has implied an exponential increase of the scientific production, reaching a growth rate between 8 and 9% from the period since the Second World War [3]. According to Scimago Journal & Country Rank (<https://www.scimagojr.com/journalrank.php>), 2,761,624 articles were published in 2018, and 8,185,591 articles were published from 2015 to 2017. The agricultural and biological sciences constitute around 8% of such production, with 226,310 articles published in 2018 and 659,111 from 2015 to 2017. The availability of a larger base of scientific papers is mostly useful for scientists and science, but this is at the cost of making it difficult to find relevant related research. Since a human being is not usually able to review all these documents, this task should be delegated to software tools, which are able to quickly process large amounts of data. By improving the methods used for literature searches, a scientist could quickly find the most interesting articles, which will augment the efficiency of his/her research work.

Scientific publications are usually published in human-friendly formats such as PDF or HTML, whose content is hardly understood by machines. Therefore, the first requisite to use automatic

processes in order to find relevant articles is that these articles are represented in machine readable formats. The Semantic Web [4] proposes different technologies which can contribute to this goal, namely, OWL ontologies [5], for representing domain knowledge; RDF [6], for describing resources; and SPARQL [7] as a query language. These technologies allow representing domain knowledge by using formal languages, which can be understood by machines, enabling the development of knowledge-based automated tasks. Semantic Web technologies are being adopted by publishers due to those benefits. David Shotton defines semantic publishing as *anything that enhances the meaning of a published journal article, facilitates its automated discovery, enables its linking to semantically related articles, provides access to data within the article in actionable form, or facilitates integration of data between papers* [8]. This involves enriching articles with machine-readable metadata, allowing the verification of published information and providing the capacity for automated discovery and for summarizing. Thus, there are several key problems in semantic publishing: (1) what metadata should be included, (2) how this metadata is modelled, and (3) how to store and access to this metadata.

Our previous studies [9,10] addressed problems 1 and 2 by proposing an RDF model for scientific literature description that reuses terms from well known domain ontologies and vocabularies, such as *Bibliographic Ontology* (bibo) (<http://bibliontology.com/>), *Dublin Core Terms* (dcterms) [11] or *Friend Of A Friend Ontology* (foaf) ([www.foaf-project.org/](http://www.foaf-project.org/)), among others, following the Biotea model. This model covers most publishing information, including authors, references, journals, publishers, the structure and the content of the articles, and semantic annotations that enrich these contents. Moreover, we provided tools for RDF generation from literature represented in Journal Article Tag Suite (JATS) format [12].

In this paper, we address problem 3 by proposing the use of graph databases for storing and querying RDF data generated according to the Biotea model. Thus, in this work, we provide a workflow to populate property graphs including not only scientific literature data, but also semantic annotations and domain knowledge, which facilitate better understanding of the data by automatic agents. We exemplify the advantages of our model by means of different use cases, including article similarity, literature search, or literature metrics, among others.

The remaining structure of the paper is described next. The Related Work section describes how the three key problems of semantic publishing have been approached by the scientific community. In the Materials and Methods section, we describe a pipeline which uses Biotea for generating an RDF knowledge graph of scientific literature, together with Neosemantics (<https://github.com/Neo4j-labs/neosemantics>) for translating RDF into property graph model implemented in Neo4j. The Results section illustrates the knowledge graph generated and the use of this pipeline on a set of 127 articles in the agriculture domain. The comparison with state of the art solutions and future work is considered in the Discussion. Finally, some conclusions are put forward.

## 2. Related Work

State-of-the-art solutions propose different methods to address the three key problems of semantic publishing: (1) what metadata should be included, (2) how this metadata is modelled, and (3) how to store and access to this metadata. For instance, SciGraph (<https://scigraph.springernature.com>), created by Springer–Nature, address problem 1 by providing the most common metadata for scientific literature, including authors, affiliations, references or publishers, among others. The abstract of the articles is the only textual content taken into account. In connection with problem 2, this project uses an RDF model based in known vocabularies, such as dcterms, foaf or schema.org [13], in addition to own ontologies. Although semantic enrichment is not performed, each article is classified according Field of research (FoR) codes [14]. Regarding problem 3, the knowledge graph itself is not directly accessible by using any query language. Nonetheless, raw data is provided through JSON-LD [15] files. Also, there is a REST API available, which provides functionality for a literature search, supporting metadata-based filters, including DOI, FoR code and year of publication or journal, among others.

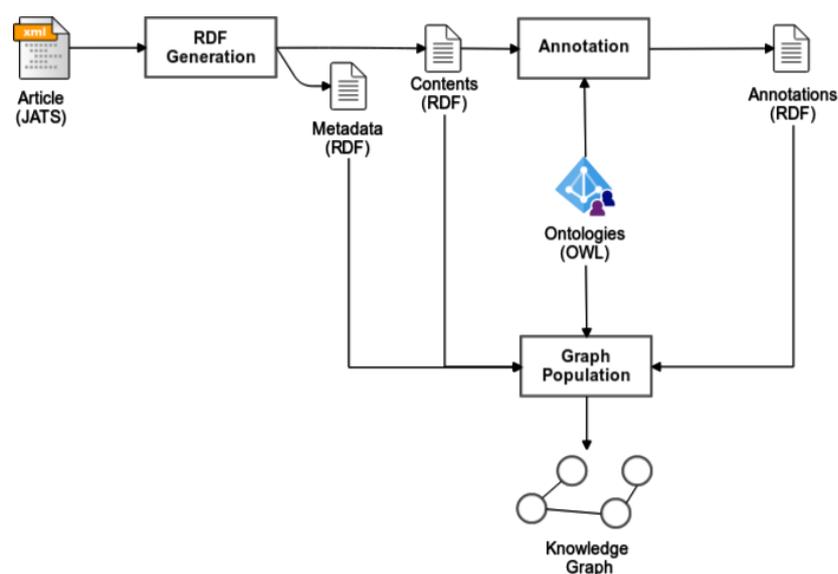
Other related work is *The Semantic Lancet project* [16]. This work proposes a model compliant with the *Semantic Publishing And Referencing Ontologies* (SPAR ontologies) [17], which include common bibliographic metadata. In this case, the content of the articles is taken into account for semantic enriching purposes, particularly, processing the abstracts by means of FRED tool [18] to obtain an RDF representation that captures the meaning of the free text. This project proposes a triplestore in order to publish the knowledge graph, making it accessible through an SPARQL endpoint. Moreover, they provide several applications built on the top of the knowledge graph, which offer an easy to use user interface.

Another recent project is the *Open Research Knowledge Graph* (ORKG) [19], whose model represents not only bibliographic metadata but also semantic descriptions of scholarship knowledge. The semantic enrichment in this model consists of describing each article by means of four generic concepts that summarize its contribution: process, method, material and data. Then, concrete annotations found in the text are linked to these generic concepts. Regarding how to store the data, this project makes use of a labelled property graph, a triplestore and a relational database, where each database engine is used for specific purposes in order to provide different services.

OpenBiodiv [20] is another knowledge graph focused on biodiversity in the scientific literature. It is defined by the OpenBiodiv-O ontology [21], which integrates bibliographic and biodiversity domain ontologies, such as SPAR ontologies or Treatment Ontology (<https://github.com/plazi/TreatmentOntologies>). This model is defined in RDF, and it includes information about article metadata, authors, institutions, geographical locations or taxon names, among others. Although the content of the articles is also included in this model, it is not processed by any semantic enriching process. This knowledge graph is implemented in a triplestore, and accessible through an SPARQL endpoint.

### 3. Materials and Methods

This section describes the pipeline developed for generating a knowledge graph from a set of articles (see Figure 1). In summary, this process consists of three main steps, namely, *RDF generation*, in which the files with the content of the scientific publications are translated into RDF, including both article metadata and contents; *Annotation*, where the content of the article is annotated with ontology concepts, generating new RDF triples describing these annotations, and *Knowledge graph population*, in which generated RDF files, together with the ontologies used for annotation, are populated into a knowledge graph.



