Efficient, semantics-rich transformation and integration of large datasets

José Antonio Bernabé-Díaz[a], María del Carmen Legaz-García[b], José Manuel García[c], Jesualdo Tomás Fernández-Breis[a,*]

[a]*Departamento de Informática y Sistemas, Universidad de Murcia, IMIB-Arrixaca, CP 30100 Murcia, Spain*
[b]*Fundación para la Formación e Investigación Sanitarias de la Región de Murcia, Biomedical Informatics and Bioinformatics Platform, IMIB-Arrixaca, Calle Luis Fontes Pagán, No 9, 30003 Murcia, Spain*
[c]*Departamento de Ingeniería y Tecnología de Computadores, Universidad de Murcia, CP 30100 Murcia, Spain*

Abstract

The digital age is making more datasets available through the Internet, but their interoperability is still limited. The Semantic Web should play a fundamental role to achieve interoperable datasets. The semantic exploitation of data requires its efficient transformation into semantic formats and the integration of heterogeneous sources. The scalability of the existing tools for the semantic transformation of large volumes of data is limited or they do not provide a semantics-rich representation of the data.

The goal of this work is to show how scalable semantic data transformation processes can be designed and implemented, so addressing the first type of limitation. We propose to apply High Performance Computing techniques to overcome the scalability limitation. Our method has been implemented as an upgrade of the SWIT framework. Additional improvements for supporting the transformation process in SWIT are also described in this paper. We have evaluated the new method in two case studies in the area of bioinformatics and movies. The results show a significant speed up with respect to the original

*Corresponding author.
Email addresses: `joseantonio.bernabe1@um.es` (José Antonio Bernabé-Díaz), `mcarmen.legaz@ffis.es` (María del Carmen Legaz-García), `jmgarcia@um.es` (José Manuel García), `jfernand@um.es` (Jesualdo Tomás Fernández-Breis)

SWIT algorithm. The lessons learned in our work permit to configure semantic transformation processes efficiently.

Keywords: Semantic Web, High Performance Computing, Data transformation

1. Introduction

The digital age is making more and more datasets available through the Internet (Khan et al., 2017). Recent political and scientific decisions such as the promotion and adoption of FAIR principles (Wilkinson et al., 2016) makes us believe that the number of datasets available will grow significantly in the next years. Heterogeneity is likely to be the major problem for large scale data management (Gandomi & Haider, 2015). Data suppliers do not only differ in how they store and provide the data but also in the semantics associated with the data entities. The semantic interoperability of data is a challenge in diverse areas such as biomedicine or cyber-physical systems (Jirkovskỳ et al., 2017). Consequently, general methods and tools able to exploit efficiently those highly heterogeneous data are needed.

The Semantic Web (Berners-Lee et al., 2001) provides machine-readable information whose infrastructure is based on the World Wide Web Consortium (W3C)[1] standards. The Semantic Web offers a natural space for data integration and interoperability (Goble & Stevens, 2008), and it has opened a series of new possibilities for information processing and management in different areas such as recommender systems (Musto et al., 2017), generation of user interfaces (Kapłański et al., 2017), or sharing clinical experiences (Roldán-García et al., 2018). There has been an increase in the amount of resources available in semantic formats (Abele et al., 2017; Oliveira et al., 2017), but most of them are still in traditional ones. This is also hampering the use of interoperable data by knowledge-based systems.

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

Linked Open Data is a Semantic Web initiative for publishing and sharing the web content in a semantic format like RDF. Best practices in the Linked Open Data community, which are also reinforced by the FAIR principles for data management and stewardship, establish that ontologies should be used for restricting the meaning of RDF triples, since they provide a context/vocabulary to enrich the data. Ontologies are the cornerstone technology in the Semantic Web, and they represent common, shareable and reusable specifications of particular application domains (Gruber, 1993). OWL (Bechhofer, 2009) is the most common language to represent ontologies.

In recent years different approaches for getting RDF representations of datasets have been proposed. These approaches can be classified as follows:

- Syntactic transformation: these methods use a canonical transformation from different types of schemas (e.g., XML or relational data) to RDF (Erling & Mikhailov, 2009; Huang et al., 2015; Wangli et al., 2017). These approaches have the advantage of simplicity, but the disadvantage that the transformation is not driven by the semantics of the datasets, since all the datasets are transformed using the same rules.

- Ontology-driven transformation: these methods use ontologies to provide a precise meaning to the RDF data. The transformation rules for each dataset depend on its semantics. We think that this is the type of transformation of interest for the Web of Data. The first group of approaches, like Bio2RDF (Hu et al., 2017), use dataset specific transformation scripts. Transformation rules are not explicitly shared between datasets, so more manual effort is required to ensure interoperable transformation processes. The second group of approaches exploit mappings between data schemas and ontologies for generating RDF content. Examples of such approaches are RDB2OWL (Čerāns & Būmans, 2015), Karma (Dimou et al., 2014), and our method SWIT (Legaz-García et al., 2016). Ontology-driven transformation enables consistency checking, thus making possible to prevent the creation of logically inconsistent content.

Unfortunately, the scalability of the existing tools for the semantic transformation of large volumes of data is limited or they do not provide a real semantic representation of the data. This is mainly due to the need of applying reasoning during the process or the amount of operations needed to generate the semantic dataset, for instance, to guarantee that the logical consistency of the semantic dataset.

High Performance Computing (HPC) techniques have been frequently used in the last years to improve the efficiency and scalability of applications (Bourne et al., 2015; Le et al., 2018; Cafaro et al., 2018). However, in the area of semantic systems, the application of HPC techniques has mainly focused on distributed reasoning over RDF data (Sakr et al., 2018), but not to support the development of semantic web tools. We aim to contribute to the latter one. For this purpose, we propose the application of HPC techniques to the process of generating semantic datasets. This method has been implemented in our SWIT tool (Legaz-García et al., 2016; Bernabé-Díaz et al., 2018) for the semantic transformation and integration of datasets.

The contributions of this paper are: 1) the complete description of the HPC SWIT tool, which is a domain independent framework for the efficient generation of semantics rich repositories; 2) the detailed description of the application of the HPC techniques to the SWIT algorithm, whose execution time has drastically reduced; and 3) the lessons learned to ease the parallel execution of the tool. The paper illustrates how semantic web and high-performance computing technologies can be leveraged for semantics-rich transformation and integration of large datasets.

The paper is structured as follows. Section 2 describes the original SWIT method and the design of its scalable version. The results obtained in two case studies, namely, orthology and movies, are described in Section 3. Finally, some discussion and conclusions are put forward in Section 4.