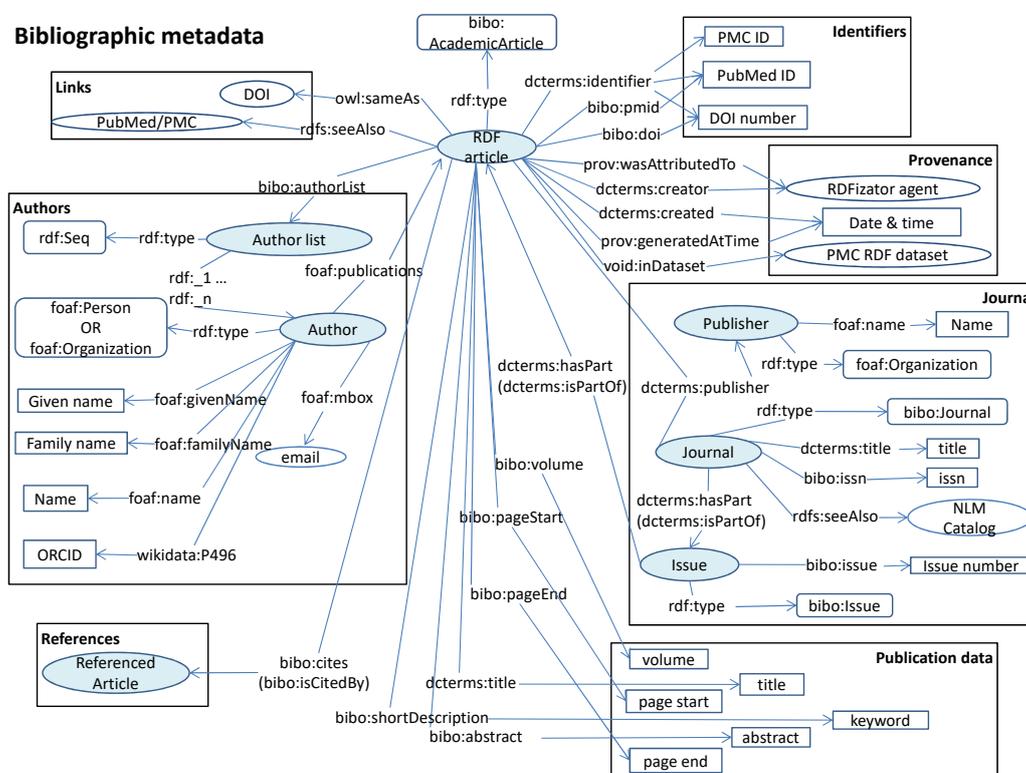
**Figure 1.** General overview of the pipeline used for generating a knowledge graph from the scientific publications.

### 3.1. RDF Generation

This step converts scientific publications into RDF by using well known bibliographic domain ontologies and vocabularies. Our current method takes as input scientific publications in JATS format, which is an XML file that contains all the information about an article, including metadata (authors, references, identifiers or journal data) and contents. This translation is performed by applying the Biotea RDFization process (<https://github.com/biotea/biotea-rdfization>). Two RDF files are generated for each JATS input file, namely, metadata and sections, which include the bibliographic information, and the structure and contents, respectively.

The RDF model used for describing the metadata resulting from the conversion is depicted in Figure 2. In this model, the `rdf:seeAlso` property is used to link articles and journals to their web page in PubMed and the National Library of Medicine (NLM) catalog (<https://www.ncbi.nlm.nih.gov/nlmcatalog/>), respectively. Besides, this model reuses properties and terms from:

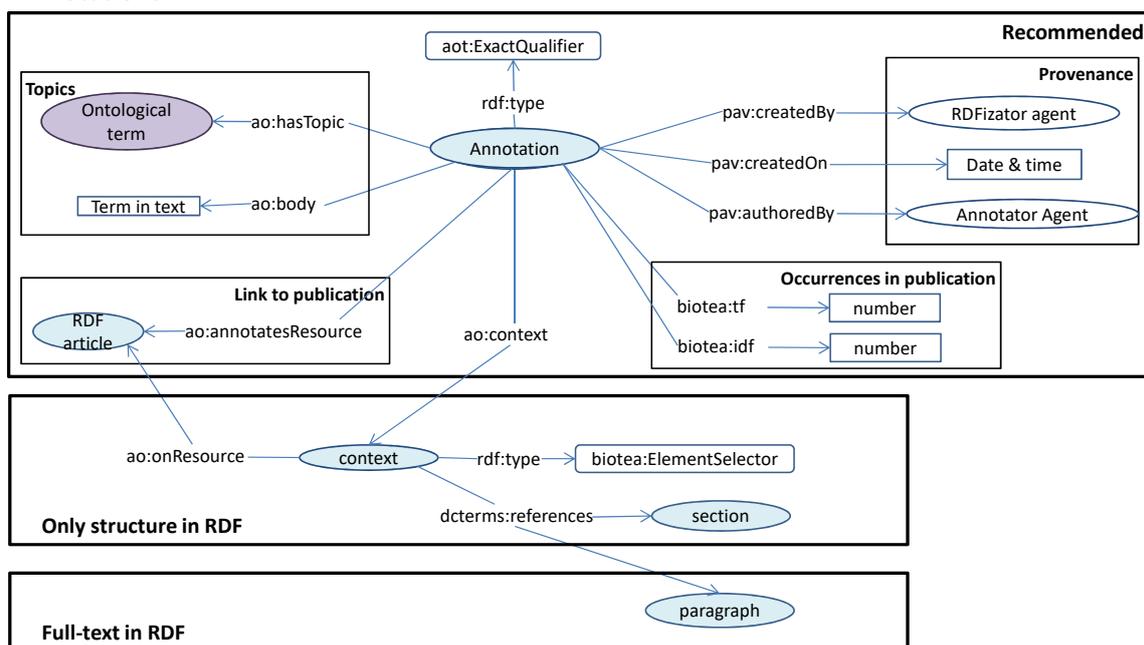
- *Bibliographic Ontology* (bibo) to represent bibliographic data such as author lists, literature identifiers, cites, references or abstracts;
- *Dublin Core Terms* (dcterms) [11], to describe predicates like `hasPart`, `title` or `publisher`;
- *Friend Of A Friend Ontology* (foaf), to represent information about authors;
- *Provenance Ontology* (prov) [22], using the predicate `generatedAtTime` in order to save the date in which the RDF was generated;
- *Wikidata* [23], to store the authors ORCID's, which was retrieved from ORCID search web service endpoint (<https://pub.orcid.org/v2.0/search>) from article DOI, author given name, and author family name, when possible;
- *Semanticscience Integrated Ontology* (sio) [24], to relate an article to a PMC RDF dataset through the predicate `is_data_item_in`.



**Figure 2.** Model used for describing article metadata, including authors, references, identifiers and journal information. Core concepts in the model are marked in blue. Adapted from [10].



## Annotations



**Figure 4.** Model used for describing annotations according to the Annotation Ontology. Core concepts in the model are marked in blue. The ontology terms marked in purple are provided by the annotator. Adapted from [10].

### 3.3. Knowledge Graph Population

This step generates a graph database by loading both the RDF files previously created, which include article metadata, sections and annotations, in addition to the ontologies used for the annotation, which add domain knowledge to the graph. This has been implemented using the Python script available at <https://github.com/biotea/biotea-graph-population> and the graph database used was Neo4j [28], whose model differs from RDF because it implements a property graph model.

RDF [6] is a graph model used for describing resources, where resources are identified by a URI and they represent entities. These resources are described by a set of triples of the form <subject, predicate, object>, where the subject is linked to the object through the predicate. The subject will always be a resource, and the object could be another resource or a literal value. Then, a list of triples can be seen as a graph where resources are nodes and properties are edges.

In [29], the property graph model is described as follows:

1. A property graph is made up of nodes, relationships and properties.
2. Nodes contain properties [...] in the form of arbitrary key-value pairs. The keys are strings and the values are arbitrary data types.
3. A relationship always has a direction, a label, and a start node and an end node.
4. Like nodes, relationships can also have properties.

In the Neo4j property graphs, nodes are also labelled with their categories in order to establish their types. Taking these differences into account, we need to apply a model transformation in order to store RDF data into a property graph database. This transformation was carried out by the Neosemantics plugin for Neo4j. This plugin is able to load RDF data into a Neo4j graph according two kind of conversions depending on the desired detail level, namely `importRDF` and `loadOntology`.

On the one hand, the `importRDF` function was used for uploading the metadata, sections and annotation RDF files. This function ingests all triples in an RDF dataset and uploads them into the graph database by applying the following rules [30]:

- Rule 1:** Every RDF resource becomes a property graph node. Since the subject in RDF is always a resource, there is at least one property graph node created or merged into an existing node.
- Rule 2:** If the object is a literal, the predicate and object become respectively a property name and value. The property is added to the created or existing node corresponding to the resource.
- Rule 3:** If the object is a resource, then both the subject and objects are transformed into a node. A relationship between them is created and it holds the predicate’s name as a relationship type.
- Rule 4:** If the predicate is `rdf:type`, the subject becomes a node having a label set to the objects name (which is actually the resource type).

Figure 5 shows the RDF representation of two resources and the property graph model resulting from applying the rules described before. In this case, `res:article1` and `res:article2` are the unique subjects of the triples in the RDF model. Thus, according to rule 1, they are mapped onto nodes in the property graph model, labelling them as `Resource`. Also, the URIs in the RDF model are stored as a property in the property graph node. Then, the RDF properties `dct:title` are translated to properties inside the nodes in the property graph model, as rule 2 states. Also, the RDF property `bibo:cites` is converted into a relation according to rule 3. Finally, RDF `rdf:type` statements are translated to category labels in the property graph model, following rule 4.

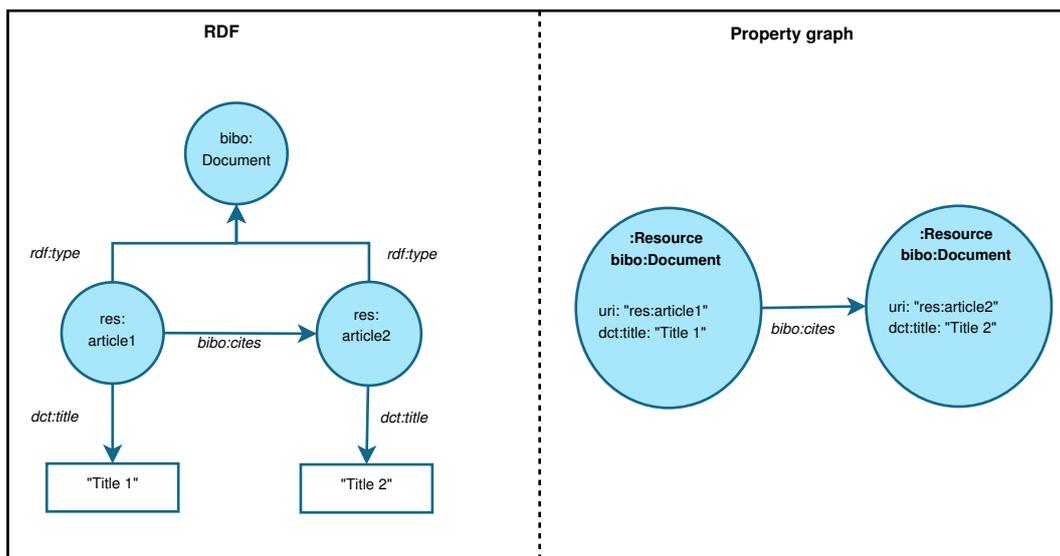


Figure 5. RDF and property graph equivalence.

On the other hand, the `importOntology` function was used for uploading the ontologies used by the annotator. Although ontologies can also be expressed in RDF, they usually contain complex axioms involving anonymous nodes that would add complexity to the graph model if they were uploaded through the `importRDF` function. The `importOntology` function only imports the following RDF statements:

1. Named class (category) declarations with both `rdfs:Class` and `owl:Class`. The nodes representing named classes are labelled as `Class`.
2. Explicit class hierarchies defined with `rdf:subClassOf` statements. The edges representing `rdf:subClassOf` relations are labelled as `subClassOf`.
3. Property definitions with `owl:ObjectProperty`, `owl:DatatypeProperty` and `rdfs:Property`, whose corresponding nodes have been labelled as `Relation`, `Property` and `Property`, respectively.
4. Explicit property hierarchies defined with `rdfs:subPropertyOf` statements. The edges representing this relation were labelled as `subPropertyOf`.

5. Domain and range information for properties described as `rdfs:domain` and `rdfs:range` statements, whose corresponding edges in the graph were labelled as `domain` and `range`.

#### 4. Results

In this section, we present our main results. First, we explain the experiment carried out. Then, we provide a description of the resulting knowledge graph as well as possible use cases for information retrieval. All the queries associated with the use cases can be found in Appendix B.

##### 4.1. Description of the Experiment

We have applied the pipeline described in Section 3 to a set of 127 articles randomly selected from PubMed Central Repository (<https://www.ncbi.nlm.nih.gov/pmc/>), thus creating a Neo4j-based knowledge graph about agriculture. The 127 JATS files were downloaded through the NCBI OA web service (<https://www.ncbi.nlm.nih.gov/pmc/tools/oa-service/>). Some of these articles belong to special agriculture collections, while others were found by searching agricultural related articles manually (see Appendix A for a complete list). We used the annotator provided by AgroPortal [31] for the annotation step. This annotator uses more than 100 ontologies covering domains like agricultural research, animal science, ecology, nutrition or farming, among others, including ontologies such as *Global Agricultural Concept Scheme* (gacs) [32], *Thesaurus for Animal Physiology and Livestock systems* (TriPhase) [33], *Environment Ontology* (envo) [34], *FoodOn* [35] or *AgroRDF* [36]. These ontologies were obtained through the AgroPortal REST API and, when possible, converted to RDF/XML format with the ROBOT tool [37] in order to insert them into the knowledge graph.

##### 4.2. The Knowledge Graph

The resulting Neo4j-based knowledge graph has been released as a Docker image, which is available at <https://hub.docker.com/r/fanavarro/neo4j-agro-dataset>. This knowledge graph contains information about 127 agriculture related articles, including metadata, data and annotations provided by the AgroPortal annotator, which uses the AgroPortal ontologies to provide a semantic context to the annotations. This graph is formed by 2,061,133 nodes and 3,176,199 edges in total (see Query 1).

###### 4.2.1. Basic Metrics of the Graph

The knowledge graph supports queries to extract relevant information about article metadata. For example, in connection with article authorship, the article with a greater number of authors is *Food systems for sustainable development: proposals for a profound four-part transformation*, with 27 authors. In contrast, the article *The economic value of mussel farming for uncertain nutrient removal in the Baltic Sea* has the lowest number of authors, with only one (see Query 2). The average number of authors per article in the data set is 6.98 (see Query 3).

Another metric that could be retrieved from the knowledge graph is the number of articles per author (see Query 4). In this case, In-Jung Lee is the most prolific author in the data set, with 4 articles written. The average number of publications per author is 1.04 (see Query 5). With respect to bibliographic references, *Plant Defense against Insect Herbivores* is the article with the largest number of references (394), while *Which factor contribute most to empower farmers through e-Agriculture in Bangladesh?* has the lowest (10) (see Query 6). The articles in the data set contain on average 80.4 references (see Query 7). Moreover, the most cited reference has the PubMed ID 18444910 (*Mechanisms of salinity tolerance*), with 9 cites by the articles in the data set (see Query 8). Each reference is cited on average once (see Query 9).

Journals and publishers are also covered by the knowledge graph. The articles in the data set have been published in 29 journals, being PLoS ONE the journal with the largest number of articles (see Figure 6) (see Query 10). On the other hand, MDPI and Public Library of Science are the publishers with the largest number of articles in the data set with 34 and 25 articles, respectively (see Figure 7) (see Query 11).

#### 4.2.2. Analysis of the Annotations Set

305,853 annotations were produced for the 127 articles (see Query 12). We can extract the most common annotations produced in the whole data set by querying the knowledge graph (see Query 13). These annotations include general domain terms such as PLANTS, STRESS, RICE, or SOIL among others (see Table 1). On the other hand, the query for the least frequent annotations (see Query 14) returns that 5742 annotations appear only once in the data set (see Query 15). Table 2 shows an example set of ten such annotations, most of them referring to concrete concepts such as 1-METHYLCYCLOPROPENE or ABSCISIC ACID METABOLISM.

**Table 1.** The ten most common annotations in the whole data set, their number of occurrences and the number of articles annotated with them, sorted by the number of occurrences.

Annotation	Occurrences	Annotated Articles
PLANTS	10,960	109
PLANT	3656	112
STRESS	2365	74
SOIL	2334	88
RICE	1883	65
EXPRESSION	1683	70
GROWTH	1465	106
OTHER	1311	126
STUDY	1286	118
GENES	1283	72

**Table 2.** Example of ten annotations that only appear once in the data set.

Annotation
1-METHYLCYCLOPROPENE
1-PROPANOL
ABA RESPONSE
ABSCISIC ACID METABOLISM
ACETYLATION
ACR2
BOTANICAL GARDEN
BREAST ADENOCARCINOMA
CHAPERONIN
SESAME OIL

The specificity of a concept can be measured through its depth in the corresponding ontology. Hence, we can obtain (see Query 16) for each annotation, the number of classes that are supporting the annotation, and the average depth of such classes in the hierarchy of their source ontologies. The most frequent annotations are associated with general concepts, whereas the least frequent ones are more specific. Correspondingly, the most frequent annotations are supported, in general, by ontology classes with smaller average depth (see Table 3).

We can extract data about which ontologies are mostly represented in the annotations set. For example, Table 4 shows the ten largest ontologies together with the number of ontology entities related with annotations, the total number of entities in that ontology, and the percentage of ontology usage in annotation, which results of the division between these two previous values (see Query 17). For this purpose, a Neo4j plugin for URI management was developed (<https://github.com/fanavarro/URIManager>) to distinguish between the ontology base URI and the local identifier for an entity; for instance, the URI [http://purl.obolibrary.org/obo/G0\\_0003904](http://purl.obolibrary.org/obo/G0_0003904) is split into <http://purl.obolibrary.org/obo/G0> (ontology base URI) and 0003904 (entity identifier).

**Table 3.** Common and unique annotations together with the number of ontology classes supporting them and the average depth of these classes in their ontology. Annotations marked with \* are linked with individuals and not with classes, thus, average class depth could not be calculated.

Annotation	Annotated Classes	Average Class Depth in Hierarchy
PLANTS	7	3.57
PLANT	15	3.67
STRESS	1	2
SOIL	5	4.2
RICE	8	9.25
EXPRESSION *	0	-
GROWTH	3	4
OTHER	1	3
STUDY	1	5
GENES *	0	-
1-METHYLCYCLOPROPENE *	0	-
1-PROPANOL *	0	-
ABA RESPONSE	1	7
ABSCISIC ACID METABOLISM	1	7
ACETYLATION *	0	-
ACR2	2	9
BOTANICAL GARDEN	1	10
BREAST ADENOCARCINOMA	1	9
CHAPERONIN	1	13
SESAME OIL	2	8

**Table 4.** Set of the top-ten largest ontologies indicating, for each one, how many elements were used for annotating articles, the size of the ontology, and a percentage indicating the portion of the ontology used in annotation.