2. Methods

2.1. The original SWIT framework

SWIT transforms relational or XML data into repositories in Semantic Web formats. SWIT provides semantics-rich, ontology-driven transformation and integration of datasets. SWIT repositories are not redundant and are logically consistent with the axioms of the ontology used in the process. Besides, the current version of SWIT supports the generation of five-stars open datasets[2].

SWIT has been applied in projects related to clinical data, chemical compounds and evolutionary relations between genes. This has allowed identifying the performance issue of the method for large volumes of data. SWIT executes a number of checking operations that consume a high amount of time, more than expected with large datasets. The complexity should grow linearly with the number of the entities, but this does not apply in all the cases. As it will be further detailed, the conversion of the InParanoid database (Sonnhammer & Östlund, 2014) (43 GB) required 919 computational hours.

SWIT transformation uses an input dataset, the input schema, an OWL ontology and two user-defined transformation files. The first file is the mapping rules, which define how the entities of the input schema are mapped to the ontology. The second file is the identity rules, which describe when two entities are equivalent, so preventing the generation of redundant data. This file is optional and makes sense when integration is needed.

The SWIT architecture is conceptually shown in Figure 1. The input parameters are:

- Input instances: XML/relational databases to be transformed into semantic formats such as RDF or OWL. An input example is presented in Listing 1. Such listing contains data in OrthoXML format (Schmitt et al., 2011), which will be used in our experiments.

[2]`http://www.w3.org/DesignIssues/LinkedData.html`

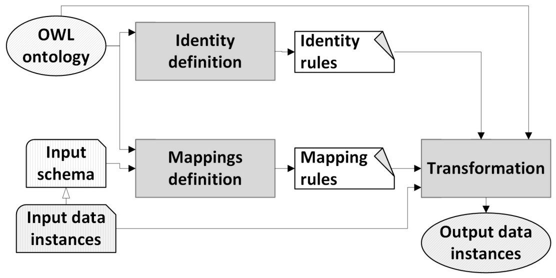

Figure 1: SWIT architecture. The framework requires the input instances, a mapping rules file and an ontology.

- OWL ontology: It provides the meaning for the transformed data, and its semantics ensures the generation of logically consistent data.

```
<species name="Escherichia-coli" NCBITaxId="83333">
 <genes>
  <gene id="1" protId="P07118" geneId="valS"/>
 </genes>
</species>
<species name="Nematocida-parisii" NCBITaxId="881290">
 <genes>
  <gene id="2" protId="I3EQN8" geneId="NEPG_00863"/>
 </genes>
</species>
<groups>
 <orthologGroup id="1">
  <geneRef id="1"/>
  <geneRef id="2"/>
 </orthologGroup>
</groups>
```

Listing 1: Reduced example of an input OrthoXML file.

- Transformation rules: They determine how the input data entities have to be transformed into ontology individuals. These transformation rules are divided into two groups: mapping rules and identity rules.

(1) **Mapping rules** control that the information represented according to the input schema is correctly transformed. We identify (1) entity rules, that define how the entities in the input dataset are converted into OWL individuals; (2) attribute rules, that define the values of the data properties of the new individuals; and (3) relation rules, that define how the new individuals are related between them. An example of an entity rule is shown in Listing 2. These rules are first defined by a 'map' clause, next the 'type' tag specifies the type of the rule: Arch2Class (entity rule), Arch2Prop (attribute rule) or Arch2Rel (relation rule). The clauses 'class' and 'arch/nodepath' describe the ontology class of the entities, and the XPath expression inside <nodepath> describes its location in the input file in the case of XML data sources. In this example, the execution of the XPath expression results in creating several individuals of the ontology class 'http://purl.org/net/orth#Gene'. For instance, taking the content of Listing 1 as input data and the mapping rule defined in Listing 2, SWIT generates two individuals 'gene_1' and 'gene_2', as the ontology class is 'http://purl.org/net/orth#Gene' and the 'nodepath' clause points to the field 'id' from the input file.

(2) **Identity rules** are in charge of detecting redundant entities in the input data (see Listing 3) and avoiding such redundancy in the semantic dataset. An example of the syntax of this rule is shown in Listing 4, meaning that two individuals of the ontology class 'Gene' are the same one if they have the same value for the property 'identifier' in a non case sensitive comparison.

An identity rule is structured as follows: A <class> clause specifies the ontology class to which the rule affects. Next, the condition of the rule is defined: AND/OR. Several <requirement> can be specified in the domain of the condition clause. The requirements dictate which property or relationship cannot have the same value for two different individuals of the same ontology class. A 'dataproperty' tag for properties, or ‘objectproperty’ one for relationships can be used. The additional clause <value> permits to specify whether the comparison is case sensitive (‘EQUALS’ or ‘EQUALS IGNORE CASE’).

```
<map>
<type>Arch2Class</type>
  <class><id>http://purl.org/net/orth#Gene</id></class>
  <arch>
    <nodepath>/species/genes/gene/@id</nodepath>
  </arch>
</map>
```

Listing 2: Class mapping rule for genes.

```
<gene id="1" protId="P07118" geneId="valS"/>
<gene id="2" protId="p07118" geneId="VaLs"/>
```

Listing 3: Example redundant data in an input file.

```
<class><id>http://purl.org/net/orth#Gene</id></class>
 <and>
  <requirement>
   <scope>ALL</scope>
   <dataproperty>http://purl.org/dc/terms/identifier</
       dataproperty>
   <value>EQUALS IGNORE CASE</value>
  </requirement>
 </and>
```

Listing 4: Identity rule for genes.

Once defined all the mapping and identity rules, SWIT generates semantic data repositories following the OWL ontology. Listing 5 shows the resulting individual which represents the previous mentioned gene ‘valS’ from *Escherichia* *coli* organism. The output generated by SWIT is an RDF/OWL dataset, which can be obtained as files using RDF/OWL and Turtle syntaxes or directly generated in an instance of the Virtuoso triple store[3].

```
<rdf:Description rdf:about="http://identifiers.org/gene/83333/
    valS">
 <dct:identifier rdf:datatype="http://www.w3.org/2001/
     XMLSchema#string">valS</dct:identifier>
 <obo:taxonomy rdf:resource="http://identifiers.org/taxonomy
     /83333"/>
 <sio:synthesize rdf:resource="http://purl.org/net/orth#
     protein/sIO_000750_0/P07118"/>
 <rdf:type rdf:resource="http://purl.org/net/orth#Gene"/>
</rdf:Description>
```

Listing 5: Individual 'gene valS' output example.

The major performance limitation of the original SWIT version is the application of identity rules in data integration scenarios. For example, InParanoid is a database of orthology relations between genes from different species. This database is distributed as a set of files, and each file contains the relations between the genes of two particular species. Consequently, a concrete gene may appear in multiple files. The application of identity rules is fundamental in this scenario to avoid redundancy. However, the full conversion of InParanoid database required of 919 hours (around 38 days) with the original SWIT method. Therefore, we considered mandatory to improve the execution time to enhance the utility and productivity of SWIT framework.

2.2. The new SWIT algorithm

High-Performance Computing (HPC) and code modernization techniques (Mike, 2017) allows to work at two design levels: Node-level and Cluster-level (Vladimirov, 2013). The Node-level includes improvement actions such as scalar

[3]https://virtuoso.openlinksw.com/

tuning, vector instructions, memory management and parallelization. Cluster-level optimizations add the use of multiple servers in a synchronized way. In this work, we apply node-level optimizations to develop the new SWIT algorithm, hereinafter HPC SWIT, whose goals is to decrease the execution time. In each step the improvements made are:

Scalar tuning: the SWIT kernel has been fully reimplemented to properly handle the low level features of the new SWIT algorithm, and its code has been refactored from scratch in C/C++. However, the change of implementation language is not a major optimization. The computational complexity has changed due to refactoring in HPC SWIT (see Table 1). The major improvement in complexity is the execution of the attribute rules (Arch2Prop) during the creation of individuals.

Table 1: Computational complexity in the creation of individuals, properties and relationships for the different versions of SWIT. In HPC SWIT, we improved the performance by including the property establishment into the individual creation.

Version	Individuals	Properties	Relationships
Original	$O(nmk)$	$O(n+m(k(m'+k')))$	$O(n+m(k(n'+m'k')))$
HPC	$O(nmk(n'm'))$	-	$O(n+m(k(n'+m'k')))$

Vectorization: We have vectorized some SWIT methods by inserting pragmas like *#pragma simd* when the compiler detected an incorrect data dependence or a multi-versioned chunk of code.

Memory optimizations: We have rearranged how data are stored in the kernel of HPC SWIT since it determines how fast identity rules are executed. To handle the search for individuals, we created two differently hashed hash maps. One is hashed by the line number (position) of the entity in the input file and its ontology class, resulting in a hash table easy to compute. This structure acts as a *greedy* search algorithm, so it might miss in finding the entity. The second one has slower access time, because the hash function used the concatenation of the individual's ontology class and the IRI (Internationalized Resource Identifier), which sometimes is not trivial to generate. This map grants no misses upon searching the individual. When SWIT retrieves an entity, it attempts to access the first map before proceeding to use the slower one.

The execution of identity rules has been fully redesigned. The original version of SWIT used SPARQL queries to detect redundant data, which is not efficient for large datasets. These queries take more time to detect equivalent data as the ontology grows by the addition of new individuals. SPARQL queries are $O(n)$ for conditional searches. The new algorithm uses two hash maps to retrieve the individuals when identity rules are applied. One hash map is used for AND rules and another one is used for OR ones. This technique allows us to find individuals in complexity $O(1)$ in the best case or $O(m) : m \ll n$ for the worst one. The new algorithm detects equivalent data as follows:

1. Initialization and creation of the individual structure.
2. Process the mapping rules and update the individual with its properties and relationships.
3. Check if identity rules must be applied. If so, HPC SWIT extracts each value from the individual according to the rule requirements. Otherwise, it stores the individual into the system.
4. If the identity rule has an *AND* condition, a concatenation with all the values is created and hashed. If it has an *OR* condition, each extracted value is hashed separately, so several pointers in different positions will stand for the same entity.
5. Once the hash function returns the position, HPC SWIT accesses it in the corresponding hash map. At this point, two possible scenarios may occur:
   (a) The accessed slot is free: HPC SWIT knows that no individual is equivalent to the one processed at the moment with complexity $O(1)$, whereas the original implementation had $O(n)$, being n the total number of individuals created so far by the transformation process.
   (b) The slot is not free: Hashing might deliver non-unique positions producing ‘fake’ collisions. The fact that individuals share a slot does

not necessary mean that they are similar, so we keep a small vector storing the collided individuals. Then, the desired individual is found in such vector with complexity $O(m)$, whereas the complexity with the SPARQL-based (SWIT original version) execution was $O(n) : m < n$.

The four described hash map structures store arrays of pointers to individuals and not copies, hence coherence is maintained between them if updates are made on individuals.

Parallelization: The parallelization is achieved at process level by using the GNU Parallel tool, which permits running one instance of HPC SWIT with one file per thread. Hence, parallel computation is only ran when multiple files are specified. We have considered using other parallel APIs like OpenMP, which allows explicit parallelization at code level. However, the main problem of using these APIs is that SWIT algorithm is memory intensive: the algorithm makes many memory petition calls (*malloc*, *calloc*, *new*), which require mutual exclusion to be thread-safe. When a thread performs a memory request, the rest of the running threads are stalled until the memory call is resolved, slowing down the execution of the application.

I/O Bottleneck: In order to avoid I/O bottleneck due to how fast the new implementation attempts to write on disk, we enabled the compression of the output data stream using *DEFLATE*, which relies on the standard *zip* format.

2.3. Additional improvements to the SWIT framework

- Tabular data: Many open datasets are available in the Internet in tabular format. We have developed a method for the automatic extraction of XML Schemas from tabular files, which is able to generate headers in case they are not present in the tabular files. Consequently, tabular data is transformed in SWIT as XML data.

- Flexible URIs: URIs are the identifiers of the data in the SWIT repositories. The original SWIT method used a base URI for the whole dataset.

Consequently, the transformation of data whose value was a URI required the generation of a local URI and the inclusion of direct links to external datasets was difficult. The new method permits to include a URI pattern for the entities to be generated by each rule. If this URI is not specified, then the base URI for the data transformation process is applied. The new URI for an entity can use parameters. Its structure consists of a prefix plus a series of parameters, whose value would come from the input data entity to which it is bound. For example, `http://www.identifiers.org/taxonomy/$1` would mean that the prefix would be `http://www.identifiers.org/taxonomy`, which would be followed by the value of \$1. This parameter could be bound to the field with the identifier of the organism. If the organism is *Homo sapiens*, whose identifier is 9606, then the URI would be `http://www.identifiers.org/taxonomy/9606`.

- Support services: Some transformation processes require to obtain additional information from external resources. This is handled in SWIT as additional services. These services are modeled as support services and their invocation has to be specified in the corresponding mapping rule. So far, we have created two support services:

  – Advanced URI: The desired URI might be retrieved from an external resource. For example, we wish to use SNOMED CT[4] URIs for clinical concepts but the source data only contains the label of the concept. Thus, a generic service for getting the URI associated with a label in an OWL file (i.e., SNOMED CT) is available now in SWIT.