Ontology prefix	Elements Used for Annotation	Total Elements	Percentage
<a href="http://purl.bioontology.org/ontology/NCBITAXON">http://purl.bioontology.org/ontology/NCBITAXON</a>	1688	906,780	0.19%
<a href="http://taxref.mnhn.fr/lod/taxon">http://taxref.mnhn.fr/lod/taxon</a>	1174	236,507	0.5%
<a href="http://purl.obolibrary.org/obo/PR">http://purl.obolibrary.org/obo/PR</a>	2947	215,546	1.36%
<a href="http://purl.obolibrary.org/obo/GR_tax">http://purl.obolibrary.org/obo/GR_tax</a>	752	58,598	1.28%
<a href="http://purl.obolibrary.org/obo/GO">http://purl.obolibrary.org/obo/GO</a>	864	49,933	1.73%
<a href="http://www.ebi.ac.uk/efo/EFO">http://www.ebi.ac.uk/efo/EFO</a>	503	10,174	4.94%
<a href="http://purl.obolibrary.org/obo/FOODON">http://purl.obolibrary.org/obo/FOODON</a>	864	5530	15.62%
<a href="http://purl.obolibrary.org/obo/SO">http://purl.obolibrary.org/obo/SO</a>	270	4687	5.76%
<a href="http://opendata.inra.fr/PO2">http://opendata.inra.fr/PO2</a>	6	3477	0.17%
<a href="http://opendata.inra.fr/PO2_DG">http://opendata.inra.fr/PO2_DG</a>	969	3465	28%

An advantage of having a knowledge graph supported by ontologies is that the hierarchy of the ontology, defined through `subclassOf` edges in the graph, can be exploited for improving the quality of the results of queries. For instance, if we are interested in articles about polymers, not only articles annotated with POLYMERS would be retrieved but also those annotated with types of polymers. This is exemplified in Query 18, where we use the ontology PO2\_DG to obtain all the subclasses of polymers class, and then we search all articles annotated with these classes. The results of this query include articles annotated with POLYMERS, POLYACRYLAMIDE or TEFLON, among others.

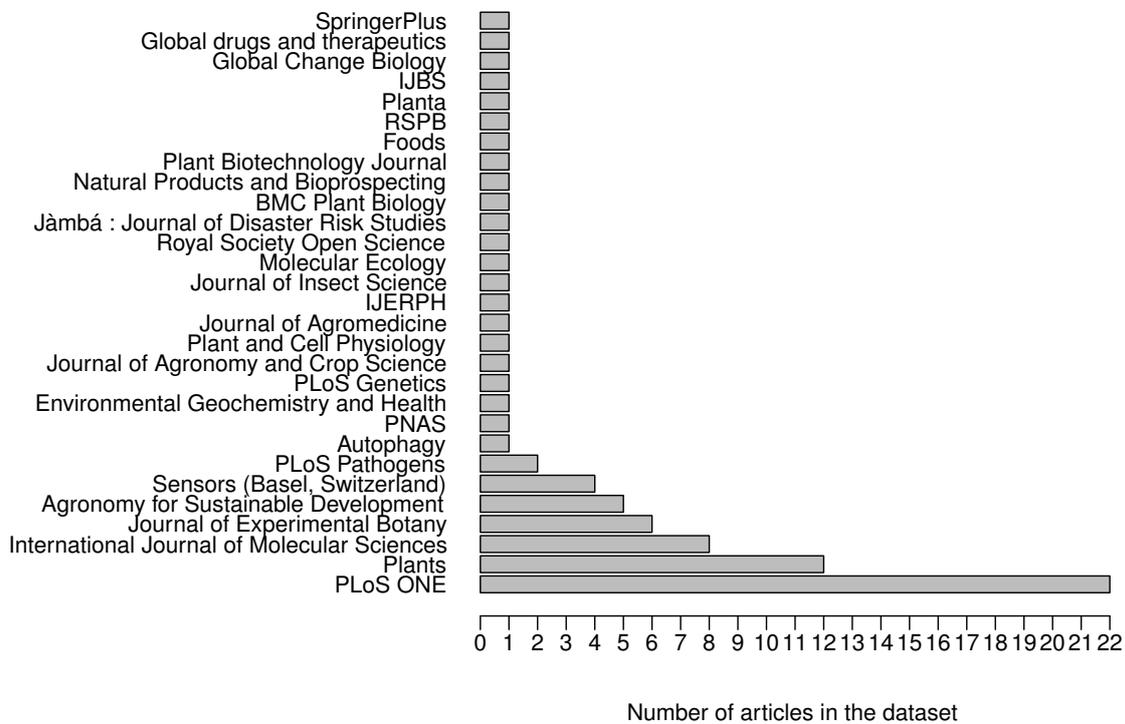


Figure 6. Number of articles per journal.

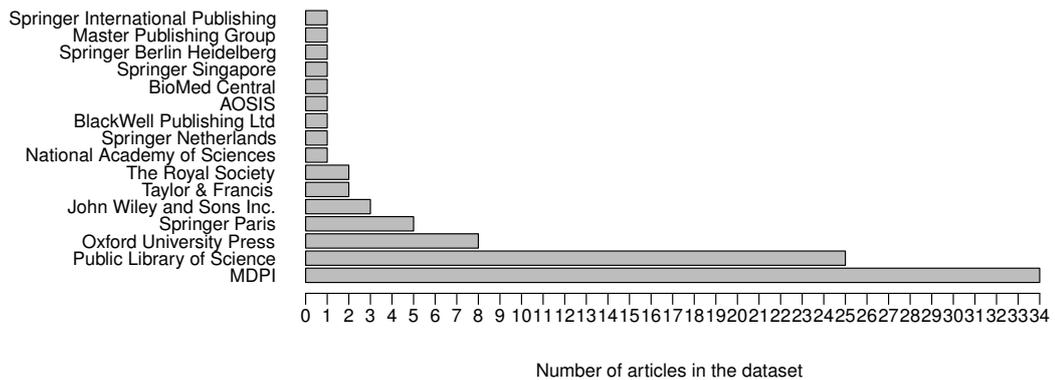


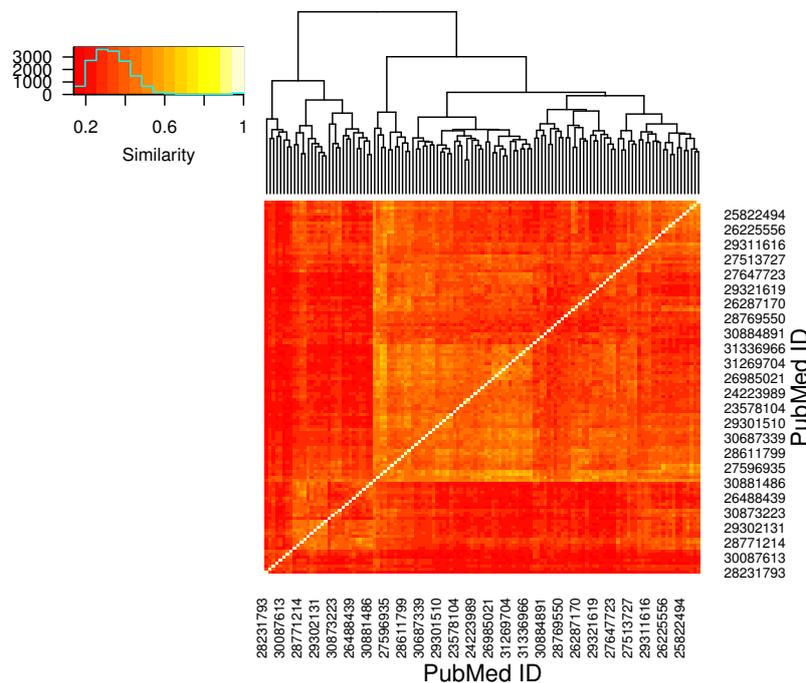
Figure 7. Number of articles per publisher.