  – External link: Linked Open Datasets are expected to have links to external datasets, but those links are not usually available in the source data. SWIT offers a service to query SPARQL endpoints or RDF files to get mappings, which are represented as *owl:sameAs*

[4]`https://www.snomed.org/snomed-ct`

statements.

2.4. Experimental evaluation method

We have evaluated the new SWIT framework by transforming data from two different domains, namely, orthology and movies. The datasets for each use case are described in the next subsections. A *git* repository[5] is available to ease the reproducibility of our experiments.

The experimental evaluation has consisted of transforming the datasets with the original and the new framework, and comparing the transformation times. The experiments were run in a server whose specifications are: 2 chips of Intel® Xeon® E5-2698 v4 with 20 cores each (Hyper-threading of 2), making a total of 40 physical cores or 80 virtual cores, running at 2,2 GHz and 128 GB RAM DDR4. The OS is CentOS Linux release 7.2.1511, and the GNU Compiler Collection (GCC) version is 7.3.0-2.

2.4.1. Use case 1: orthology data

Orthology is the life sciences field that investigates evolutionary relations between genes. Such relations are relevant for health research since the conservation of functions across species is usually inferred for genes holding a one-to-one orthology relation, that is, genes that diverged from a common ancestor by a speciation event.

We have used orthology datasets transformed by us in a previous work (Fernández-Breis et al., 2016):

- Inparanoid[6] (Sonnhammer & Östlund, 2014): This resource stores orthology relations between pairs of species. It provides one file per pair of species. We have used the InParanoid files for the species *S.pombe*, *C.elegans* and *G.gorilla*, whose sizes are 49 MB, 318 MB and 371 MB. These three data collections include 50, 233 and 174 files respectively.

[5]`https://bitbucket.org/Neobernad/swit-test`
[6]`http://inparanoid.sbc.su.se/download/8.0_current/Orthologs_OrthoXML/`

- TreeFam[7] (Schreiber et al., 2013): This resource stores groups of orthologs for several genomes. We have used the whole database, which is distributed in one 612 MB file.

- OMA[8] (Altenhoff et al., 2010, 2017): This resource also stores groups of orthologs for several genomes. We have used the whole database, which is distributed in one $1,5$ GB file.

The datasets have been mapped to the Orthology Ontology[9]. We have also used mapping rules[10], and identity rules[11]. The identity rules are needed in InParanoid because the same gene can appear in different files.

2.4.2. Use case 2: Internet Movie Database

The Internet Movie Database (IMDB) [12] offers a series of datasets about movies, actors or TV series. The following datasets[13] (more than 2,7 GB) have been used:

- General information about IMDB contents: information such as titles, language, type of titles, start/end year for tv series or genres. This information is distributed in two files: title.akas.tsv (179,9 MB) and title.basics.tsv (448,9 MB).

- Directors and writers are included in the title.crew.tsv file (166,7 MB).

- The episodes information of TV series is contained in the title.episode.tsv file (90,5 MB).

- The cast of the different movies and TV series is contained in the title.principals.tsv file (1,34 GB).

[7] `http://www.treefam.org/download`
[8] `https://omabrowser.org/oma/current/`
[9] `http://purl.bioontology.org/ontology/ORTH`
[10] `http://sele.inf.um.es/swit/ortho/mappingsOrthoXML.xml`
[11] `http://sele.inf.um.es/swit/ortho/identity.xml`
[12] `https://www.imdb.com`
[13] `https://www.imdb.com/interfaces/`

- Information about ratings and votes is available in the title.ratings.tsv file (14,8 MB).

- Personal information of actors, directors, etc. is offered in the name.basics.tsv file (535 MB).

The datasets have been mapped to an extended version[14] of the Movie Ontology[15]. We have also used mapping rules[16], and identity rules[17]. The identity rules are needed because the information about a particular content or person is distributed in various files.

3. Results

3.1. Impact of identity rules on performance

The negative impact of identity rules on performance for the original SWIT can be observed in Figure 2, which shows execution times with identity rules (Original V. W.I) and without them (Original V. N.I). This figure also includes, for comparison purposes, the execution times for HPC SWIT (labeled as 'HPC V.') with and without identity rules. These results show that the algorithm solves the bottleneck of the original SWIT to avoid redundant data.

3.2. Sequential runs

We have evaluated the performance of original SWIT version versus HPC SWIT considering only single core executions and making use of identity rules. The execution time for the InParanoid datasets is shown in Figure 3, the ones for OMA and TreeFam are shown in Figure 4 and for IMDB datasets in Figure 5.

[14]`http://sele.inf.um.es/swit/imdb/movieontology.owl`
[15]`http://www.movieontology.org`
[16]`http://sele.inf.um.es/swit/imdb/imdb-mappings.xml`
[17]`http://sele.inf.um.es/swit/imdb/identity/`

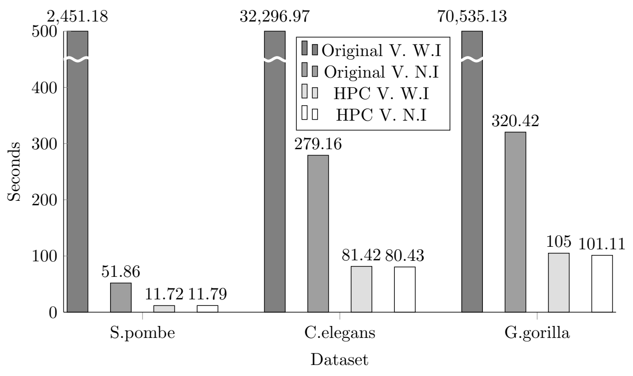

Figure 2: SWIT execution comparison using 3 datasets of InParanoid database. Original V. W.I and HPC V. W.I stands for the original version and optimized sequential version of SWIT with identity rules, whilst Original V. and HPC V. N.I denotes the usage of these versions when no identity rules are applied.

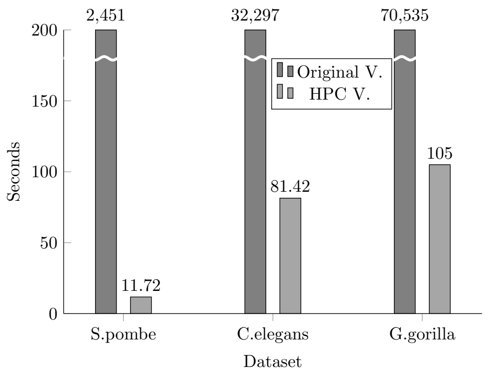

Figure 3: Execution time comparison from original version of SWIT (Original V.) and the sequential HPC version (HPC V.) from *S.pombe*, *C.elegans* and *G.gorilla* datasets.