### 4.3. Use Case 1: Similarity between Articles

An important feature in scientific literature repositories is the ability to search articles that are similar to a given one. We can use the annotations of the articles for comparing them so that a higher number of common annotations between two articles entails a higher degree of similarity. In this work, the similarity between two articles is calculated as shown in Equation 1, where  $A$  and  $B$  are the sets of annotations associated with the two articles. The implementation of this algorithm is available in the Neo4j library *Graph Algorithms (algo)* (<https://Neo4j.com/docs/graph-algorithms>).

$$O(A, B) = \frac{|A \cap B|}{\min(|A|, |B|)} \tag{1}$$

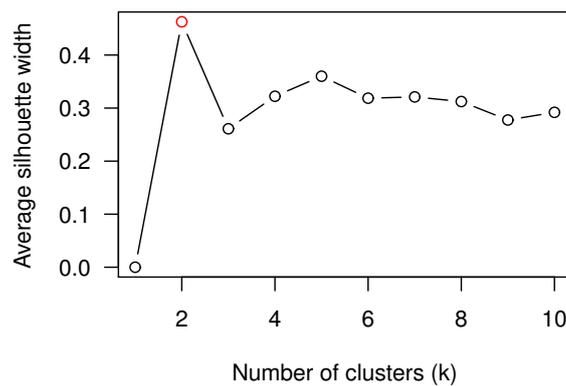
The most similar articles are *Potato Annexin STANN1 Promotes Drought Tolerance and Mitigates Light Stress in Transgenic Solanum tuberosum L. Plants* (pmid = 26172952) and *Expression of miR159 Is Altered in Tomato Plants Undergoing Drought Stress* (pmid = 31269704), with a similarity of 0.69. In contrast, the lowest similarity (0.14) is reached for the articles *Tolerance of Transplastomic Tobacco Plants Overexpressing a Theta Class Glutathione Transferase to Abiotic and Oxidative Stresses* (pmid = 30687339) and *Folk use of medicinal plants in Karst and Gorjanci, Slovenia*. pmid = 28231793 (see Query 19). The clustered heatmap in Figure 8, obtained with R [38] and gplots [39], shows the similarity between all articles and a histogram indicating the similarity distribution (note that not all article labels are included in the figure due to readability reasons).



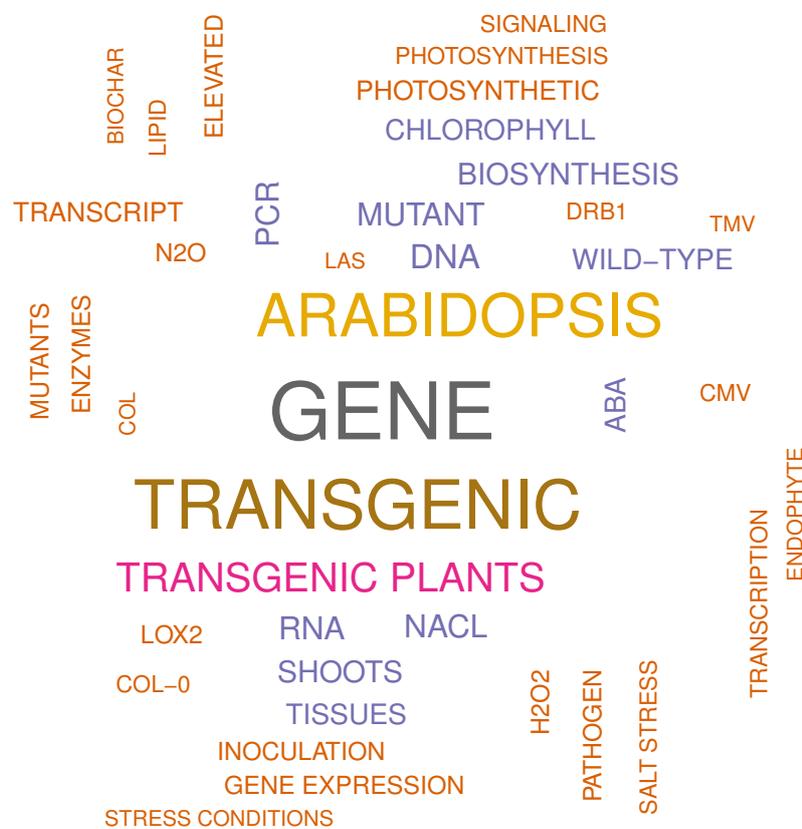
**Figure 8.** Heatmap showing the similarity between all articles in the data set, together with an histogram indicating how many article pairs exist with a concrete similarity.

We analyzed the underlying clusters resulting from the heatmap calculation. We used the *dendextend* R package [40] to estimate the optimal number of clusters  $k$ , by applying the average silhouette width method [41]. Figure 9 shows that the optimal number of clusters is 2, which is in concordance with using the similarity score for the clustering. The largest cluster includes 94 articles which present similarity between them. The second cluster contains 31 articles with low similarity.

Finally, we extracted the most significant annotations for related articles from the cluster. For this, we generated a data set per cluster, which included the annotations of the articles in such cluster and their frequency in the cluster. The annotations appearing in both data sets were removed to keep only cluster-specific annotations. Finally, we can obtain the ranking of annotations representing the cluster by sorting them by frequency. Figure 10 shows a word cloud graph, obtained with *wordcloud* R package [42], including the forty most frequent annotations in the cluster of articles with similar documents.



**Figure 9.** Estimation of the optimal number of clusters by using the average silhouette width.



**Figure 10.** Word cloud with the forty most frequent annotations representing the cluster of articles with similar documents.

4.4. Use Case 2: Annotations in the Same Context

The graph model used for annotating the scientific articles provides a paragraph level granularity, that is, annotations are associated with a concrete paragraph in an article. Thus, it is possible to retrieve annotations that usually appear in the same context (paragraph). This is important because it enables interesting features such as identifying related concepts that could be used for disambiguation.

For instance, the annotation SIGMA FACTORS appears in 3 paragraphs. Two of these paragraphs are part of the article with PubMed ID 28948393, while the other one appear in the article 25244327 (see Query 20). All these paragraphs have the following annotations in common: FACTORS, GENES, TRANSCRIPTION, SIGMA FACTORS and EXPRESSION (see Query 21). In this case, we have a context for SIGMA FACTORS conformed by genetic concepts. Thus, all these annotations can be classified in the

same domain. This could help to disambiguate the annotation EXPRESSION, for example, which is referring to gene expression in this context, but could act as synonym of formula or equation in other contexts. This is exemplified in the following paragraph, which was taken from the article with PubMed ID 30486880 (see Query 22):

In this **expression**, Nur indicates the total number of plant species used to extract a certain color and Nt refers to the number of species simultaneously approved by all informants for dyeing a certain color. ICF values range between 0 and 1, with 0 indicating the highest level of informant consent and 1 the lowest. To quantify the use frequency of certain species [[CR22]], the below formula was adopted.

The annotations found in this paragraph are the following: SPECIES, DYEING, NUMBER OF, NUMBER OF PLANT, FORMULA, EXPRESSION, TOTAL, NUR, TO EXTRACT, COLOR, PLANT SPECIES, NUMBER, PLANT, CONSENT, FREQUENCY, LEVEL, RANGE, VALUES and ICF. In this case, this context does not contain any genetic concept but mathematical terms like FORMULA, NUMBER OF or FREQUENCY, which can help to define the meaning of EXPRESSION in a mathematical domain.

## 5. Discussion

In this paper, we have presented a workflow to generate knowledge graphs about scientific literature, providing an example in the agronomy domain. Our model includes the information about articles (authors, cites, sections, contents, etc.) and the semantic annotations on the contents, together with the ontologies from which these annotations come from. Next, we discuss the most relevant aspects of our work.

### 5.1. Progress beyond the State-of-the-Art

The inclusion of the contents of the articles is a differentiating point between our model and some state-of-the-art solutions commented on in Section 1. The models in Nature SciGraph and The Semantic Lancet project [16] only represent and process the abstracts of the articles. Although abstracts should summarize the whole article, other sections usually have relevant information to classify or extract features from the article. For this reason, our model is in line with OpenBioDiv, which provides the full text content organized by sections. In our case, our model also divides the content of a section in paragraphs, providing a finer granularity. Moreover, having a model containing the whole structure of the articles allows for the development of visualization applications since all the information is available.

Another differentiating factor in our model is the semantic enrichment. Nature SciGraph and OpenBioDiv only provides keywords for each article. In the case of Nature SciGraph, these keywords are taken from FoR, a standard classification for fields of research, while OpenBioDiv is not using any controlled vocabulary. For its part, *The Semantic Lancet project* generates an RDF representation of the abstracts, which involves the generation of RDF triples describing the meaning from this textual content. This translation from natural language to RDF is complex and it probably requires human supervision to avoid errors in the process. This could make the resulting RDF graphs difficult to compare because of a lack of homogeneity between them. This problem is not present in the model provided by *Open Research Knowledge Graph* (ORKG) [19]. In this case, they train a neural network for named entity recognition, which classifies entities as process, method, material and data, thus providing an homogeneous model that enables comparison between articles. Nonetheless, classifying instances directly under this four generic classes derives in a lack of domain knowledge that prevents the semantic understanding of the meaning of each instance. In our case, our semantic enrichment process consists on annotating the articles with entities from domain ontologies. Thus, these ontologies, which are also included in the knowledge graph, provide the semantic context for each annotation.

### 5.2. New Possibilities for Literature Management

This semantic context empowers interesting features in literature management. For instance, we can find articles annotated with a set of related concepts rather than with a single concept. This has been exemplified in Section 4.2, where we have shown a query for retrieving all articles annotated with “polymer”, together with any concept taxonomically linked to polymer. Also, thanks to having the ontologies in the knowledge graph, we could measure the specificity of the annotations by checking the depth of the concepts in their corresponding ontology (see Section 4.2). This metric could help to determine which ontology concepts are specific enough to find relevant articles in a domain, or those that are too general. The latter could be a sign of heterogeneity of the set of articles. Furthermore, we also measured ontology usage in the annotation process by checking how many terms of each ontology were used in the annotations. This serves to calculate the degree of usage of each ontology. This kind of metric could help to (1) identify the ontologies that are not representing the data set, or (2) improve those ontologies that represent the data set in a certain degree, but they could be complemented by adding new elements.

Several use cases are shown in Section 4. In these examples, we showed how our model contains enough information to compute similarity between articles (see Section 4.3) and context extraction (see Section 4.4). Regarding the similarity, we were able to implement only with Neo4j features a similarity measure between all articles based on their common annotations. Then, we used these similarity values as a feature to calculate a clustering in R, which resulted in two clusters: related and unrelated articles. Nonetheless this is only a usage example; we could have calculated the clustering by using the frequency of each annotation in each article as feature, which would have resulted in different clusters of articles, where each cluster would represent a set of topics. In addition to this, our model is able to retrieve all annotations in a paragraph, which helps to determine what annotations usually appear in the same context. This could be useful for the disambiguation of annotations, as shown in Section 4.4.

### 5.3. Property Graphs for RDF Representation

In this work we have explored the use of Neo4j for storing and querying our data, in opposition to the use of RDF native triplestores as Virtuoso [43] or GraphDB (<https://www.ontotext.com/products/graphdb/>). The main difficulty was how to describe RDF blank nodes or OWL anonymous classes, which are used for defining axioms in domain ontologies. There is no a direct way to represent this kind of axioms in a property graph. Consequently, we avoid them when loading ontologies into the graph, as described in Section 3.3. Another important limitation of Neo4j against triplestores is the impossibility of having more than one graph in a single instance; or the lack of mechanisms to query different databases at once, feature offered by the RDF triplestores implementing SPARQL1.1 federated queries. On the other hand, Cypher, the Neo4j query language, is easier to use than SPARQL, and its functionality could be easily extended by means of plugins. Furthermore, there is an emerging community that supports Neo4j with new functions or visualization tools.

### 5.4. Limitations and Further Work

Finally, this work presents some limitations that we plan to improve in the future. The current graph contains 127 articles, which is a small number in comparison to the number of articles published. This is due to the fact that the input to our method are files in JATS format, which is not currently provided by many journals. PubMed is one of the few resources providing content in JATS format, so we used it as content source. However, PubMed is more oriented to biomedicine, so the number of papers related to agronomy is low. Therefore, we will extend our tools to accept input data in other formats, which will enable the inclusion of data from more sources. Another aspect that could be improved is the annotation context. At the moment, our model relates annotations at the level of paragraphs; however, sometimes these paragraphs are large and they could contain annotations that are semantically far between them. Therefore, we could improve this by relating annotations at the

level of sentence, which would provide a more accurate context. Finally, the current Neo4j knowledge graph can be difficult to query and exploit by non-experienced users. Thus, we plan to fill the gap between the knowledge graph and the final user by developing a web query interface.

## 6. Conclusions

In this paper, we have presented a workflow to generate scientific literature knowledge graphs in the agriculture domain by using an emerging technology, Neo4j. Our method takes advantage of the possibilities offered by the Semantic Web and natural language processing technologies to generate semantically-rich graphs that contain the data, their semantic annotations and domain knowledge. We have shown how this combination facilitates the execution of knowledge-based complex data exploitation tasks. Hence, we believe that our work provides an example of how Semantic Web technologies can be effectively applied for the management of agriculture data and knowledge.

**Author Contributions:** Conceptualization, A.G.-C.; software, F.A.-N. and J.A.B.-D.; validation, F.A.-N. and J.A.B.-D.; formal analysis, F.A.-N.; investigation, F.A.-N., J.A.B.-D., A.G.-C. and J.T.F.-B.; writing—original draft preparation, F.A.-N.; writing—review and editing, F.A.-N., J.A.B.-D., A.G.-C. and J.T.F.-B.; supervision, A.G.-C. and J.T.F.-B. All authors have read and agreed to the published version of the manuscript.

**Funding:** “This research was partially funded by the Ministerio de Economía, Industria y Competitividad, Gobierno de España and the European Regional Development Fund through grant number TIN2017-85949-C2-1-R, and by the Fundación Séneca, Agencia Regional de Ciencia y Tecnología through grant number 20963/PI/18.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

## Appendix A. Articles in the Data Set

The data set used in this paper contains the following articles (pmcid):

PMC3310815  
PMC3547067  
PMC3668195  
PMC3672096  
PMC3676804  
PMC3676838  
PMC3818224  
PMC3904951  
PMC3951315  
PMC4067337  
PMC4112640  
PMC4128675  
PMC4171492  
PMC4237146  
PMC4378629  
PMC4379157  
PMC4392563  
PMC4397498  
PMC4421779  
PMC4427476  
PMC4449599  
PMC4501783  
PMC4520592  
PMC4581289

PMC4598160  
PMC4623198  
PMC4720047  
PMC4824577  
PMC4844397  
PMC4867051  
PMC4879578  
PMC4898372  
PMC4913302  
PMC4932627  
PMC4981420  
PMC4998141  
PMC5011652  
PMC5052518  
PMC5055509  
PMC5066492  
PMC5095167  
PMC5125658  
PMC5253511  
PMC5299730  
PMC5313494  
PMC5324297  
PMC5362684  
PMC5373544  
PMC5413563  
PMC5441857  
PMC5447229  
PMC5465592  
PMC5472609  
PMC5521873  
PMC5559270  
PMC5579920  
PMC5620588  
PMC5633626  
PMC5637878  
PMC5669304  
PMC5688476  
PMC5738964  
PMC5755014  
PMC5758783  
PMC5762720  
PMC5763371  
PMC5800158  
PMC5853279  
PMC5860692  
PMC5882813  
PMC5908804  
PMC5935394

PMC5995558  
PMC6013986  
PMC6021986  
PMC6055696  
PMC6066661  
PMC6070203  
PMC6084956  
PMC6166800  
PMC6191707  
PMC6198449  
PMC6213855  
PMC6215673  
PMC6232530  
PMC6242374  
PMC6262949  
PMC6277847  
PMC6322579  
PMC6326430  
PMC6337918  
PMC6358044  
PMC6366173  
PMC6376693  
PMC6390929  
PMC6390932  
PMC6394436  
PMC6399567  
PMC6404675  
PMC6412671  
PMC6417397  
PMC6417402  
PMC6449481  
PMC6471123  
PMC6471620  
PMC6472519  
PMC6473438  
PMC6514985  
PMC6524069  
PMC6524378  
PMC6529577  
PMC6538708  
PMC6539879  
PMC6539957  
PMC6560427  
PMC6570029  
PMC6571617  
PMC6587683  
PMC6630288  
PMC6630593  
PMC6630798  
PMC6681330

PMC6681344  
 PMC6681968  
 PMC6724085  
 PMC6730492  
 PMC6736833

## Appendix B. Cypher Queries

This appendix contains all the Cypher queries used in the article for extracting information from the knowledge graph. These are referenced from the corresponding sections.

Query 1: Cypher query to retrieve the number of nodes and edges in the knowledge graph

```
MATCH (a)
WITH count(a) as nodes
MATCH ()-[r]->()
WITH nodes, count(r) as edges
RETURN nodes, edges
```

Query 2: Cypher query to retrieve the number of authors per article, sorted by the number of authors

```
MATCH (article:bibo__AcademicArticle) -[:bibo__authorList]->
(authorList:rdf__Seq) --> (author:foaf__Person)
RETURN article.dct__title, count(author) as authors
ORDER BY authors desc
```

Query 3: Cypher query to retrieve the average number of authors per article

```
MATCH (article:bibo__AcademicArticle) -[:bibo__authorList]->
(authorList:rdf__Seq) --> (author:foaf__Person)
WITH article.dct__title as title, count(author) as authors
RETURN avg(authors)
```

Query 4: Cypher query to retrieve the number of articles per author

```
MATCH (author:foaf__Person) -[:foaf__publications]-> (article)
RETURN author.foaf__name, count(article) as articles
ORDER BY articles desc
```

Query 5: Cypher query to retrieve the average number of articles per author

```
MATCH (author:foaf__Person) -[:foaf__publications]-> (article)
WITH author.foaf__name as authorName, count(article) as articles
RETURN avg(articles)
```

Query 6: Cypher query to retrieve the number of references per article, sorted by the number of references

```
MATCH (article:bibo__AcademicArticle) -[:bibo__cites]-> (reference)
RETURN article.dct__title, count(reference) as references
ORDER BY references desc
```

Query 7: Cypher query to retrieve the average number of references per article

```
MATCH (article:bibo__AcademicArticle) -[:bibo__cites]-> (reference)
WITH article.dct__title as title, count(reference) as references
RETURN avg(references)
```

Query 8: Cypher query to retrieve how many times a reference is cited

```
MATCH (reference) -[:bibo__citedBy]->(article:bibo__AcademicArticle)
RETURN reference.uri as reference, count(article) as articles
ORDER BY articles desc
```

Query 9: Cypher query to retrieve the average of how many times a reference is cited

```
MATCH (reference) -[:bibo__citedBy]->(article:bibo__AcademicArticle)
WITH reference, count(article) as articles
RETURN avg(articles)
```