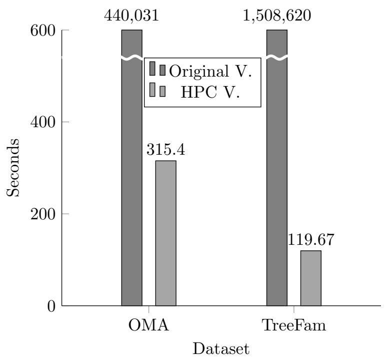

Figure 4: Execution time comparison from original version of SWIT (Original V.) and the sequential HPC version (HPC V.) from *OMA* and *TreeFam* databases.

The execution time of SWIT HPC reaches a minimum speed up of 209x for *S.pombe*, 396x for *C.elegans* collection and a maximum of 671x for *G.gorilla* in these three InParanoid files. For the OMA database HPC SWIT is 1395 times faster than the original one. The acceleration for TreeFam reaches 12606x, which means that an execution with a duration of two weeks is decreased to around 2 minutes, as the larger the files are the worse the original SWIT performs.

For the IMDB use case we firstly need to perform a conversion from TSV source data files to XML SWIT's input format. Figure 5 shows the execution time of three of their datasets, including the time required for data conversion. This conversion time makes SWIT HPC to reach lower speed-up values: 9.21x for *title.crew*, 6.80x in *title.episode* and 7.85x for *title.ratings*.

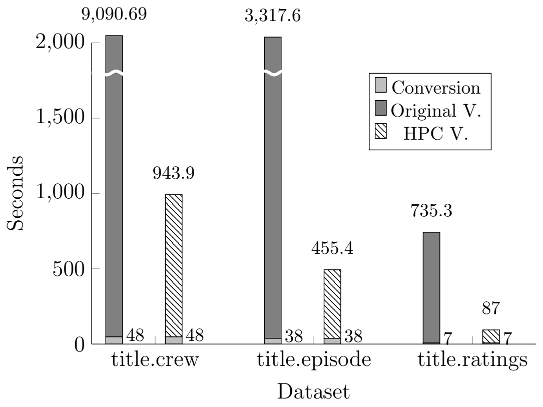

Figure 5: Execution time comparison between the original SWIT and the sequential optimized version for *title.crew*, *title.episode* and *title.ratings* IMDB datasets.

3.3. Parallel runs

We have also done parallel executions for InParanoid and IMDB. No parallel executions were possible for OMA and TreeFam since they provide the database in one single file. We have employed a number of threads equal to the number of virtual cores, i.e 80. The execution times for *S.pombe*, *C.elegans* and *G.gorilla* species are displayed in Figure 6. A summary table of the obtained speed up is shown in Table 2. The IMDB results are shown in Figures 7 and 8, taking into account conversion and transformation times separately.

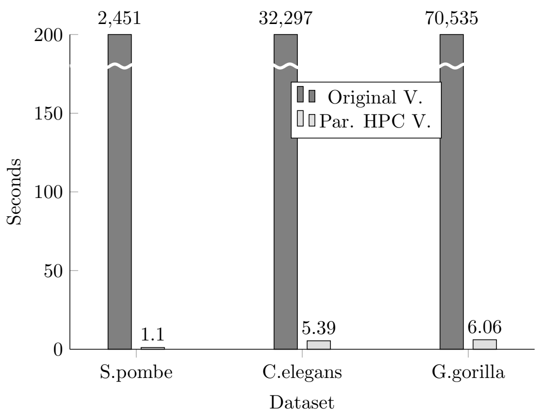

Figure 6: Execution time comparison from original SWIT version against parallel HPC version for *S.pombe*, *C.elegans* and *G.gorilla* collections, using identity rules.

Table 2: Speed up achieved among sequential and parallel HPC version versus original SWIT with InParanoid data.

Database	Dataset	Sequential speed up	Parallel speed up
InParanoid	*S.pombe*	209, 07x	1964.08x
	C.elegans	396, 67x	6195.86x
	G.gorilla	671, 79x	11187.17x
OMA		1395.14x	Not applicable
TreeFam		12606.01x	Not applicable

Additionally, the original SWIT algorithm required 38 days to process the whole InParanoid database. The size of InParanoid is 43 GB and it generated a 193 GB dataset in RDF format. HPC SWIT accomplishes this transformation in less than 1 hour ($\approx$ 55 minutes). Regardless I/O issues, the total speed up of the transformation is 1003.2x (see Table 3).

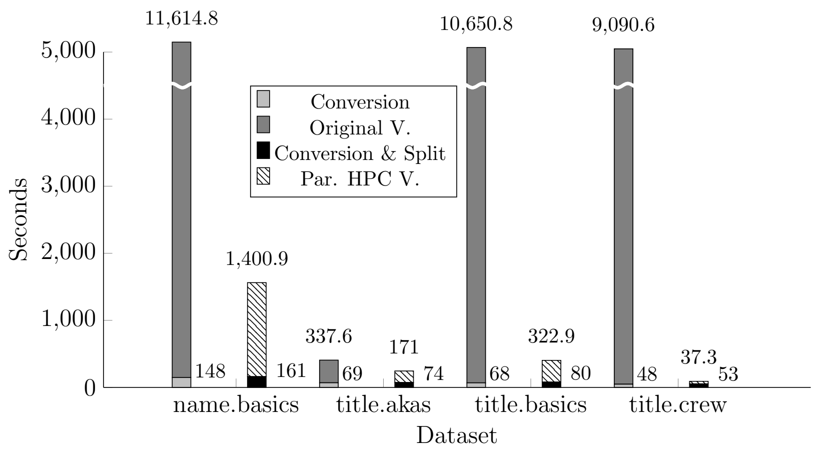

Figure 7: Execution time comparison from original version of SWIT and parallel HPC version for *name.basics*, *title.askas*, *title.basics* and *title.crew* datasets, using identity rules.

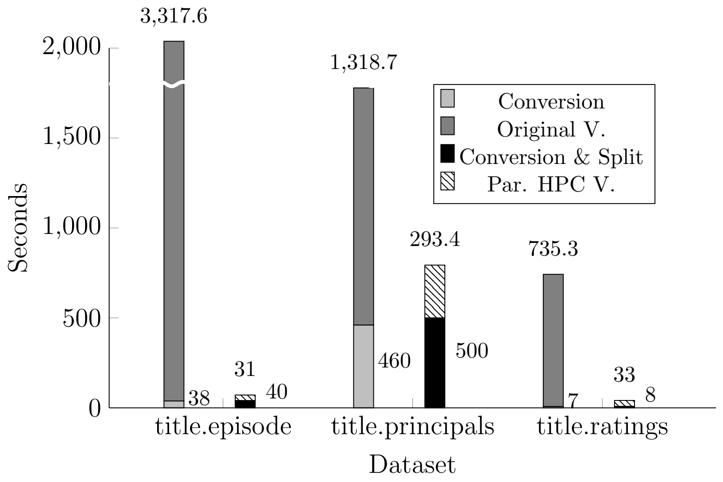

Figure 8: Execution time comparison from original version of SWIT and parallel HPC version for *title.episode*, *title.principals* and *title.ratings* IMDB datasets, using identity rules.