Query 10: Cypher query to retrieve the number of the articles per journal

```
MATCH (article:bibo__AcademicArticle) -[:dct__isPartOf]->
(issue:bibo__Issue) -[:dct__isPartOf]->(journal:bibo__Journal)
RETURN journal.dct__title, count(article) as articles
ORDER BY articles desc
```

Query 11: Cypher query to retrieve the number of the articles per publisher

```
MATCH (article:bibo__AcademicArticle) -[:dct__isPartOf]->
(issue:bibo__Issue) -[:dct__isPartOf]->
(journal:bibo__Journal) -[:dct__publisher]->
(publisher:foaf__Organization)
RETURN publisher.foaf__name, count(distinct article) as articles
ORDER BY articles desc
```

Query 12: Cypher query to retrieve the number of the annotations in the data set

```
MATCH (annotation:aot__ExactQualifier)
WHERE annotation.aoc__body <> 'UNAVAILABLE'
RETURN sum(annotation.biotea__tf)
```

Query 13: Cypher query to retrieve the top ten most frequent annotations in the data set

```
MATCH (annotation:aot__ExactQualifier) -[:aoc__annotatesResource]->
(article:bibo__AcademicArticle)
WHERE annotation.aoc__body <> 'UNAVAILABLE'
RETURN annotation.aoc__body,
sum(annotation.biotea__tf) as 'annotation count',
count(article) as articles
ORDER BY 'annotation count' desc limit 10
```

Query 14: Cypher query to retrieve the annotations that appear only once in the dataset

```
MATCH (annotation:aot__ExactQualifier) -[:aoc__annotatesResource]->
(article:bibo__AcademicArticle)
WITH annotation.aoc__body as text,
sum(annotation.biotea__tf) as 'annotation count',
count(article) as 'article count'
WHERE 'annotation count' = 1
RETURN text, 'annotation count', 'article count'
```

Query 15: Cypher query to retrieve the number of the annotations that only appear once in the data set

```
MATCH (annotation:aot__ExactQualifier)
WITH annotation.aoc__body as text,
sum(annotation.biotea__tf) as 'annotation count'
WHERE 'annotation count' = 1
RETURN count(text)
```

Query 16: Cypher query to retrieve annotations, the number of classes that are supporting each annotation, and the average depth in which these classes are in the ontology hierarchy

```
MATCH (topic:Class)<-[:aoc__hasTopic]-:(annotation:aot__ExactQualifier)
WHERE annotation.aoc__body in ['PLANTS', 'PLANT', 'STRESS', 'SOIL', 'RICE',
'EXPRESSION', 'GROWTH', 'OTHER', 'STUDY', 'GENES',
'1-METHYLCYCLOPROPENE', '1-PROPANOL', 'ABA RESPONSE',
'ABSCISIC ACID METABOLISM', 'ACETYLATION', 'ACR2',
'BOTANICAL GARDEN', 'BREAST ADENOCARCINOMA', 'CHAPERONIN',
'SESAME OIL']
WITH topic, annotation
MATCH (root:Class) where not (root)-[:subClassOf]->()
WITH topic, root, annotation
MATCH path = (topic)-[:subClassOf*]->(root)
WITH length(path) as pathLength,
topic,
annotation
WITH annotation.aoc__body as text,
topic.uri as topicUri,
min(pathLength) + 1 as depth
RETURN text,
count(topicUri) as annotatedClasses,
avg(depth) as averageDepth
```

Query 17: Cypher query to retrieve the number of ontology elements used in annotations per ontology, together with the total number of elements in the ontology and a percentage indicating the proportion of the elements used in annotations with respect to the total elements in each ontology.

```
MATCH (annotation:aot__ExactQualifier)-[:aoc__hasTopic]->(concept:Class)
WHERE annotation.aoc__body <> 'UNAVAILABLE'
WITH distinct concept.uri as concept
WITH URIManager.getNamespace(concept) as ontologyPrefix,
count(concept) as usedElements
MATCH (element:Class)
WHERE element.uri STARTS WITH ontologyPrefix
WITH ontologyPrefix,
usedElements,
count (distinct element.uri) as totalElements
RETURN ontologyPrefix,
usedElements,
totalElements,
(100.0*usedElements)/totalElements as percentage
ORDER BY totalElements desc
```

Query 18: Cypher query to retrieve all articles annotated with polymers or any of its sub classes according PO2\_DG ontology

```

MATCH (polymerTopic:Class)<-[:subClassOf*]- (subTopics:Class)
WHERE polymerTopic.uri = 'http://opendata.inra.fr/PO2_DG/product/polymers'
WITH COLLECT(polymerTopic) + COLLECT(subTopics) as topicList
UNWIND topicList as topics
MATCH (article:bibo__AcademicArticle)<-
[:aoc__annotatesResource]-
(annotation:aot__ExactQualifier)-
[:aoc__hasTopic]->(topics)
RETURN article.dct__title , annotation.aoc__body

```

Query 19: Cypher query to compute the overlap similarity between each pair of articles according their common annotations

```

MATCH (topic:Resource)<-[:aoc__hasTopic]-
(annotation:aot__ExactQualifier)-[:aoc__annotatesResource]->
(article:bibo__AcademicArticle)
WHERE annotation.aoc__body <> 'UNAVAILABLE'
WITH {item:id(article),
categories: collect(distinct id(topic))} as userData
WITH collect(userData) as data
CALL algo.similarity.overlap.stream(data)
YIELD item1, item2, count1, count2, intersection, similarity
RETURN algo.asNode(item1).dct__title AS from,
algo.asNode(item2).dct__title AS to,
count1, count2, intersection, similarity
ORDER BY similarity DESC

```

Query 20: Cypher query to retrieve the paragraphs and the articles in which the annotation SIGMA FACTORS appears, and the other annotations within the paragraphs

```

MATCH (annotation:aot__ExactQualifier{aoc__body:"SIGMA FACTORS"})
-[:aoc__context]->(context:biotea__ElementSelector)
-[:dct__references]->(paragraph:doco__Paragraph)
WITH paragraph
MATCH (otherAnnotations:aot__ExactQualifier)-[:aoc__context]->
(context:biotea__ElementSelector)
-[:dct__references]->(paragraph)-[:dct__isPartOf*]->
(article:bibo__AcademicArticle)
RETURN article.bibo__pmid as pmid,
paragraph.rdf__value as paragraph,
collect(distinct otherAnnotations.aoc__body) as annotations

```

Query 21: Cypher query to retrieve the annotations that are always appearing together with SIGMA FACTORS in the same paragraph

```

MATCH(annotation:aot__ExactQualifier{aoc__body:"SIGMA FACTORS"})
-[:aoc__context]->(context:biotea__ElementSelector)
-[:dct__references]->(paragraph:doco__Paragraph)
WITH paragraph
MATCH (otherAnnotations:aot__ExactQualifier)-[:aoc__context]->

```

```
( context : biotea__ElementSelector ) -[:dct__references]->
( paragraph )
WITH paragraph.rdf__value as paragraphText ,
collect( distinct otherAnnotations.aoc__body ) as annotationsPerParagraph
WITH collect( annotationsPerParagraph ) as annotations
WITH reduce( commonAnnotations = head( annotations ) ,
annotation in tail( annotations ) |
apoc.coll.intersection( commonAnnotations , annotation ) ) as commonAnnotations
RETURN commonAnnotations
```

Query 22: Cypher query to retrieve the annotations belonging to a concrete paragraph, together with the paragraph text

```
MATCH ( annotations : aot__ExactQualifier ) -[:aoc__context]->
( context : biotea__ElementSelector ) -[:dct__references]->
( paragraph : doco__Paragraph )
WHERE paragraph.uri = 'http://purl.org/io/linkingdata/
paragraph/pmcdoc_resource/6262949/
Methods_Statistical-analysis:para_2'
RETURN paragraph.rdf__value as text ,
COLLECT( annotations.aoc__body ) as annotations
```