The transformation of IMDB datasets delivered several different speed ups. In order to compute each dataset in parallel, we split the files into several chunks of 5 MB of XML data. The conversion and split times are considered in the results labeled as *Conversion & Split*. The lowest speed-up was obtained for *title.akas*, 1.66x. The maximum one is 101x for *title.crew* dataset.

3.4. I/O Management

Since the size of the output data transformation might be huge (e.g., the InParanoid database transformation produces 193 GB of output data), we analyzed the I/O time consumption by varying the number of threads used in each transformation of the entire InParanoid database (Figure 9). The percentage of I/O usage increases with the number of threads, so we can empirically conclude that HDD access limits the parallel executions of HPC SWIT.

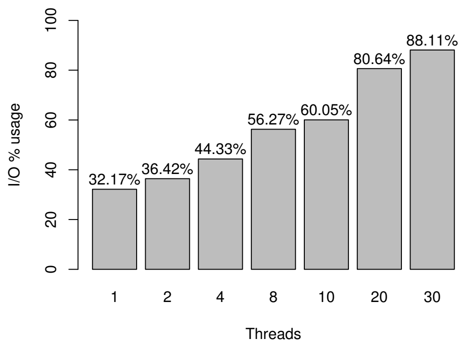

Figure 9: HPC SWIT: I/O usage (%) for InParanoid database varying the thread count.

To overcome this problem, we added a compression method (DEFLATE algorithm, standard *zip*) with a compression ratio of ≈27x for RDF/OWL files. Since the conversion of the InParanoid database generates an output of 193 GB and considering an ideal writing bandwidth of 120 MB/s in a standard HDD, writing this amount of data would take $(193 * 1024)/120 = 1646.93$ seconds (or 27 minutes). Taking into account the compression ratio reached, the new amount of bytes to be written in disk is shrunk to 6.9 GB, which is a volume that the HDD can handle easily in a short period of time ($(6.9 * 1024)/120 = 58.88$ seconds). Table 3 shows the acceleration working out I/O problems. With this optimization HPC SWITS transforms InParanoid in approximately 8 minutes and 56 seconds, reaching a total speed up of 6173.28x.

Table 3: Execution time comparison transforming the all the InParanoid data collections, solving the I/O bottleneck.

Version	Compression	Time (s)	Speed up	Time (hrs)
Original	No	$3,310,9$	Not applicable	919 (≈38 days)
HPC	No	$3,300.3$	1003.2x	0.91 (≈55 m)
HPC	Yes	536.3	$6,173.28$x	0.14 (≈8 m 56 s)

450 We analyzed the impact of compressing the output data on executions with less than 80 threads, since compression might be adding up an excessive overhead in some cases. Figure 10 shows the time required to complete the InParanoid transformation from 1 to 80 threads. The figure is split into two groups: (left) time without compression; (right) time with compression. The execution
455 of HPC SWIT on a single thread takes longer due to the compression overhead. However, from 2 threads onwards, shrinking the output data pays off for all the scenarios.

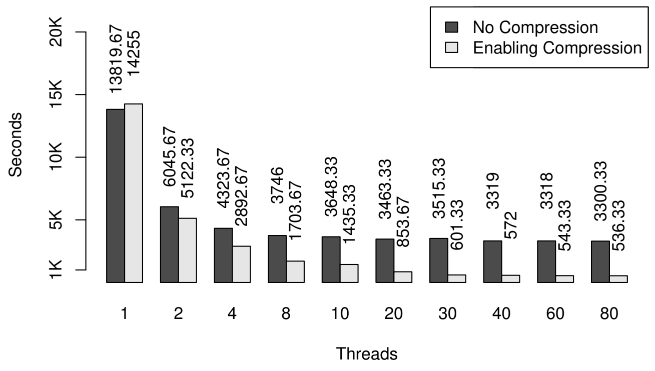

Figure 10: HPC SWIT version: Execution times for InParanoid database varying the thread count, enabling or disabling the compression algorithm.

Figure 11 shows the execution times in sequential and parallel (if applicable) for each database/dataset, calculating the time difference between a run using compression against the same run with compression disabled. This figure includes two additional datasets, the orthology related Hieranoid database (Kaduk et al., 2017) and a private dataset of colorectal cancer, which we have used in previous works (Fernández-Breis et al., 2013; Legaz-García et al., 2016). The results show that enabling compression pays off only when the input data is a large collection that can be processed in parallel (multiple files). For example, we achieved an improvement of 43.94% and 515.35% for Hieranoid and InParanoid execution times when they are processed in parallel. Compression strongly benefits InParanoid executions since InParanoid generates almost 200GB of output data and Hieranoid only produces 4.6GB.

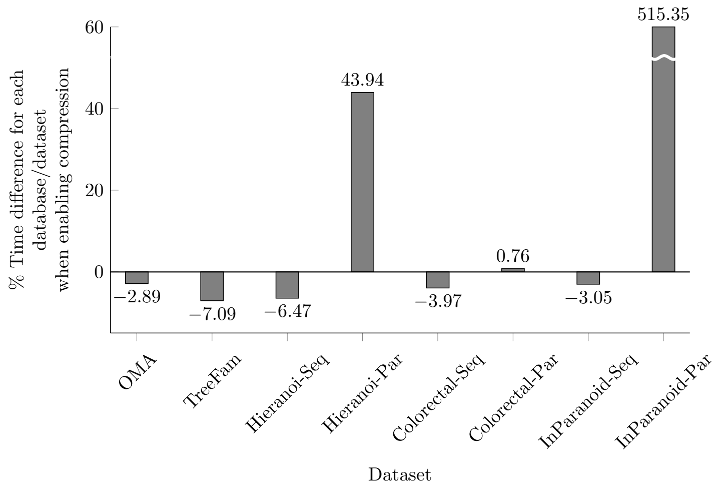

Figure 11: HPC SWIT: Time difference in percentage, for each dataset/database, between executions enabling compression or disabling it. The greater the percentage is, the faster the execution is when compression is enabled.

3.5. Karma and HPC SWIT performance

Karma (Dimou et al., 2014) is an ontology-driven tool capable of exploiting data schemas and ontologies by using user-defined mappings. Karma has similarities with SWIT, but also differences:

- Karma has to be used through a GUI.

- Karma rearranges the input data into a table-like appearance whilst SWIT processes XML data by using XPath queries.
- An input entity in SWIT may be considered an entity, attribute or relation depending on its semantic context, whereas Karma allows only one semantic type per input entity.
- Karma mapping rules are created in the GUI, but cannot be reused.
- SWIT offers detection of redundant data.

Figure 12 shows the execution times for Karma and SWIT with IMDB datasets. Karma is designed for small data, loading title.basics.tsv file (428MB) took more than 30 minutes (the upload time only took around 2 minutes), also the user interface slows down when using large scale data. The RDF transformation exceeded a time-out of 2 hours. For *title.episode* (86MB) and *title.ratings* (14MB) Karma run smoothly. Comparing the total time in each tool, the speed ups against Karma are: 23x for *title.basics*, 22x for *title.episode* and 11x for *title.ratings*.

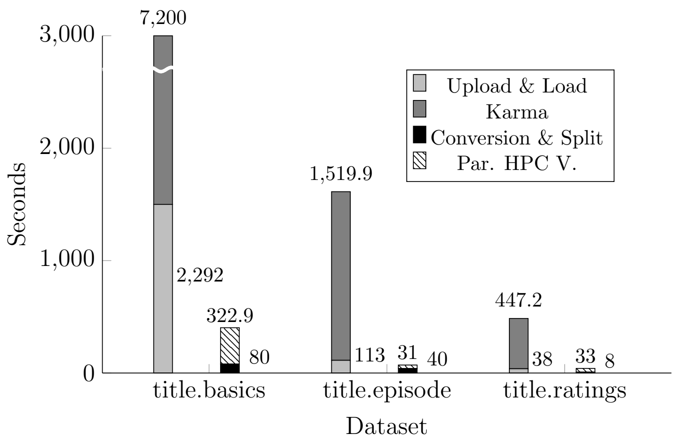

Figure 12: Karma and HPC SWIT performance, considering load and conversion time depending on the software used.

4. Discussion and Conclusions

4.1. General discussion

The success of initiatives such as Linked Open Data or FAIR will depend on the availability of efficient tools and methods for generating datasets rich in semantics, in which ontologies provide the meaning and logical constraints for the data.

The starting point for our work was the performance limitation of the original version of SWIT framework when large, distributed datasets have to be integrated. We have described the optimization of the SWIT algorithm by applying HPC techniques. The new data transformation process included in HPC SWIT has drastically reduced the transformation time.

The results show accelerations ranging from 175x to 11631x for orthology datasets. The transformation of the entire InParanoid database has decreased the execution time from 38 days to 55 minutes (1003x speed up). We have obtained a minimum speed up of 1.66x and a maximum of 101x for IMDB datasets. HPC SWIT time performance was penalized in the IMDB use case by the need of transforming TSV data into XML, which could be overcome in the future. Output data compression also showed to accelerate data transformation (6173x for InParanoid).

The performance comparison with Karma, which is the most similar software to SWIT considering ontology-driven data transformation, shows positive results for HPC SWIT, with speed up ranging from 11x to 23x for the datasets tested. Karma was not able to transform large datasets with a time-out of 7200 seconds (2 hours). HPC SWIT was able to transform such datasets.

4.2. Lessons learned

Our experience with the transformation of datasets of different size and different ways of distribution have allowed us to learn some lessons and to develop some heuristics to get the best possible results when using SWIT.

Parallel executions should be run on the maximum number of threads for small datasets whose overall size is less than $\approx$150 MB. However, if (1) the dataset is larger, (2) solid state disks are used for storage and (3) there are more than 10 cores, then the use of file compression is recommended. In that case the output transformations of SWIT might slow down the writing speed, thus increasing the execution time. On the contrary, if the storage unit is a hard drive disk, file compression is always recommended.

In order to parallelize the transformation of single file databases, we implemented a tool capable of splitting an XML file into several XML chunks. The practical experience in the two use cases was different. This approach worked fine for IMDB datasets but failed for orthology data. This was due to the properties of the OrthoXML format. A list of several gene and protein identifiers are grouped by species, followed by a list of orthology groups pointing to gene identifiers from the species defined above. In order to deliver a consistent output, the division into many XML files was done depending on the orthology groups and the species associated. As a result, multiple copies of the same species were created in different XML files. The volume of data is higher than the original size of the single file, therefore HPC SWIT required more time to process the dataset and the parallel execution was slower. Our recommendation for the providers of large datasets to be consumed in semantic formats is to use data formats with zero or few dependencies between different elements in its content facilitates a fast HPC-driven transformation. Having such fast transformation processes would also enable on-the-fly semantic data transformation process. The distribution format should be taken into account by the semantic consumers of the datasets because of the implication mentioned in the transformation of the datasets.

4.3. Limitations and further work

Further work will automate the selection of the optimal configuration for the transformation process that is, sequential/parallel, number of cores, use of compression, etc. For this purpose we will implement the heuristics described in the previous section. Another limitation regards the definition of the mappings used by SWIT to generate the datasets. The mappings need to be manually defined. Further work will adapt alignment methods available in our OntoEnrich framework (Quesada-Martínez et al., 2015) to suggest matches between the input schemas and the target ontology.

4.4. Conclusions

The semantic richness of the datasets is crucial for their interoperability. We have described a method for the efficient, semantics-rich transformation and integration of large datasets. High Performance Computing techniques have proven to be fundamental for the scalability and performance of a semantic web framework like SWIT. The evaluation of the method shows that the method is domain-independent and that the speed up is affected by properties of the source datasets.

5. Acknowledgements

This work has been partially funded by the Spanish Ministry of Economy, Industry and Competitiveness, the European Regional Development Fund (ERDF) Programme and by the Fundación Séneca through grants TIN2014-53749-C2-2-R, TIN2015-66972-C5-3-R, TIN2017-85949-C2-1-R and 19371/PI/14, respectively.

References

Abele, A., McCrae, J. P., Buitelaar, P., Jentzsch, A., & Cyganiak, R. (2017). Linking open data cloud diagram. `http://lod-cloud.net`. Accessed on July 23, 2018.

Altenhoff, A. M., Glover, N. M., Train, C.-M., Kaleb, K., Warwick Vesztrocy, A., Dylus, D., de Farias, T. M., Zile, K., Stevenson, C., Long, J. et al. (2017). The oma orthology database in 2018: retrieving evolutionary relationships among all domains of life through richer web and programmatic interfaces. *Nucleic acids research*, *46*, D477–D485.