## References

1. Frazzetto, G. The changing identity of the scientist: As science puts on a new face, the identity of its practitioners evolves accordingly. *EMBO Rep.* **2004**, *5*, 18–20. [[CrossRef](#)] [[PubMed](#)]
2. Price, D.J. *Little Science, Big Science... and Beyond*; Columbia University Press: New York, NY, USA, 1986.
3. Bornmann, L.; Mutz, R. Growth rates of modern science: A bibliometric analysis based on the number of publications and cited references. *J. Assoc. Inf. Sci. Technol.* **2015**, *66*, 2215–2222. [[CrossRef](#)]
4. Berners-Lee, T.; Hendler, J.; Lassila, O. The semantic web. *Sci. Am.* **2001**, *284*, 28–37. [[CrossRef](#)]
5. McGuinness, D.L.; Van Harmelen, F. OWL web ontology language overview. *W3C Recomm.* **2004**, *10*, 2004.
6. Miller, E. An introduction to the resource description framework. *Bull. Am. Soc. Inf. Sci. Technol.* **1998**, *25*, 15–19. [[CrossRef](#)]
7. Seaborne, A.; Prud'hommeaux, E. SPARQL Query Language for RDF. W3C Recommendation, W3C, 2008. Available online: <http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/> (accessed on 22 January 2020).
8. Shotton, D. Semantic publishing: The coming revolution in scientific journal publishing. *Learn. Publ.* **2009**, *22*, 85–94. [[CrossRef](#)]
9. Castro, L.J.G.; McLaughlin, C.; Garcia, A. Biotea: RDFizing PubMed Central in support for the paper as an interface to the Web of Data. *J. Biomed. Semant. BioMed. Cent.* **2013**, *4*, S5. [[CrossRef](#)]
10. Garcia, A.; Lopez, F.; Garcia, L.; Giraldo, O.; Bucheli, V.; Dumontier, M. Biotea: Semantics for Pubmed Central. *PeerJ* **2018**, *6*, e4201. [[CrossRef](#)]
11. Weibel, S. The Dublin Core: A simple content description model for electronic resources. *Bull. Am. Soc. Inf. Sci. Technol.* **1997**, *24*, 9–11. [[CrossRef](#)]
12. Huh, S. Journal Article Tag Suite 1.0: National Information Standards Organization standard of journal extensible markup language. *Sci. Ed.* **2014**, *1*, 99–104. [[CrossRef](#)]
13. Guha, R.V.; Brickley, D.; Macbeth, S. Schema.org: Evolution of structured data on the web. *Commun. ACM* **2016**, *59*, 44–51. [[CrossRef](#)]
14. Vanclay, J.K. An evaluation of the Australian Research Council's journal ranking. *J. Inf.* **2011**, *5*, 265–274. [[CrossRef](#)]
15. Sporny, M.; Longley, D.; Kellogg, G.; Lanthaler, M.; Lindström, N. JSON-LD 1.0. *W3C Recomm.* **2014**, *16*, 41.

16. Bagnacani, A.; Ciancarini, P.; Di Iorio, A.; Nuzzolese, A.G.; Peroni, S.; Vitali, F. The semantic lancet project: A linked open dataset for scholarly publishing. In *International Conference on Knowledge Engineering and Knowledge Management*; Springer: New York, NY, USA, 2014; pp. 101–105.
17. Peroni, S.; Shotton, D. The SPAR ontologies. In *International Semantic Web Conference*; Springer: New York, NY, USA, 2018; pp. 119–136.
18. Gangemi, A.; Presutti, V.; Recupero, D.R.; Nuzzolese, A.G.; Draicchio, F.; Mongiovì, M. Semantic Web Machine Reading with FRED. *Semant. Web* **2017**, *8*, 873–893. [[CrossRef](#)]
19. Jaradeh, M.Y.; Oelen, A.; Farfar, K.E.; Prinz, M.; D'Souza, J.; Kismihók, G.; Stocker, M.; Auer, S. Open Research Knowledge Graph: Next Generation Infrastructure for Semantic Scholarly Knowledge. In *Proceedings of the 10th International Conference on Knowledge Capture, Marina Del Rey, CA, USA, 19–21 November 2019*; pp. 243–246.
20. Penev, L.; Dimitrova, M.; Senderov, V.; Zhelezov, G.; Georgiev, T.; Stoev, P.; Simov, K. OpenBiodiv: A Knowledge Graph for Literature-Extracted Linked Open Data in Biodiversity Science. *Publications* **2019**, *7*, 38. [[CrossRef](#)]
21. Senderov, V.; Simov, K.; Franz, N.; Stoev, P.; Catapano, T.; Agosti, D.; Sautter, G.; Morris, R.A.; Penev, L. OpenBiodiv-O: Ontology of the OpenBiodiv knowledge management system. *J. Biomed. Semant.* **2018**, *9*, 5. [[CrossRef](#)] [[PubMed](#)]
22. Lebo, T.; Sahoo, S.; McGuinness, D.; Belhajjame, K.; Cheney, J.; Corsar, D.; Garijo, D.; Soiland-Reyes, S.; Zednik, S.; Zhao, J. Prov-o: The prov ontology. *W3C Recomm.* **2013**, *30*.
23. Vrandečić, D. Wikidata: A new platform for collaborative data collection. In *Proceedings of the 21st International Conference on World Wide Web, Lyon, France, 16–20 April 2012*; pp. 1063–1064.
24. Dumontier, M.; Baker, C.J.; Baran, J.; Callahan, A.; Chepelev, L.; Cruz-Toledo, J.; Del Rio, N.R.; Duck, G.; Furlong, L.I.; Keath, N.; et al. The SemanticScience Integrated Ontology (SIO) for biomedical research and knowledge discovery. *J. Biomed. Semant.* **2014**, *5*, 14. [[CrossRef](#)]
25. Constantin, A.; Peroni, S.; Pettifer, S.; Shotton, D.; Vitali, F. The document components ontology (DoCO). *Semant. Web* **2016**, *7*, 167–181. [[CrossRef](#)]
26. Ciccarese, P.; Ocana, M.; Castro, L.J.G.; Das, S.; Clark, T. An open annotation ontology for science on web 3.0. *J. Biomed. Semant. BioMed Cent.* **2011**, *2*, S4. [[CrossRef](#)]
27. Ciccarese, P.; Soiland-Reyes, S.; Belhajjame, K.; Gray, A.J.; Goble, C.; Clark, T. PAV ontology: Provenance, authoring and versioning. *J. Biomed. Semant.* **2013**, *4*, 37. [[CrossRef](#)] [[PubMed](#)]
28. Miller, J.J. Graph database applications and concepts with Neo4j. In *Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA, 8–9 March 2013*; Volume 2324.
29. Robinson, I.; Webber, J.; Eifrem, E. *Graph Databases*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2013.
30. Bouhali, R.; Laurent, A. Exploiting RDF open data using NoSQL graph databases. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*; Springer: New York, NY, USA, 2015; pp. 177–190.
31. Jonquet, C.; Toulet, A.; Arnaud, E.; Aubin, S.; Yeumo, E.D.; Emonet, V.; Graybeal, J.; Laporte, M.A.; Musen, M.A.; Pesce, V.; et al. AgroPortal: A vocabulary and ontology repository for agronomy. *Comput. Electron. Agric.* **2018**, *144*, 126–143. [[CrossRef](#)]
32. Baker, T.; Caracciolo, C.; Doroszenko, A.; Suominen, O. GACS core: Creation of a global agricultural concept scheme. In *Research Conference on Metadata and Semantics Research*; Springer: New York, NY, USA, 2016; pp. 311–316.
33. Avril, P.; Bernard, É.; Corvaisier, M.; Girard, A.; Golik, W.; Nédellec, C.; Touzé, M.L.; Wacrenier, N. Analyser la production scientifique d'un département de recherche: construction d'une ressource termino-ontologique par des documentalistes. *Cahier des Tech. de l'INRA* **2016**, *89*, 1–12.
34. Buttigieg, P.L.; Morrison, N.; Smith, B.; Mungall, C.J.; Lewis, S.E.; Consortium, E. The environment ontology: Contextualising biological and biomedical entities. *J. Biomed. Semant.* **2013**, *4*, 43. [[CrossRef](#)]
35. Dooley, D.M.; Griffiths, E.J.; Gosal, G.S.; Buttigieg, P.L.; Hoehndorf, R.; Lange, M.C.; Schriml, L.M.; Brinkman, F.S.; Hsiao, W.W. FoodOn: A harmonized food ontology to increase global food traceability, quality control and data integration. *NPJ Sci. Food* **2018**, *2*, 23. [[CrossRef](#)]
36. Martini, D.; Schmitz, M.; Mietzsch, E. agroRDF as a Semantic Overlay to agroXML: A General Model for Enhancing Interoperability in Agrifood Data Standards. In *Proceedings of the CIGR Conference on Sustainable Agriculture through ICT Innovation, Turin, Italy, 24–27 June 2013*.

37. Jackson, R.C.; Balhoff, J.P.; Douglass, E.; Harris, N.L.; Mungall, C.J.; Overton, J.A. ROBOT: A Tool for Automating Ontology Workflows. *BMC Bioinform.* **2019**, *20*, 407. [[CrossRef](#)]
38. R Core Team. *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2018.
39. Warnes, G.R.; Bolker, B.; Bonebakker, L.; Gentleman, R.; Liaw, W.H.A.; Lumley, T.; Maechler, M.; Magnusson, A.; Moeller, S.; Schwartz, M.; et al. *gplots: Various R Programming Tools for Plotting Data*; R package version 3.0.1.; The R Foundation: Vienna, Austria, 2016.
40. Galili, T. dendextend: An R package for visualizing, adjusting, and comparing trees of hierarchical clustering. *Bioinformatics* **2015**.10.1093/bioinformatics/btv428. [[CrossRef](#)]
41. Rousseeuw, P.J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **1987**, *20*, 53–65. [[CrossRef](#)]
42. Fellows, I. *Wordcloud: Word Clouds*; R package version 2.6.; The R Foundation: Vienna, Austria, 2018.
43. Erling, O.; Mikhailov, I. RDF Support in the Virtuoso DBMS. In *Networked Knowledge-Networked Media*; Springer: New York, NY, USA, 2009; pp. 7–24.



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).