Altenhoff, A. M., Schneider, A., Gonnet, G. H., & Dessimoz, C. (2010). Oma 2011: orthology inference among 1000 complete genomes. *Nucleic acids research*, *39*, D289–D294.

Bechhofer, S. (2009). Owl: Web ontology language. In *Encyclopedia of database systems* (pp. 2008–2009). Springer.

Bernabé-Díaz, J. A., Legaz-García, M. C., García, J. M., & Fernández-Breis, J. T. (2018). Application of high performance computing techniques to the semantic data transformation. In *World Conference on Information Systems and Technologies* (pp. 691–700). Springer.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific american*, *284*, 34–43.

Bourne, P. E. et al. (2015). Biomedicine as a data driven science. In *National Data Integrity Conference-2015*. Colorado State University. Libraries.

Cafaro, M., Pulimeno, M., & Epicoco, I. (2018). Parallel mining of time-faded heavy hitters. *Expert Systems with Applications*, *96*, 115–128.

Čerāns, K., & Būmans, G. (2015). Rdb2owl: a language and tool for database to ontology mapping. In *Proceedings of CAiSE FORUM*.

Dimou, A., Vander Sande, M., Slepicka, J., Szekely, P., Mannens, E., Knoblock, C., & Van de Walle, R. (2014). Mapping hierarchical sources into rdf using the rml mapping language. In *Semantic Computing (ICSC), 2014 IEEE International Conference on* (pp. 151–158). IEEE.

Erling, O., & Mikhailov, I. (2009). Rdf support in the virtuoso dbms. In *Networked Knowledge-Networked Media* (pp. 7–24). Springer.

Fernández-Breis, J. T., Chiba, H., Legaz-García, M. C., & Uchiyama, I. (2016). The Orthology Ontology: development and applications. *Journal of biomedical semantics*, *7*, 34.

Fernández-Breis, J. T., Maldonado, J. A., Marcos, M., Legaz-García, M. C., Moner, D., Torres-Sospedra, J., Esteban-Gil, A., Martínez-Salvador, B., & Robles, M. (2013). Leveraging electronic healthcare record standards and semantic web technologies for the identification of patient cohorts. *Journal of the American Medical Informatics Association*, *20*, e288–e296.

Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, *35*, 137–144.

Goble, C., & Stevens, R. (2008). State of the nation in data integration for bioinformatics. *Journal of biomedical informatics*, *41*, 687–693.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, *5*, 199–220.

Hu, W., Qiu, H., Huang, J., & Dumontier, M. (2017). Biosearch: a semantic search engine for bio2rdf. *Database*, *2017*.

Huang, J.-Y., Lange, C., & Auer, S. (2015). Streaming Transformation of XML to RDF using XPath-based Mappings. In *Proceedings of the 11th International Conference on Semantic Systems* (pp. 129–136).

Jirkovskỳ, V., Obitko, M., & Mařík, V. (2017). Understanding data heterogeneity in the context of cyber-physical systems integration. *IEEE Transactions on Industrial Informatics*, *13*, 660–667.

Kaduk, M., Riegler, C., Lemp, O., & Sonnhammer, E. L. (2017). Hieranoidb: a database of orthologs inferred by hieranoid. *Nucleic Acids Research*, *45*, D687–D690.

Kapłański, P., Seganti, A., Cieśliński, K., Chrabrowa, A., & Ługowska, I. (2017). Automated reasoning based user interface. *Expert Systems with Applications*, *71*, 125–137.

Khan, S., Liu, X., Shakil, K. A., & Alam, M. (2017). A survey on scholarly data: From big data perspective. *Information Processing & Management*, *53*, 923–944.

Le, B., Huynh, U., & Dinh, D.-T. (2018). A pure array structure and parallel strategy for high-utility sequential pattern mining. *Expert Systems with Applications*, *104*, 107–120.

Legaz-García, M. C., Miñarro-Giménez, J. A., Tortosa, M. M., & Fernández-Breis, J. T. (2016). Generation of open biomedical datasets through ontology-driven transformation and integration processes. *J. Biomedical Semantics*, *7*, 32.

Mike, P. (2017). What is code modernization? `https://software.intel.com/en-us/articles/what-is-code-modernization`. Accessed on July 23, 2018.

Musto, C., Basile, P., Lops, P., de Gemmis, M., & Semeraro, G. (2017). Introducing linked open data in graph-based recommender systems. *Information Processing & Management*, *53*, 405–435.

Oliveira, J., Delgado, C., & Assaife, A. C. (2017). A recommendation approach for consuming linked open data. *Expert Systems with Applications*, *72*, 407–420.

Quesada-Martínez, M., Mikroyannidi, E., Fernández-Breis, J. T., & Stevens, R. (2015). Approaching the axiomatic enrichment of the gene ontology from a lexical perspective. *Artificial Intelligence in Medicine*, *65*, 35–48.

Roldán-García, M. d. M., Uskudarli, S., Marvasti, N. B., Acar, B., & Aldana-Montes, J. F. (2018). Towards an ontology-driven clinical experience sharing ecosystem: Demonstration with liver cases. *Expert Systems with Applications*, *101*, 176–195.

Sakr, S., Wylot, M., Mutharaju, R., Le Phuoc, D., & Fundulaki, I. (2018). Distributed reasoning of rdf data. In *Linked Data* (pp. 109–126). Springer.

Schmitt, T., Messina, D., Schreiber, F., & Sonnhammer, E. (2011). Letter to the editor: SeqXML and OrthoXML: standards for sequence and orthology information. *Briefings in bioinformatics*, *12*, 485–488.

Schreiber, F., Patricio, M., Muffato, M., Pignatelli, M., & Bateman, A. (2013). Treefam v9: a new website, more species and orthology-on-the-fly. *Nucleic acids research*, *42*, D922–D925.

Sonnhammer, E. L., & Östlund, G. (2014). Inparanoid 8: orthology analysis between 273 proteomes, mostly eukaryotic. *Nucleic acids research*, *43*, D234–D239.

Vladimirov, A. (2013). Multithreaded transposition of square matrices with common code for intel xeon processors and intel xeon phi coprocessors. `https://colfaxresearch.com/multithreaded-transposition-of-square-matrices-with-common-code-for-intel-xeon-processors-and-intel-xeon-phi-coprocessors/`. Accessed on July 19, 2018.

Wangli, Y., Xueyun, Z., & Man, Y. (2017). Research on the transformation from relational model to rdf (s) model. *Microcomputer Applications*, *9*, 003.

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E. et al. (2016). The fair guiding principles for scientific data management and stewardship. *Scientific data*, *3*.